Interpolation of Shifted-Lacunary Polynomials [Extended Abstract]

Mark Giesbrecht and Daniel S. Roche

Abstract. Given a "black box" function to evaluate an unknown rational polynomial $f \in \mathbb{Q}[x]$ at points modulo a prime p, we exhibit algorithms to compute the representation of the polynomial in the sparsest shifted power basis. That is, we determine the sparsity $t \in \mathbb{Z}_{>0}$, the shift $\alpha \in \mathbb{Q}$, the exponents $0 \le e_1 < e_2 < \cdots < e_t$, and the coefficients $c_1, \ldots, c_t \in \mathbb{Q} \setminus \{0\}$ such that

$$f(x) = c_1(x-\alpha)^{e_1} + c_2(x-\alpha)^{e_2} + \cdots + c_t(x-\alpha)^{e_t}.$$

The computed sparsity t is *absolutely* minimal over any shifted power basis. The novelty of our algorithm is that the complexity is polynomial in the (sparse) representation size and in particular is logarithmic in $\deg f$. Our method combines previous celebrated results on sparse interpolation and computing sparsest shifts, and provides a way to handle polynomials with extremely high degree which are, in some sense, sparse in information. We give both an unconditional deterministic algorithm which is polynomial-time but has a rather high complexity, and a more practical probabilistic algorithm which relies on some unknown constants.

Mathematics Subject Classification (2000). Primary 68W30; Secondary 12Y05.

Keywords. Sparse interpolation, Sparsest shift, Lacunary polynomials.

1. Introduction

Interpolating an unknown polynomial from a set of evaluations is a problem which has interested mathematicians for hundreds of years, and which is now implemented as a primitive operation in most computer algebra systems. To illustrate some different kinds of interpolation problems, consider the following three representations for a polynomial

Appears in Mathematical Aspects of Computer and Information Sciences (MACIS'07), Paris, France, December 5–7, 2007.

$f(x)$ of degree n:

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \qquad (1.1)$$
$$f(x) = b_1 x^{d_1} + b_2 x^{d_2} + \cdots + b_s x^{d_s} \qquad (1.2)$$
$$f(x) = c_1 (x-\alpha)^{e_1} + c_2 (x-\alpha)^{e_2} + \cdots + c_t (x-\alpha)^{e_t} \qquad (1.3)$$

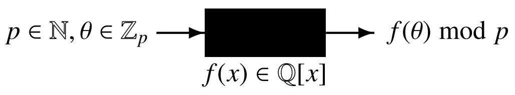
In (1.1) we see the dense representation of the polynomial, where all coefficients are represented, even if they are zero. Newton and Waring discovered methods to interpolate $f(x)$ in time proportional to the size of this representation in the 18th century. The sparse or lacunary representation is shown in (1.2), wherein only the terms with non-zero coefficients are shown. Less than 20 years ago, Ben-Or and Tiwari [3] discovered a method to interpolate in time polynomial in the size of this representation (see [12]). This method has recently been pointed out to be essentially equivalent to the method of de Prony from 1795 [6].

When α is chosen so that t is absolutely minimal in (1.3), we call this the *sparsest shift* of $f(x)$. We present new algorithms to interpolate $f(x) \in \mathbb{Q}[x]$, given a black box for evaluation, in time proportional to the size of the shifted-lacunary representation corresponding to (1.3). It is easy to see that t could be exponentially smaller than both n and s, for example when $f(x) = (x+1)^n$, demonstrating that our algorithms are providing a significant improvement in complexity over those previously known.

The main applications of all these methods for polynomial interpolation are signal processing and reducing intermediate expression swell. Dense and sparse interpolation have been applied successfully to both these ends, and our new algorithms effectively extend the class of polynomials for which such applications can be made.

The most significant challenge here is computing the sparsest shift index $\alpha \in \mathbb{Q}$. Computing this value from a set of evaluation points was stated as an open problem in [4]. An algorithm for a generalization of our problem in the dense representation was given in [9], though its cost is exponential in the size of the output; they admit that the dependency on the degree of the polynomial is probably not optimal. Our algorithm achieves deterministic polynomial-time complexity for polynomials over the rational numbers. We are always careful to count the *bit complexity* — the number of fixed-precision machine operations — and hence account for any coefficient growth in the solution or intermediate expressions.

The black box model we use is slightly modified from the traditional one:



Given a prime p and an element θ in $\mathbb{Z}_p$, the black box computes the value of the unknown polynomial evaluated at θ over the field $\mathbb{Z}_p$. Note that in some sense this restriction is necessary for rational (or integer) polynomials, since the value of a polynomial of degree n at any point other than $0, \pm 1$ will typically have n bits or more. Thus, any algorithm whose complexity is proportional to $\log n$ cannot evaluate such a polynomial over $\mathbb{Q}$ or $\mathbb{Z}$.

To be precise, define for a rational polynomial $f(x)$ as in (1.3):

$$\mathrm{size}(f(x)) = \mathrm{size}(\alpha) + \sum_{i=1}^{t} (\mathrm{size}(c_i) + \mathrm{size}(e_i)) .^{*} \tag{1.4}$$

Our algorithms will have polynomial complexity in the smallest possible size$(f(x))$.

2. Computing the Sparsest Shift

The input to the algorithms will be a modular black box for evaluating a rational polynomial, as described above, and a bound $B \in \mathbb{N}$ on size$(f(x))$.

Here, and for the remainder of this discussion, for $f(x) \in \mathbb{Q}[x]$ as in (1.3), define

$$f_p(x) = (c_1 \bmod p)(x - \alpha_p)^{e_1 \bmod (p-1)} + \ldots + (c_t \bmod p)(x - \alpha_p)^{e_t \bmod (p-1)}, \tag{2.1}$$

where $\alpha_p = \alpha \bmod p$.

Using Fermat's Little Theorem, we can see that $f(\theta) \bmod p = f_p(\theta)$ for all $\theta \in \mathbb{Z}_p$ with $\theta \neq \alpha_p$. If we are "lucky", then $f(\alpha_p) \bmod p = f_p(\alpha_p)$ also, so that the unique polynomial in $\mathbb{Z}_p[x]$ interpolating any $(p-1)$ of the points $(i, f(i) \bmod p)$, for $i \in \mathbb{Z}_p$, is just $f_p(x)$.

The basic outline of our approach to computing the sparsest shift index α is as follows:

1. Choose a prime p with $p \in O(B^{O(1)})$
2. Use dense (Lagrange) interpolation to compute $f_p(x)$ or determine that $f(\alpha_p) \bmod p \neq f_p(\alpha_p)$.
3. Use a dense method to compute the sparsest shift β_p of $f_p(x)$ and determine whether $\beta_p = \alpha_p$.
4. Repeat Steps 1 to 3 enough times to recover α

As an example, suppose we are given a modular black box for the following polynomial:

$$\begin{aligned} f(x) = x^{15} &- 45x^{14} + 945x^{13} - 12285x^{12} + 110565x^{11} - 729729x^{10} + 3648645x^{9} \\ &- 14073345x^{8} + 42220035x^{7} - 98513415x^{6} + 177324145x^{5} - 241805625x^{4} \\ &+ 241805475x^{3} - 167403375x^{2} + 71743725x - 14348421. \end{aligned}$$

In fact, it is easy to see that $f(x) = (x-3)^{15} - 2(x-3)^5$. Now suppose the algorithm chooses $p = 7$ in Step 1. We use the black box to find $f(1), f(2), \ldots, f(6)$ in $\mathbb{Z}_7$, and give this input to the Lagrange interpolation algorithm to compute

$$f_7(x) = 5x^5 + 2x^4 + 3x^3 + 6x^2 + x + 4.$$

We confirm that $f(\alpha_7) \bmod 7 = f_7(\alpha_7)$ and therefore that we have computed the correct $f_7(x)$ (as in (2.1)) by checking that $f(0) \bmod 7 = f_7(0)$.

Next, in Step 3, we use a dense sparsest shift computation method to find that $f_7(x) = 5(x+4)^5 + (x+4)^3$. Thus, the sparsest shift index must be congruent to 4 mod 7. In this

[*] For $q = \frac{a}{b} \in \mathbb{Q}$ and $a \in \mathbb{Z}$, $b \in \mathbb{N}$, $\gcd(a, b) = 1$, size(q) is $\lceil \log_2(|a| + 1) \rceil + \lceil \log_2(b + 1) \rceil + 1$, the number of bits needed to represent q.

case, we know that $\alpha = -3$, which of course is congruent to 4 modulo 7. We have one modular image to use in recovering α at Step 4.

The following theorem gives sufficient conditions for success in Steps 2 and 3.

Theorem 2.1. *$f_p(x)$ is computed correctly and $\beta_p = \alpha_p$ whenever the following hold:*

- $\forall i \in \{1, 2, \ldots, t-1\}$ s.t. $e_i \neq 0,\ e_i \not\equiv 0 \mod (p-1)$.
- $c_t \not\equiv 0 \mod p$;
- $\forall i \in \{1, 2, \ldots, t-1\},\ e_t \not\equiv e_i \mod (p-1)$;
- $\forall i \in \{0, 1, \ldots, 2t-1\},\ e_t \not\equiv i \mod (p-1)$.

Proof. The first condition tells us that the constant coefficient of $f_p(x)$ will be the same as the constant coefficient of $f(x)$ (either it is c_1 or 0, depending on whether $e_1 = 0$). Therefore $f(\alpha_p) \bmod p = f_p(\alpha_p)$, so $f_p(x)$ will be computed correctly.

The second condition guarantees that the last term of $f_p(x)$ as in (2.1) does not vanish. We also know that there is no other term with the same degree from the third condition. Finally, the fourth condition tells us that the degree of the last term will be at least $2t - 1$.

This means that the degree of $f_p(x)$ is at least $2t - 1$. Lakshman and Saunders [13] prove that there can be at most one shift with sparsity at most t. Since α_p must be at worst a t-sparse shift, it is the unique sparsest shift, and therefore $\beta_p = \alpha_p$. □

Now define

$$C = \max\{e_1, 1\} \prod_{i=2}^{t-1} e_i \prod_{i=1}^{t-1} (e_t - e_i) \prod_{i=0}^{2t-2} (e_t - i).$$

Step 3 succeeds whenever $p \nmid c_t$ and $(p-1) \nmid C$. Of course, c_t and C are unknown, but we do have $c_t < 2^B$ and $C < 2^{4B^2}$.

Additionally, we know the numerator and denominator of α are both bounded by 2^B, so we need a total of $2B + 1$ "good" primes to recover both of them and the sign.

Also notice that if $\deg f(x) < 2t - 1$, then $\prod_{i<2t-1}(e_t - i) = 0$, so Theorem 2.1 will never hold. But this case is easy to discard, as here $\deg(f) \leq 2B$, so we can just use dense interpolation modulo a single "big" prime and recover the sparsest shift directly.

2.1. A Deterministic Algorithm For The Sparsest Shift

For the deterministic algorithm, we choose primes p from a pre-constructed set $\mathcal{P}$. This set is computed by first finding the first k primes $q_1, q_2, \ldots, q_k$, where $k \in O(B^2 \log B)$. For each q_i, let p_i be the least prime such that $q_i \mid (p_i - 1)$, and set $\mathcal{P} = \bigcup_{1 \leq i \leq k} \{p_i\}$.

We need to show that at least $2B + 1$ of the primes p_i satisfy the conditions of Theorem 2.1. For this, we first show that only $O(\log k)$ of the p_i's can share any single value, and therefore the number of distinct p_i's is at least $\Omega(k/\log k)$. [†] We can see that only $O(B)$ of the primes in $\mathcal{P}$ can divide c_t, and $O(B^2)$ of the q_i's can divide C. Thus the rest satisfy the conditions of Theorem 2.1; we choose k just large enough so that there are at least $2B + 1$ such p_i's.

[†] We use the normal notation for the asymptotic behaviour function Ω, namely $f(n) \in \Omega(g(n))$ iff $g(n) \in O(f(n))$.

Step 2 involves first computing a candidate $f_p(x)$ by dense interpolation on the points $f(\theta) \bmod p$ for $\theta = 1, 2, \ldots, p-1$, then checking that $f(0) \bmod p = f_p(0)$ to confirm that $f_p(x)$ was computed correctly. Note that we can also detect Step 3 failures at this stage, namely when $\deg f_p(x) < 2B - 1$.

For Step 3, we use the algorithm `UniSparsestShifts<symbolic>` from [7], which has worst-case complexity $O(p^2 \log p \, \mathsf{M}(p^5))$. [‡]

These steps will be repeated at most k times, which again is $O(B^2)$. But since we detect failures on Step 2, Step 3 will never be executed more than $2B + 1$ times. In fact, this step dominates the complexity, which is $O(Bp^2 \log p \, \mathsf{M}(p^5))$. Now we need the upper bound on the p_i's provided by Linnik's Theorem, which tells us that, for each q_i, $p_i \in O({q_i}^L)$, for some constant L [14]. Current best results give $L \leq 5.5$ [10]. Also, the effective bounds on the prime number theorem given in [16] tell us that the kth prime q_k is $O(k \log k)$. Combining these sizes, we have that, for all $p \in \mathcal{P}$, $p \in O(B^{11} (\log B)^{11})$.

Theorem 2.2. *Computing the sparsest shift of a rational polynomial given a modular black box for evaluation can be performed in output-sensitive polynomial time.*

This all leads us to the rather nasty (yet still polynomial!) complexity bound of $O(B^{23} (\log B)^{23} \, \mathsf{M}(B^5 5 (\log B)^5 5))$, or $O(B^{78} (\log B)^{24} \log\log B)$ if we use the results from [5]. We do note, however, that the high exponent in the complexity is mostly a result of the best known bound of 5.5 for Linnik's constant L. In fact, under the Extended Riemann Hypothesis (ERH), $p_i \leq 2{q_i}^2 (\ln q_i)^2$ [2], and it has been conjectured that $L = 2$ [10].

2.2. Improvements to the Algorithm

The following significant improvements in the algorithm's complexity are possible by introducing randomization and accepting some unknown constants.

First, we choose the primes p_i via randomization "on the fly" rather than explicitly pre-computing a set $\mathcal{P}$. For this, first choose a random prime $q \in O(B^2)$. With high probability (using [16]), $q \nmid C$. Next, search for a prime $p = kq + 1$, initially with $1 < k < q$. After $O(\log q)$ failures, search with $q < k < q^2$ and repeat. Next try $q^2 < k < q^4$ and so on, until a prime is found. The bounds on the number primes with μ bits from [8] allow us to use [17] to prove the density of primes here is just under half the density of primes in general once k is larger than some unspecified constant, and so the number of primality tests necessary to find p is expected logarithmic if B is large enough.

We have $p \in O(B^4)$ and the same argument as before shows that only an expected $O(B^2 \log B)$ randomly-chosen p's are necessary to find $2B + 1$ which are "good".

Step 2 can be performed in exactly the same way, but for Step 3, we can now use a probabilistic algorithm. The fastest one, from [7], has complexity $O(p^2 \log p \, \mathsf{M}(p^2))$.

This brings the total cost down to $O(B^8 \log B \, \mathsf{M}(B^8))$, or $O(B^{16} (\log B)^9 \log\log B)$ if we use [5]. This is probably still not the optimal complexity, but it is certainly much more manageable than that of the deterministic algorithm above.

[‡]Here and for the remainder of the document, $\mathsf{M}(n)$ refers to the number of operations in $\mathbb{Z}_p$ needed to compute the product of two polynomials of degree less than n over $\mathbb{Z}_p[x]$, for any p. Current best results in [5] give $\mathsf{M}(n) \in O(n \log n \log\log n)$.

3. Interpolation

Having computed α, we know that $f(x+\alpha)$ has sparsity at most B in the standard power basis. And this polynomial can be evaluated with our black box and the value of α.

This means we can use the methods of sparse interpolation, in the style of Ben-Or and Tiwari [3], to compute the coefficients c_i and exponents e_i of $f(x)$ in the α-shifted basis. For example, we can employ the randomized algorithm of [11] or deterministic lifting algorithm of [1], which require black box evaluation modulo a prime power p^k.

One of the bottlenecks in Ben-Or/Tiwari over $\mathbb{Z}_p$ (that is, maintaining prime modulus) is doing discrete logarithm in $\mathbb{Z}_p^*$. Using $p = q^k s + 1$, for some small prime q and small k and s, we can restrict our discrete logarithm computations to the subgroup of size q^k inside $\mathbb{Z}_p^*$. This can be accomplished in polynomial-time using the Pohlig-Hellman Algorithm [15]. A randomized algorithm for constructing such a p is straightforward using [17] as before, or assuming the ERH, though an unconditional deterministic algorithm is still elusive.

4. Conclusions and Future Work

Our work here provides the first step towards a general solution to the problem of interpolating an unknown polynomial given by its values into the sparsest shifted power basis in time polynomially-bounded by the size of the output. The main tool we have used is mapping down modulo small primes where the sparse shift is also mapped nicely. This technique could be useful for other problems involving lacunary polynomials as well, although it is not clear how it would apply in finite domains where there is no notion of "size".

Another obvious avenue for investigation is computing sparsest shift representations for multivariate polynomials. Of course, all of the sparse interpolation algorithms work well in the multivariate case, and in fact this is their primary domain of application, but computing multivariate shifts seems to be more difficult [7]. Nevertheless, the ideas we present here may shed some new light on this problem.

References

[1] M. Avendano, T. Krick, and A. Pacetti. Newton-hensel interpolation lifting. *Foundations of Computational Mathematics*, 6(1):82–120, 2006.

[2] E. Bach and J. Sorenson. Explicit bounds for primes in residue classes. *Math. Comput.*, 65(216):1717–1735, 1996.

[3] Michael Ben-Or and Prasoon Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation. In *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 301–309, New York, NY, USA, 1988. ACM Press.

[4] Allan Borodin and Prasoon Tiwari. On the decidability of sparse univariate polynomial interpolation. *Comput. Complexity*, 1(1):67–90, 1991.

[5] D. Cantor and E. Kaltofen. Fast multiplication of polynomials over arbitrary algebras. *Acta Informatica*, 28:693–701, 1991.

[6] M. Giesbrecht, G. Labahn, and W s. Lee. Symbolic-numeric sparse interpolation of multivariate polynomials. In *Proc. ACM International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 116–123, 2006.

[7] Mark Giesbrecht, Erich Kaltofen, and Wen-shin Lee. Algorithms for computing sparsest shifts of polynomials in power, Chebyshev and Pochhammer bases. *J. Symbolic Comput.*, 36(3-4):401–424, 2003. International Symposium on Symbolic and Algebraic Computation (ISSAC'2002) (Lille).

[8] Mark Giesbrecht and Arne Storjohann. Computing rational forms of integer matrices. *J. Symbolic Comput.*, 34(3):157–172, 2002.

[9] Dima Grigoriev and Marek Karpinski. A zero-test and an interpolation algorithm for the shifted sparse polynomials. In *Applied algebra, algebraic algorithms and error-correcting codes (San Juan, PR, 1993)*, volume 673 of *Lecture Notes in Comput. Sci.*, pages 162–169. Springer, Berlin, 1993.

[10] D. R. Heath-Brown. Zero-free regions for Dirichlet L-functions, and the least prime in an arithmetic progression. *Proc. London Math. Soc. (3)*, 64(2):265–338, 1992.

[11] E. Kaltofen, Y. N. Lakshman, and J.-M. Wiley. Modular rational sparse multivariate polynomial interpolation. In *ISSAC '90: Proceedings of the international symposium on Symbolic and algebraic computation*, pages 135–139, New York, NY, USA, 1990. ACM.

[12] Erich Kaltofen and Wen-shin Lee. Early termination in sparse interpolation algorithms. *J. Symbolic Comput.*, 36(3-4):365–400, 2003. International Symposium on Symbolic and Algebraic Computation (ISSAC'2002) (Lille).

[13] Y. N. Lakshman and B. David Saunders. Sparse shifts for univariate polynomials. *Appl. Algebra Engrg. Comm. Comput.*, 7(5):351–364, 1996.

[14] U. V. Linnik. On the least prime in an arithmetic progression. I. The basic theorem. *Rec. Math. [Mat. Sbornik] N.S.*, 15(57):139–178, 1944.

[15] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. *IEEE Trans. Information Theory*, 24, 1978.

[16] J. Barkley Rosser and Lowell Schoenfeld. Approximate formulas for some functions of prime numbers. *Illinois J. Math.*, 6:64–94, 1962.

[17] Bruno Rousselet. Inégalités de type Brun-Titchmarsh en moyenne. In *Groupe de travail en théorie analytique et élémentaire des nombres, 1986–1987*, volume 88 of *Publ. Math. Orsay*, pages 91–123. Univ. Paris XI, Orsay, 1988.

Mark Giesbrecht
School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada
e-mail: `mwg@uwaterloo.ca`

Daniel S. Roche
School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada
e-mail: `droche@cs.uwaterloo.ca`