Zero Dimensional Regular Chains

Regular Chains play a fundamental role in polynomial system solving. Namely, they can encode the generic points of the irreducible components of algebraic varieties [3]. Of particular interest in practice is when these varieties are zero dimensional (i.e., finite). For instance, the authors of [1] have developed a probabilistic and modular algorithm for solving zero-dimensional polynomial systems with rational coefficients. Their algorithm requires to invert polynomial matrices modulo regular chains. For sufficiently large problems, this operation is a bottleneck, mainly due to memory consumption when testing the invertibility of an element modulo a regular chain.

The Leverrier-Faddeev Algorithm

The Leverrier-Faddeev [2] algorithm is a method for finding a matrix inverse that only does one division but requires repeated matrix multiplication.

Consider the characteristic polynomial of the \( m \times m \) matrix \( A \):

\[ p(\lambda) = \det (\lambda I - A) = \lambda^m - a_1 \lambda^{m-1} - \cdots - a_{m-1} \lambda - a_m. \]

An expression for the inverse of \( A \) is given by evaluating \( p(A) \), multiplying by \( A^{-1} \) and re-arranging terms:

\[ \begin{align*}
0 &= A^m - a_1 A^{m-1} - \cdots - a_{m-1} A - a_m \\
A^{-1} a_m &= A^{m-1} - a_1 A^{m-2} - \cdots - a_m \\
A^{-1} &= (A^{-1} - \sum_{i=1}^{m-1} a_i A^{i-1}) a_m. \end{align*} \]

We express this as a function of \( m \) using \( s_k \)'s by "baby step giant step":

\[ p(A) = \left( \prod_{i=0}^{m-1} A_i \right) N_i = \prod_{i=0}^{m-1} (A_i + a_i)^{N_i} \]

The complexity of the \( s_k \)'s by "baby step giant step":

Store \( M_0, M_1, \ldots, M_i = A^{i+1}, \ldots, A^m \) on the fly (repeatedly multiplying by \( A^{-1} \), without storing).

Get the \( \langle \rangle \)'s in blocks by \( \langle \rangle=M_i N_i = \text{tr}(\langle \rangle A_i A_i^{-1}) \) taking \( 0 \leq i, j, k \leq m \) for \( m=8 \). For example, number of \( \langle \rangle \)'s for the traces is \( 2m^3 \).

Expand \( \langle \rangle \)'s:

\[ p(A) = \left( \sum_{i=0}^{m-1} A_i \right) = \left( \sum_{i=0}^{m} a_i A_i \right) = \sum_{i=0}^{m} N_i = \sum_{i=0}^{m} \frac{m!}{(i+1)!} \sum_{i=0}^{\infty} a_i A_i \]

The complexity is given by \( \sum_{i=0}^{m} N_i = \sum_{i=0}^{m} \frac{m!}{(i+1)!} \sum_{i=0}^{\infty} a_i A_i \).

Using Leverrier-Faddeev Recursively

Use Leverrier-Faddeev algorithm to find \( a_m^{-1} \) recursively. For a zero-dimensional regular chain with coefficients in the field \( K \), the following recurrence:

\[ m_f : K[x_1, \ldots, x_n] / (T_1, \ldots, T_n) \to K[x_1, \ldots, x_n] / (T_1, \ldots, T_n) \]

such that \( m_f([g]) = [f] \cdot [g] = [f g] \) (or more simply: \( m_f([g]) = \sqrt{g} T \)). Since \( K[x_1, \ldots, x_n] / T \) finite dimensional it has a finite monomial basis \( B \). We can thus represent \( m_f \) by its matrix with respect to this basis. The multiplication matrix satisfies \( m_f \cdot m_f = m_f \) and thus we can find the inverse of \( a_m \) by inverting its corresponding multiplication matrix.

Space Complexity

For Leverrier-Faddeev. Let \( F(m, d_1, \ldots, d_n) \) be the number of field elements required to invert an \( m \times m \) matrix modulo a regular chain \( T = (T_1, \ldots, T_n) \) with \( d_i = \text{deg}(T_i) \). Assuming completely dense input we have

\[ F(m, d_1, \ldots, d_n) = \sqrt{m} \cdot m \cdot m \cdot d_i \cdot \ldots \cdot d_n \]

input and \( M_i \)s traces

recursive call

expansion

For GCD based Algorithm. Here one follows the method of Bareiss testing invertibility by using an Euclidean-like algorithm. In [4] the space complexity for this is given by (setting \( \delta = \prod \text{deg}(T_i) \) and otherwise reusing the above notation):

\[ 2m^2 \delta + O(2m^2) \sum_{i=0}^{m-1} (d_i^{-2} \cdot \delta) \]

field elements.

Experimental Results

We compare two approaches: recursive Leverrier-Faddeev algorithm and the existing (Bareiss based) method. We choose a random dense regular chain \( T \subseteq K[x_1, \ldots, x_n] \) with \( \text{deg}(T) = 6 \), varying \( m \) and \( n \) and using the above notation:

\[ 2m^2 \delta + O(2m^2) \]

\( \sum_{i=0}^{m-1} (d_i^{-2} \cdot \delta) \)

field elements.

Acknowledgements


