A fast algorithm for solving linearly recurrent sequences

Seung Gyu Hyun, Stephen Melczer, Catherine St-Pierre

Abstract

We present an algorithm which computes the D^{th} term of a sequence satisfying a linear recurrence relation of order d over a field $\mathbb{K}$ in $O(\mathsf{M}(\bar{d})\log(D) + \mathsf{M}(d)\log(d))$ operations in $\mathbb{K}$, where $\bar{d} \leq d$ is the degree of the squarefree part of the annihilating polynomial of the recurrence and M is the cost of polynomial multiplication in $\mathbb{K}$. This is a refinement of the previously optimal result of $O(\mathsf{M}(d)\log(D))$ operations, due to Fiduccia.

Overview. Consider a sequence $(a_i)_{i\geq 0}$ with entries in a field $\mathbb{K}$ that is generated by the recurrence

$$a_{i+d} = \sum_{j=0}^{d-1} c_j a_{i+j} \tag{1}$$

for all $i \geq 0$, where $c_0 \neq 0$. Given initial conditions a_j, for $0 \leq j \leq d-1$, along with the annihilating polynomial $P = x^d - \sum_{j=0}^{d-1} c_j x^j$, we are interested in the complexity of computing one term a_D of the sequence for some index $D \gg 0$.

A naive solution entails computing all terms $a_0, \ldots, a_D$, but one can do much better. It has been known since at least Fiduccia's work [2] that computing a_D can be reduced to multiplication modulo P. Explicitly, define $\mathbb{A} = \mathbb{K}[x]/P$, together with the $\mathbb{K}$-linear form $\ell : \mathbb{A} \to \mathbb{K}$ given by $\ell(x^i) = a_i$ for $i = 0, \ldots, d-1$. Then, since the residue class of x in $\mathbb{A}$ is a root of P, its powers (x^i) in $\mathbb{A}$ satisfy (1), and so do the values $(\ell(x^i))_{i\geq 0}$; this implies that $\ell(x^i) = a_i$ holds for all values of $i \geq 0$. In other words, to compute a_D it is enough to compute $R = x^D \bmod P$ as $R = r_0 + \cdots + r_{d-1}x^{d-1}$, since then $a_D = r_0 a_0 + \cdots + r_{d-1}a_{d-1}$.

Letting M denote a function such that polynomials of degree n can be multiplied in $\mathsf{M}(n)$ operations (under the assumptions of [6, Chapter 8]), computing R costs $O(\mathsf{M}(d)\log(D))$ operations in $\mathbb{K}$; we can then deduce a_D in $O(d)$ steps.

A new algorithm. In this note, we present an improvement over this previous result that is useful when P has multiple factors of high multiplicities.

First, we reduce to the case where P has the form $P = Q^m$, for squarefree Q. To that end, let $P = \prod_i Q_i^{m_i}$ be the squarefree factorization of P, with pairwise distinct $m_1, \ldots, m_s$ and Q_i squarefree of degree f_i for all i; note $d = \sum_i f_i m_i$. We can compute $x^D \bmod P$ by applying the Chinese Remainder Theorem to the quantities $x^D \bmod Q_i^{m_i}$, giving the following algorithm.

Algorithm 1 Computing one element in a linear recurrent sequence

Input:

- P: characteristic polynomial of the sequence
- $v = [a_0, \ldots, a_{d-1}]$: vector of initial conditions
- D: index

Output: D^{th} element of the sequence (a_i) as in (1)

1. compute the squarefree factorization of P as $P = \prod_i Q_i^{m_i}$
2. for $i = 1, \ldots, n$, compute $C_i = x^D \bmod Q_i^{m_i}$
3. compute $R = x^D \bmod P$ by CRT as $R = r_0 + \cdots + r_{d-1}x^{d-1}$
4. return $a_D = r_0 a_0 + \cdots + r_{d-1}a_{d-1}$.

To compute the C_i's efficiently, we will use bivariate computations. Indeed, for $i = 1, \ldots, s$ define $\mathbb{A}_i = \mathbb{K}[X]/Q_i^{m_i}$; then there exists a $\mathbb{K}$-algebra isomorphism

$$\pi_i : \mathbb{A}_i = \mathbb{K}[X]/Q_i^{m_i} \to \mathbb{K}[y, x]/\langle Q_i(y), (x-y)^{m_i}\rangle.$$

Following van der Hoeven and Lecerf [5], we will call π_i the operation of *untangling* (this is a conversion from a univariate representation to a bivariate one) and its inverse *tangling*. (To be precise, van der Hoeven and Lecerf consider a mapping to $\mathbb{K}[y, x]/\langle Q_i(y), x^{m_i}\rangle$, which is isomorphic to $\mathbb{K}[y, x]/\langle Q_i(y), (x-y)^{m_i}\rangle$ through the shift $x \mapsto x + y$).

van der Hoeven and Lecerf prove that for a given index i, untangling can be done in $O(\mathsf{M}(f_i m_i)\log(m_i))$ operations in $\mathbb{K}$; note that input and output sizes are $f_i m_i$ in this case. They also give an algorithm for tangling of cost $O(\mathsf{M}(f_i m_i)\log^2(m_i) + \mathsf{M}(f_i)\log(f_i))$; we give below a Las Vegas algorithm of cost $O(\mathsf{M}(f_i m_i)\log(f_i m_i))$ for this task. Taking the existence of such algorithms for granted, we write

$$C_i = x^D \bmod Q_i^{m_i} = \pi_i^{-1}(\delta_i), \quad \text{with} \quad \delta_i = x^D \bmod \langle Q_i(y), (x-y)^{m_i}\rangle.$$

The following allows us to compute δ_i efficiently. Define coefficients $e_0, \ldots, e_{m_i-1}$ by

$$x^D \bmod (x-1)^{m_i} = e_0 + e_1 x + \cdots + e_{m_i-1}x^{m_i-1},$$

and define $S_i(x) = (yx)^D \bmod (x-1)^{m_i} \in \mathbb{K}[y][x]$, where y is seen as an element of $\mathbb{K}[y]/Q_i(y)$. Then

$$\delta_i = x^D \bmod (x-y)^{m_i} = S_i\left(\frac{x}{y}\right),$$

where we note y is invertible in $\mathbb{K}[y]/Q_i(y)$ as $c_0 \neq 0$. Now, $S_i = y^D x^D \bmod (x-1)^{m_i} = y^D(e_0 + e_1 x + \cdots + e_{m_i-1}x^{m_i-1})$, so that

$$\delta_i = y^D e_0 + y^{D-1}e_1 x + \cdots + y^{D-(m_i-1)}e_{m_i-1}x^{m_i-1}.$$

In this algorithm, we first need to compute coefficients $e_0, \ldots, e_{m_i-1}$; assuming that $2, \ldots, m_i - 1$ are units in $\mathbb{K}$, they can be obtained in $O(\log(D) + \mathsf{M}(m_i))$ operations in $\mathbb{K}$. The powers of y we need are computed modulo Q_i, in time $O(\mathsf{M}(f_i)\log(D) + m_i\mathsf{M}(f_i))$. Altogether, we obtain δ_i using

$O(\mathsf{M}(f_i)\log(D) + \mathsf{M}(f_i m_i)))$ operations in $\mathbb{K}$. As said above, we can deduce C_i from δ_i in Las Vegas time $O(\mathsf{M}(f_i m_i)\log(f_i m_i))$. Taking all i's into account and using the super-linearity of M, the total time to compute $C_1, \ldots, C_s$ is thus

$$O(\mathsf{M}(\bar{d})\log(D) + \mathsf{M}(d)\log(d)),$$

where $\bar{d} = \sum_i f_i \leq d$ is the degree of the squarefree part of P.

Computing the squarefree factorization of P and Chinese remaindering both cost $O(\mathsf{M}(d)\log d)$ [6, Corollary 10.23], so the overall runtime is $O(\mathsf{M}(\bar{d})\log(D) + \mathsf{M}(d)\log(d))$. This is to be compared with the cost $O(\mathsf{M}(d)\log(D))$ of Fiduccia's algorithm.

Tangling and untangling. We conclude by sketching our new algorithm for tangling. van der Hoeven and Lecerf reduce the tangling operation to untangling by means of a divide-and-conquer process; we propose a direct reduction that uses transposition, inspired by an algorithm from [4] that applies in univariate situations.

In what follows we use the same notation as in the previous paragraphs, but we drop the subscript i for clarity. In particular, we write $f = \deg(Q)$ and $n = fm$ for the degree of Q^m, that is, the input and output size. Given δ in $\mathbb{K}[y,x]/\langle Q(y), (x-y)^m\rangle$, we want to find $C = c_0 + \cdots + c_{n-1}x^{n-1}$ such that $\pi(C) = \delta$; this simply means that

$$C \bmod \langle Q(y), (x-y)^m\rangle = \delta.$$

Choose a random linear form $\lambda : \mathbb{K}[y,x]/\langle Q(y), (x-y)^m\rangle \to \mathbb{K}$. For $j \geq 0$, multiply the former equality by x^j and apply λ; this gives

$$c_0\lambda(x^j) + \cdots + c_{n-1}\lambda(x^{j+n-1}) = \lambda(x^j\delta).$$

Taking $j = 0, \ldots, n-1$, we can collect these equalities in a linear system $HA = L$, with

$$H = \begin{bmatrix} \lambda(1) & \ldots & \lambda(x^{n-1}) \\ \vdots & \ddots & \vdots \\ \lambda(x^{n-1}) & \ldots & \lambda(x^{2n-2}) \end{bmatrix} \qquad A = \begin{bmatrix} c_0 \\ \vdots \\ c_{n-1} \end{bmatrix} \qquad L = \begin{bmatrix} \lambda(\delta) \\ \vdots \\ \lambda(x^{n-1}\delta) \end{bmatrix}.$$

Once H and L are known, we can recover coefficients A in $O(\mathsf{M}(n)\log(n))$ operations in $\mathbb{K}$, since the linear system is Hankel (for a generic choice of λ, matrix H has full rank n). Hence, the main question is the efficient computation of the entries of matrices H and L. Both are instances of the same problem: given a $\mathbb{K}$-linear form λ over $\mathbb{K}[y,x]/\langle Q(y), (x-y)^m\rangle$, and β in $\mathbb{K}[y,x]/\langle Q(y), (x-y)^m\rangle$, compute the values $\lambda(\beta), \ldots, \lambda(x^{n-1}\beta)$.

As already recognized by Shoup in the univariate case, this question is the transpose of the untangling map π^{-1}. As a result, using the so-called *transposition principle* [4, 3, 1], we can deduce an algorithm of cost $O(\mathsf{M}(n)\log(n)) = O(\mathsf{M}(fm)\log(fm))$ for tangling, by transposition of van der Hoeven and Lecerf's untangling algorithm.

References

[1] A. Bostan, G. Lecerf, and É. Schost. Tellegen's principle into practice. In *ISSAC'03*, pages 37–44. ACM, 2003.

[2] C. M. Fiduccia. An efficient formula for linear recurrences. *SIAM Journal on Computing*, 14(1):106–112, 1985.

[3] E. Kaltofen. Challenges of symbolic computation: my favorite open problems. *J. Symb. Comp.*, 29(6):891–919, 2000.

[4] V. Shoup. Fast construction of irreducible polynomials over finite fields. *Journal of Symbolic Computation*, 17(5):371–391, 1994.

[5] J. van der Hoeven and G. Lecerf. Composition modulo powers of polynomials. In *ISSAC '17*, pages 445–452. ACM, 2017.

[6] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, Cambridge, third edition, 2013.