Euclid's Algorithm

In this lecture, we study the algebraic complexity of the classic Euclid's algorithm for polynomials, and the asymptotically fast half-gcd approach. This lecture is based upon [1, Chap. 2].

1 Euclid's Algorithm

Given two polynomials $P_0, P_1 \in \mathbb{R}[x]$, such that $\deg(P_0) > \deg(P_1)$. The Euclidean remainder sequence $P_0, P_1, \ldots, P_k$, $k \geq 1$, for these two poltnomials is given by the recurrence:

$$P_{i+1} := P_{i-1} - Q_i P_i, \tag{1}$$

where $\deg(P_{i+1}) < \deg(P_i)$ and P_k divides P_{k-1}. The claim is that $P_k = \texttt{GCD}(P_0, P_1)$, this follows from the observation that

$$\texttt{GCD}(P_{i-1}, P_i) = \texttt{GCD}(P_i, P_{i+1}).$$

Define $Q_i := \texttt{quo}(P_{i-1}, P_i)$, $P_{i+1} := \texttt{rem}(P_{i-1}, P_i)$, and $n_i := \deg(P_i)$. Note that $\deg(Q_i) = n_{i-1} - n_i$. We introduce the convenient notation of matrices to express the recursion. In this terminology, (1) can be expressed as

$$\begin{pmatrix} P_i \\ P_{i+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -Q_i \end{pmatrix} \begin{pmatrix} P_{i-1} \\ P_i \end{pmatrix}. \tag{2}$$

For succinctness, we will express the matrix on RHS as $\langle Q\_i \rangle$. and recursively

$$\begin{pmatrix} P_i \\ P_{i+1} \end{pmatrix} = \langle Q_i \rangle \cdots \langle Q_2 \rangle \langle Q_1 \rangle \begin{pmatrix} P_0 \\ P_1 \end{pmatrix}.$$

Define the 2×2 matrix M_{ij}, $0 \leq i < j < k$, as the matrix that transforms (P_i, P_{i+1}) to (P_j, P_{j+1}). Given a number k, let $M_{I(k)}$ denote the regular matrix that takes (P_0, P_1) to the pair $(P_{I(k)}, P_{I(k)+1})$, where $I(k)$ is the index such that

$$\deg(P_{I(k)}) \geq k > \deg(P_{I(k)+1}).$$

We will often say that $I(k)$ is the index that straddles k. We would sometimes use the explicit form $M_{I(k)}^{P_0, P_1}$ to emphasize the polynomials involved; if, however, the polynomials are clear from the context then we would use the simpler notation.

¶1. Extended Euclidean Algorithm From the extended euclidean algorithm it follows that

$$M_{0j} = \begin{pmatrix} s_j & t_j \\ s_{j+1} & t_{j+1} \end{pmatrix}.$$

¶2. Algebraic Complexity The algebraic cost of one step in Euclid's algorithm is $O(M'_A(n))$, where $M'_A(n)$ is the algebraic cost of multiplying two degree n polynomials; using the FFT-based algorithm, we know that $M'_A(n) = O(n \log n)$. Why is this? Using the standard high-school algorithm, we can compute the quotient Q_i in time $O(n)$. Thus the cost of computing P_{i+1} is dominated by the cost of computing the product $P_i Q_i$. Also, $k \leq n_1$, as the degree sequence $(n_0, n_1, n_2, \ldots, n_k)$ is strictly decreasing. Thus the algebraic cost of the algorithm is $O(M'_A(n)n)$; more precisely, it is $O(M'_A(n_1)n_1)$, that is independent of the degree of P_0.

We next see an asymptotically fast version that takes $O(M'_A(n) \log n)$ time.

2 Asymptotically Fast GCD Algorithm

The improvement is based upon the following observation: suppose we want to store the euclidean remainder sequence $(P_0, \ldots, P_k)$ (say for the purpose of evaluation); then we would need roughly

$$\sum_{i=0}^{k} n_i \leq \sum_{i=1}^{n} i = n(n-1)/2$$

space to store the coefficient sequence; but this is can be reduced by observing that the quotients take less space as

$$\sum_{i=1}^{k} \deg(Q_i) = \sum_{i=1}^{k} (n_{i-1} - n_i) = n_0 - n_k \leq n_0 = n.$$

Thus we should focus on computing the quotients. [1]

To get the desired improvement of $O(M'_A(n) \log n)$ it is clear that we have to go from the pair (P_0, P_1) to a pair (P_i, P_{i+1}) such that

$$n_i \geq n/2 \geq n_{i+1}$$

i.e., reduce the degree by half rather than by one. If this could be done, then we would clearly need $\log n$ steps to find the gcd. With this in mind, we define the **half-gcd** problem (HGCD): given $P_0, P_1 \in \mathbb{R}[x]$ as above, compute a matrix $M := \mathtt{hGCD}(P_0, P_1)$ such that if

$$\begin{pmatrix} P_2 \\ P_3 \end{pmatrix} = M \begin{pmatrix} P_0 \\ P_1 \end{pmatrix}$$

then $\deg(P_2) \geq n/2 > \deg(P_3)$, i.e., the degrees of P_2 and P_3 straddle $n/2$.

Given two polynomials P_0, P_1, suppose we could compute $\mathtt{hGCD}(P_0, P_1)$ in time $T'(n)$ then we claim that we can compute their gcd in roughly the same time.

```
co-GCD
INPUT: Two degree polynomials P0P1 ∈ ℝ[x].
OUTPUT: A matrix M such that
      ( GCD(P0, P1) )       ( P0 )
      (      0      )  = M  ( P1 ) .
1.  Compute M1 := hGCD(P0, P1).
2.  Recover P2, P3 using M1:
        ( P2 )        ( P0 )
        ( P3 )  = M1  ( P1 ) .
3.  If P3 = 0 then return M1 else
        Do one Euclid-step to get P3, P4 using (2). Let ⟨Q⟩ be the matrix involved.
4.  If P4 = 0 then return ⟨Q⟩M else
        Recursively compute M2 := GCD(P3, P4).
        Return M2⟨Q⟩M1.
```

¶3. Complexity: Let $G(n)$ be the complexity to compute the co-GCD, and $\mathtt{hGCD}(n)$ the complexity to compute $\mathtt{hGCD}$. Then we have the following recursion:

$$G(n) = \mathtt{hGCD}(n) + O(M'_A(n)) + G(n/2).$$

Assuming that $\mathtt{hGCD}(n) = \Omega(M'_A(n))$, and $\mathtt{hGCD}(\alpha n) \leq \alpha \mathtt{hGCD}(n)$, for $\alpha > 0$, it follows that

$$G(n) = O(\mathtt{hGCD}(n)).$$

[1] We have only shown that the quotient sequence takes less space, but it is not clear that the bit-size of the coefficients is smaller or comparable to the bit-size of the coefficients in the remainde sequence. We defer this question till later.

2.1 Polynomial Half-GCD

Let $A, B \in \mathbb{R}[x]$ be two polynomials s.t. $\deg(A) > \deg(B)$. Let

$$A_0 := A\,\textbf{quo}\,x^k \text{ and } A_1 := A\,\textbf{mod}\,x^k;$$

similarly, define B_0 and B_1; basically, we have $A = x^k A_0 + A_1$. The idea behind the half-gcd algorithm is that it is possible to compute a substantial number of the quotients from the quotient sequence for A_0 and B_0. The follwing lemma makes it precise:

LEMMA 1. *For two polynomials, A, B, $n = \deg(A) > \deg(B)$, and for any $k \in \{0, 1, \ldots, n\}$, define A_0, B_0 as above. Then*

$$M^{A,B}_{I((n+k)/2)} = M^{A_0,B_0}_{I((n-k)/2)}.$$

That is, the quotient sequence (A, B) agrees with the quotient sequence of (A_0, B_0) until the point where the degree in the remainder sequence of the latter pair falls below $\deg(A_0)/2$.

Proof. The proof is illustrated in Figure 1. Basically, in P_{i+1} we loose $n_{i-1} - n_i$ coefficients common to P_{i-1} and P_i; in Figure 1, this is shown by the red line segments at level i, , corresponding to the quotient Q_i, which is equal to the blue line segment at level $i + 1$. Since we are loosing equal number of terms from the front and the end, we do not have any common coefficients when $\deg(P_i) < k + \deg(A_0)/2 = (n + k)/2$.

Q.E.D.

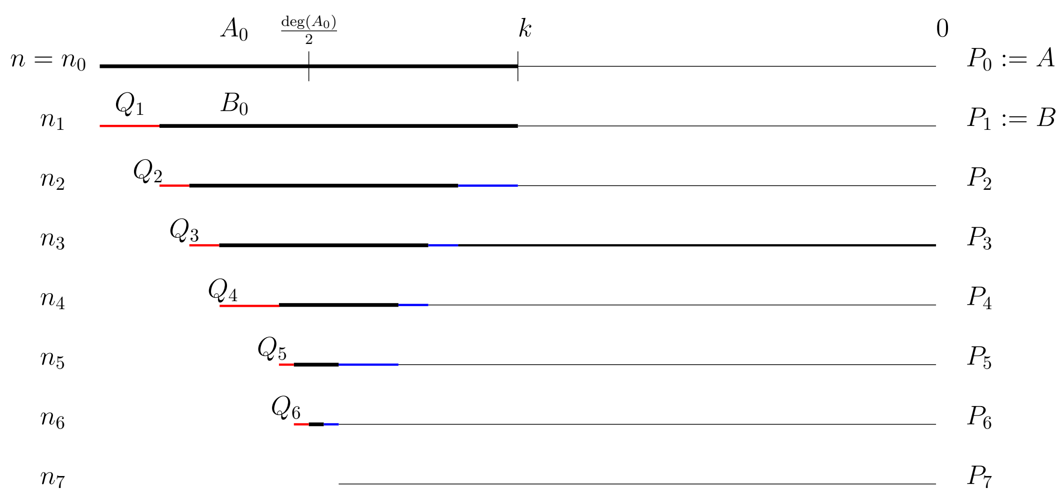

Figure 1: Illustration of the remainder sequences of (A, B) and (A_0, B_0); the coefficients common to both sequences are shown in bold black-line segments.

LEMMA 2. *Let $R := A\,\textbf{mod}\,B$ and $R_0 := A_0\,\textbf{mod}\,B_0$. If $\deg(A) - \deg(B) \leq \deg(B) - k$, or equivalently $\deg(A_0) < 2\deg(B_0)$, then*

$$A\,\textbf{quo}\,B = A_0\,\textbf{quo}\,B_0$$

and R and $x^k R_0$ agree in all coefficients of degree $\geq k + \deg(A) - \deg(B)$.

Proof. The condition implies that the quotient $A\,\textbf{quo}\,B$, which has degree $\deg(A) - \deg(B)$, is dependent only on the first $\deg(B) - k$ coefficients of B, i.e., only on B_0. Since $\deg(B) - k \geq \deg(A) - \deg(B)$, the coefficients in the remainder corresponding to the excess coefficients, namely $\deg(B) - k - (\deg(A) - \deg(B))$, in B_0 contribute to the remainder; the degrees of the coefficients are from $\deg(B) - 1$ down to $\deg(B) - (2\deg(B) - k - \deg(A)) = \deg(A) - (\deg(B) - k)$; these coefficients in the remainder are thus not affected by B_1 and A_1. **Q.E.D.**

We can now describe the half-gcd algorithm in detail:

```
Half-GCD Algorithm:  HGCD(A, B)
```
INPUT: $A, B \in \mathbb{R}[x]$, $n := \deg(A) > \deg(B)$.
OUTPUT: The matrix $M^{A,B}_{I(n/2)}$.
1. $m \leftarrow \deg(A)/2$.
   If $\deg(B) < m$ then return I_2.
2. $R \leftarrow \texttt{hGCD}(A_0, B_0)$.
   $\begin{pmatrix} A' \\ B' \end{pmatrix} \leftarrow R \begin{pmatrix} A \\ B \end{pmatrix}$.
3. If $\deg(B') < m$ then return R.
4. $\begin{pmatrix} C \\ D \end{pmatrix} \leftarrow \langle Q \rangle \begin{pmatrix} A' \\ B' \end{pmatrix}$.
5. $k \leftarrow 2m - \deg(C)$.
   ◃ *We want* $\deg(C_0)/2 \geq \deg(C) - m$,
   ◃ *i.e.,* $\deg(C) - k \geq 2(\deg(C) - m)$ *or* $k \leq 2m - \deg(C)$.
6. $S \leftarrow \texttt{hGCD}(C_0, D_0)$.
7. Return $S\langle Q \rangle R$.

We have the following bound on its complexity:

$$\texttt{hGCD}(2m) = 2\texttt{hGCD}(m) + O(M'_A(2m)),$$

which gives us the result $\texttt{hGCD}(m) = O(M'_A(m) \log m)$.

The correctness of the algorithm follows from Lemma 1 and a simple inductive argument; the variables used in the algorithm are illustrated in Figure 2.

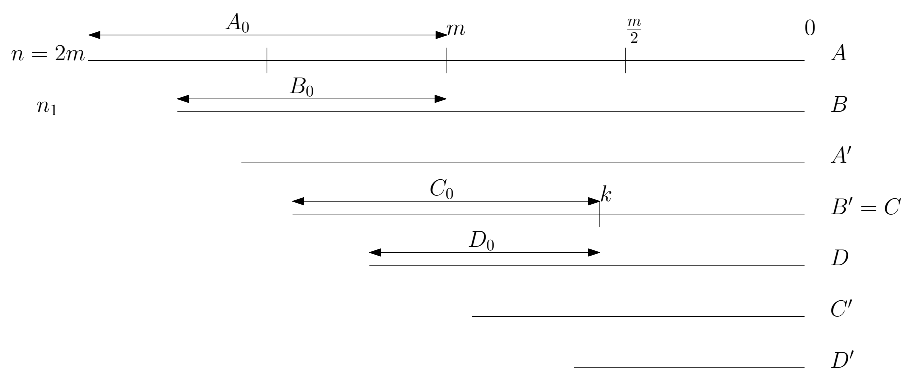

Figure 2: A run of the half-gcd algorithm.

References

[1] C. K. Yap. *Fundamental Problems of Algorithmic Algebra*. Oxford University Press, 2000.