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.