Fast decoding of univariate multiplicity codes

Based on joint work with Rohan Goyal, Prahladh Harsha and Ashutosh Shankar

Fast decoding of univariate multiplicity codes

Fast Noisy Hermite Interpolation

Based on joint work with Rohan Goyal, Prahladh Harsha and Ashutosh Shankar

A Basic Fact About Polynomials

A Basic Fact About Polynomials

A non-zero univariate polynomial of degree d over a field has at most d zeroes.

A Basic Fact About Polynomials

A non-zero univariate polynomial of degree d over a field has at most d zeroes.

In other words…

A Basic Fact About Polynomials

A non-zero univariate polynomial of degree d over a field has at most d zeroes.

In other words…

$$F - field, \quad P, Q \in F[x], \quad deg \leq d, \quad P \neq Q$$

A Basic Fact About Polynomials

A non-zero univariate polynomial of degree d over a field has at most d zeroes.

In other words…

$$F - field, \;\; P, Q \in F[x], \;\; deg \leq d, \;\; P \neq Q$$

Then, $\left| \{a \in F : P(a) = Q(a)\} \right| \leq d.$

A Consequence

A Consequence

$$S \subseteq F, \quad \left|S\right| = n, \quad R : S \to F$$

A Consequence

$$S \subseteq F, \quad |S| = n, \quad R: S \to F$$

Then, the number of polynomials $P(x) \in F[x]$ of degree at most d, such that $|\{a \in S: P(a) = R(a)\}| > \frac{n+d}{2}$ is at most 1.

A Consequence

$$S \subseteq F, \quad |S| = n, \quad R: S \to F$$

Then, the number of polynomials $P(x) \in F[x]$ of degree at most d, such that $|\{a \in S: P(a) = R(a)\}| > \frac{n+d}{2}$ is at most 1.

Question 1: Given the set S and the function R, find P efficiently.

A Consequence

$$S \subseteq F, \quad |S| = n, \quad R: S \to F$$

Then, the number of polynomials $P(x) \in F[x]$ of degree at most d, such that $|\{a \in S: P(a) = R(a)\}| > \frac{n+d}{2}$ is at most 1.

Question 1: Given the set S and the function R, find P efficiently.

R and P agree everywhere: this is univariate polynomial interpolation

A Consequence

$$S \subseteq F, \quad |S| = n, \quad R: S \to F$$

Then, the number of polynomials $P(x) \in F[x]$ of degree at most d, such that $|\{a \in S: P(a) = R(a)\}| > \frac{n+d}{2}$ is at most 1.

Question 1: Given the set S and the function R, find P efficiently.

R and P agree everywhere: this is univariate polynomial interpolation

Here – polynomial interpolation in the presence of errors/noise.

Univariate Polynomial Interpolation with Noise

Univariate Polynomial Interpolation with Noise

Many known algorithms for this problem, since the 60’s.

Univariate Polynomial Interpolation with Noise

Many known algorithms for this problem, since the 60's.

Peterson's algorithm, Berlekamp-Massey, Berlekamp-Welch

Univariate Polynomial Interpolation with Noise

Many known algorithms for this problem, since the 60's.

Peterson's algorithm, Berlekamp-Massey, Berlekamp-Welch

Elementary, but neat and beautiful, and widely useful.

Univariate Polynomial Interpolation with Noise

Many known algorithms for this problem, since the 60's.

Peterson's algorithm, Berlekamp-Massey, Berlekamp-Welch

Elementary, but neat and beautiful, and widely useful.

This talk : about generalizations of this problem

Univariate Polynomial Interpolation with Noise

Many known algorithms for this problem, since the 60's.

Peterson's algorithm, Berlekamp-Massey, Berlekamp-Welch

Elementary, but neat and beautiful, and widely useful.

This talk : about generalizations of this problem

- higher multiplicities

Univariate Polynomial Interpolation with Noise

Many known algorithms for this problem, since the 60's.

Peterson's algorithm, Berlekamp-Massey, Berlekamp-Welch

Elementary, but neat and beautiful, and widely useful.

This talk : about generalizations of this problem

- higher multiplicities
- smaller number of agreements

The Case of High Multiplicity

The Case of High Multiplicity

$P \in F[x]$ vanishes with multiplicity at least s at $a \in F$ if all the derivatives of P of order at most (s-1) vanish at a.

$$P(a) = P^{(1)}(a) = \cdots = P^{(s-1)}(a) = 0$$

The Case of High Multiplicity

$P \in F[x]$ vanishes with multiplicity at least s at $a \in F$ if all the derivatives of P of order at most (s-1) vanish at a.

$$P(a) = P^{(1)}(a) = \cdots = P^{(s-1)}(a) = 0$$

Fact:

A non-zero univariate polynomial of degree d over a field has at most d/s zeroes of multiplicity s.

A Consequence

A Consequence

$$S \subseteq F, |S| = n, R : S \to F^S$$

A Consequence

$$S \subseteq F, |S| = n, R : S \to F^s$$

Then, the number of polynomials $P(x) \in F[x]$ of degree at most d, such that

$|a \in S : P, R$ agree on a with mult s$| > \frac{n + d/s}{2}$ is at most 1.

A Consequence

$$S \subseteq F, |S| = n, R : S \to F^s$$

Then, the number of polynomials $P(x) \in F[x]$ of degree at most d, such that

$|a \in S : P, R$ agree on a with mult s$| > \frac{n + d/s}{2}$ is at most 1.

Agreement with mult at least s: for i in {0, 1, …, s-1}, $R_i(a) = P^{(i)}(a)$

A Consequence

$$S \subseteq F, |S| = n, R : S \to F^s$$

Then, the number of polynomials $P(x) \in F[x]$ of degree at most d, such that

$|a \in S : P, R$ agree on a with mult s$| > \frac{n + d/s}{2}$ is at most 1.

Agreement with mult at least s: for i in {0, 1, …, s-1}, $R_i(a) = P^{(i)}(a)$

Noisy Hermite interpolation: Given the set S and a function R, find P efficiently.

Noisy interpolation: tolerating more errors

Noisy interpolation: tolerating more errors

Noisy polynomial interpolation: errors more than (n-d)/2

Noisy interpolation: tolerating more errors

Noisy polynomial interpolation: errors more than (n-d)/2

Noisy interpolation: tolerating more errors

Noisy polynomial interpolation: errors more than (n-d)/2

Noisy Hermite interpolation: errors more than (n-d/s)/2

Noisy interpolation: tolerating more errors

Noisy polynomial interpolation: errors more than (n-d)/2

Noisy Hermite interpolation: errors more than (n-d/s)/2

The ‘close enough’ polynomial is no longer unique…but how many can there be ?

Noisy interpolation: tolerating more errors

Noisy polynomial interpolation: errors more than (n-d)/2

Noisy Hermite interpolation: errors more than (n-d/s)/2

The ‘close enough’ polynomial is no longer unique…but how many can there be ?

Theorem [Johnson’ 62]

Noisy interpolation: tolerating more errors

Noisy polynomial interpolation: errors more than (n-d)/2

Noisy Hermite interpolation: errors more than (n-d/s)/2

The ‘close enough’ polynomial is no longer unique…but how many can there be ?

Theorem [Johnson’ 62]

Number of such polynomials is a polynomially bounded for agreement more than $\sqrt{nd}$ for

Noisy interpolation: tolerating more errors

Noisy polynomial interpolation: errors more than (n-d)/2

Noisy Hermite interpolation: errors more than (n-d/s)/2

The ‘close enough’ polynomial is no longer unique…but how many can there be ?

Theorem [Johnson’ 62]

Number of such polynomials is a polynomially bounded for agreement more than $\sqrt{nd}$ for noise polynomial interpolation, and agreement more than $\sqrt{nd/s}$ for noisy Hermite interpolation.

Noisy interpolation: tolerating more errors

Noisy polynomial interpolation: errors more than (n-d)/2

Noisy Hermite interpolation: errors more than (n-d/s)/2

The ‘close enough’ polynomial is no longer unique…but how many can there be ?

Theorem [Johnson’ 62]

Number of such polynomials is a polynomially bounded for agreement more than $\sqrt{nd}$ for noise polynomial interpolation, and agreement more than $\sqrt{nd/s}$ for noisy Hermite interpolation.

Q. Can these polynomials be found efficiently ?

Noisy poly interpolation up to Johnson bound

Noisy poly interpolation up to Johnson bound

Theorem [Sudan 95, Guruswami-Sudan 98]

Noisy poly interpolation up to Johnson bound

Theorem [Sudan 95, Guruswami-Sudan 98]

Polynomial time algorithms for noisy polynomial interpolation up to Johnson bound.

Noisy poly interpolation up to Johnson bound

Theorem [Sudan 95, Guruswami-Sudan 98]

Polynomial time algorithms for noisy polynomial interpolation up to Johnson bound.

Extremely influential results:

Noisy poly interpolation up to Johnson bound

Theorem [Sudan 95, Guruswami-Sudan 98]

Polynomial time algorithms for noisy polynomial interpolation up to Johnson bound.

Extremely influential results: among the greatest hits in computer science

Noisy poly interpolation up to Johnson bound

Theorem [Sudan 95, Guruswami-Sudan 98]

Polynomial time algorithms for noisy polynomial interpolation up to Johnson bound.

Extremely influential results: among the greatest hits in computer science, elementary and simple but very clever,

Noisy poly interpolation up to Johnson bound

Theorem [Sudan 95, Guruswami-Sudan 98]

Polynomial time algorithms for noisy polynomial interpolation up to Johnson bound.

Extremely influential results: among the greatest hits in computer science, elementary and simple but very clever, re-introduces new ideas (the polynomial method),

Noisy poly interpolation up to Johnson bound

Theorem [Sudan 95, Guruswami-Sudan 98]

Polynomial time algorithms for noisy polynomial interpolation up to Johnson bound.

Extremely influential results: among the greatest hits in computer science, elementary and simple but very clever, re-introduces new ideas (the polynomial method), consequences in

Noisy poly interpolation up to Johnson bound

Theorem [Sudan 95, Guruswami-Sudan 98]

Polynomial time algorithms for noisy polynomial interpolation up to Johnson bound.

Extremely influential results: among the greatest hits in computer science, elementary and simple but very clever, re-introduces new ideas (the polynomial method), consequences in pseudorandomness,

Noisy poly interpolation up to Johnson bound

Theorem [Sudan 95, Guruswami-Sudan 98]

Polynomial time algorithms for noisy polynomial interpolation up to Johnson bound.

Extremely influential results: among the greatest hits in computer science, elementary and simple but very clever, re-introduces new ideas (the polynomial method), consequences in pseudorandomness, complexity theory,

Noisy poly interpolation up to Johnson bound

Theorem [Sudan 95, Guruswami-Sudan 98]

Polynomial time algorithms for noisy polynomial interpolation up to Johnson bound.

Extremely influential results: among the greatest hits in computer science, elementary and simple but very clever, re-introduces new ideas (the polynomial method), consequences in pseudorandomness, complexity theory, cryptography

Noisy poly interpolation up to Johnson bound

Theorem [Sudan 95, Guruswami-Sudan 98]

Polynomial time algorithms for noisy polynomial interpolation up to Johnson bound.

Extremely influential results: among the greatest hits in computer science, elementary and simple but very clever, re-introduces new ideas (the polynomial method), consequences in pseudorandomness, complexity theory, cryptography and discrete math.

Noisy poly interpolation up to Johnson bound

Theorem [Sudan 95, Guruswami-Sudan 98]

Polynomial time algorithms for noisy polynomial interpolation up to Johnson bound.

Extremely influential results: among the greatest hits in computer science, elementary and simple but very clever, re-introduces new ideas (the polynomial method), consequences in pseudorandomness, complexity theory, cryptography and discrete math.

Subsequent attempts at improving the running time further.

Noisy poly interpolation up to Johnson bound

Theorem [Sudan 95, Guruswami-Sudan 98]

Polynomial time algorithms for noisy polynomial interpolation up to Johnson bound.

Extremely influential results: among the greatest hits in computer science, elementary and simple but very clever, re-introduces new ideas (the polynomial method), consequences in pseudorandomness, complexity theory, cryptography and discrete math.

Subsequent attempts at improving the running time further.

Theorem [Alekhnovich 2002]

A nearly linear time algorithm for this problem, almost up to the Johnson bound.

Noisy Hermite interpolation up to Johnson bound

Noisy Hermite interpolation up to Johnson bound

Theorem [Nielsen 2001]

Noisy Hermite interpolation up to Johnson bound

Theorem [Nielsen 2001]

A polynomial time algorithm for noisy Hermite interpolation up to Johnson bound.

Noisy Hermite interpolation up to Johnson bound

Theorem [Nielsen 2001]

A polynomial time algorithm for noisy Hermite interpolation up to Johnson bound.

Builds on the ideas in Guruswami-Sudan and much more…..

Noisy Hermite interpolation up to Johnson bound

Theorem [Nielsen 2001]

A polynomial time algorithm for noisy Hermite interpolation up to Johnson bound.

Builds on the ideas in Guruswami-Sudan and much more…..

A nearly linear time algorithm for this remained elusive though.

Noisy Hermite interpolation up to Johnson bound

Theorem [Nielsen 2001]

A polynomial time algorithm for noisy Hermite interpolation up to Johnson bound.

Builds on the ideas in Guruswami-Sudan and much more…..

A nearly linear time algorithm for this remained elusive though.

Kopparty 2014: Is there a nearly linear time analog of Nielsen's algorithm ?

Landscape beyond Johnson bound

Landscape beyond Johnson bound

Noisy polynomial interpolation:

Landscape beyond Johnson bound

Noisy polynomial interpolation: overall unclear…depends on the set S of evaluation points, and no algorithms are known.

Landscape beyond Johnson bound

Noisy polynomial interpolation: overall unclear…depends on the set S of evaluation points, and no algorithms are known.

A very active area of research…many many open problems.

Landscape beyond Johnson bound

Landscape beyond Johnson bound

Theorem [Kopparty 13, Guruswami-Wang 14]

Landscape beyond Johnson bound

Theorem [Kopparty 13, Guruswami-Wang 14]

For all $\epsilon > 0$, and all $s > 1/\epsilon^2$, there is an efficient algorithm for noisy Hermite interpolation for agreement greater than $(1 + \epsilon)d/s.$

Landscape beyond Johnson bound

Theorem [Kopparty 13, Guruswami-Wang 14]

For all $\epsilon > 0$, and all $s > 1/\epsilon^2$, there is an efficient algorithm for noisy Hermite interpolation for agreement greater than $(1 + \epsilon)d/s.$

Apriori, not even clear that the number of such polynomials is poly bounded.

Landscape beyond Johnson bound

Theorem [Kopparty 13, Guruswami-Wang 14]

For all $\epsilon > 0$, and all $s > 1/\epsilon^2$, there is an efficient algorithm for noisy Hermite interpolation for agreement greater than $(1 + \epsilon)d/s.$

Apriori, not even clear that the number of such polynomials is poly bounded.

Note that we need agreement at least d/s to determine a deg d polynomial.

Landscape beyond Johnson bound

Theorem [Kopparty 13, Guruswami-Wang 14]

For all $\epsilon > 0$, and all $s > 1/\epsilon^2$, there is an efficient algorithm for noisy Hermite interpolation for agreement greater than $(1+\epsilon)d/s.$

Apriori, not even clear that the number of such polynomials is poly bounded.

Note that we need agreement at least d/s to determine a deg d polynomial.

Kopparty-Ron Zewi-Saraf-Wootters 2018: the number of polynomials output by the above algorithms is a constant.

Landscape beyond Johnson bound

Theorem [Kopparty 13, Guruswami-Wang 14]

For all $\epsilon > 0$, and all $s > 1/\epsilon^2$, there is an efficient algorithm for noisy Hermite interpolation for agreement greater than $(1+\epsilon)d/s.$

Apriori, not even clear that the number of such polynomials is poly bounded.

Note that we need agreement at least d/s to determine a deg d polynomial.

Kopparty-Ron Zewi-Saraf-Wootters 2018: the number of polynomials output by the above algorithms is a constant.

Q.Can all of these be done in nearly linear time ?

Our results

Our results

Theorem [Goyal, Harsha, K., Shankar 2023]

Our results

Theorem [Goyal, Harsha, K., Shankar 2023]

For all $\epsilon > 0$, and all $s > 1/\epsilon^2$, there is a **nearly linear time** algorithm for noisy Hermite interpolation for agreement greater than $(1 + \epsilon)d/s.$

Our results

Theorem [Goyal, Harsha, K., Shankar 2023]

For all $\epsilon > 0$, and all $s > 1/\epsilon^2$, there is a **nearly linear time** algorithm for noisy Hermite interpolation for agreement greater than $(1 + \epsilon)d/s.$

Some caveats :

Our results

Theorem [Goyal, Harsha, K., Shankar 2023]

For all $\epsilon > 0$, and all $s > 1/\epsilon^2$, there is a **nearly linear time** algorithm for noisy Hermite interpolation for agreement greater than $(1+\epsilon)d/s.$

Some caveats : d/sn is a constant , characteristic of the underlying field is zero or greater than d.

Our results

Theorem [Goyal, Harsha, K., Shankar 2023]

For all $\epsilon > 0$, and all $s > 1/\epsilon^2$, there is a **nearly linear time** algorithm for noisy Hermite interpolation for agreement greater than $(1+\epsilon)d/s.$

Some caveats : d/sn is a constant , characteristic of the underlying field is zero or greater than d. Good reasons for these caveats to exist.

Our results

Theorem [Goyal, Harsha, K., Shankar 2023]

For all $\epsilon > 0$, and all $s > 1/\epsilon^2$, there is a **nearly linear time** algorithm for noisy Hermite interpolation for agreement greater than $(1+\epsilon)d/s.$

Some caveats : d/sn is a constant , characteristic of the underlying field is zero or greater than d. Good reasons for these caveats to exist.

Answers Kopparty's question in a general form

Our results

Theorem [Goyal, Harsha, K., Shankar 2023]

For all $\epsilon > 0$, and all $s > 1/\epsilon^2$, there is a **nearly linear time** algorithm for noisy Hermite interpolation for agreement greater than $(1+\epsilon)d/s.$

Some caveats : d/sn is a constant , characteristic of the underlying field is zero or greater than d. Good reasons for these caveats to exist.

Answers Kopparty's question in a general form

Is an analog of Alekhnovich's result, with even larger error tolerance

Coding theoretic context

Coding theoretic context

Theorem [Goyal, Harsha, K., Shankar 2023]

Over fields of zero or large characteristic, constant rate univariate multiplicity codes can be list decoded up to list-decoding capacity in nearly linear time.

Overview of the algorithm

Overview of the algorithm

Builds upon the ideas in prior work, and in particular relies on the polynomial method

Overview of the algorithm

Builds upon the ideas in prior work, and in particular relies on the polynomial method

Essentially, takes the algorithm of Guruswami-Wang for this problem, and tries to implement each of the steps in nearly linear time

Overview of the algorithm

Builds upon the ideas in prior work, and in particular relies on the polynomial method

Essentially, takes the algorithm of Guruswami-Wang for this problem, and tries to implement each of the steps in nearly linear time

On the way, we stumble upon independently interesting questions and in some settings, interesting answers to them

Overview of the algorithm

Builds upon the ideas in prior work, and in particular relies on the polynomial method

Essentially, takes the algorithm of Guruswami-Wang for this problem, and tries to implement each of the steps in nearly linear time

On the way, we stumble upon independently interesting questions and in some settings, interesting answers to them

Two main steps:

Overview of the algorithm

Builds upon the ideas in prior work, and in particular relies on the polynomial method

Essentially, takes the algorithm of Guruswami-Wang for this problem, and tries to implement each of the steps in nearly linear time

On the way, we stumble upon independently interesting questions and in some settings, interesting answers to them

Two main steps:

1. Construct a differential equation satisfied by all the close enough polynomials

Overview of the algorithm

Builds upon the ideas in prior work, and in particular relies on the polynomial method

Essentially, takes the algorithm of Guruswami-Wang for this problem, and tries to implement each of the steps in nearly linear time

On the way, we stumble upon independently interesting questions and in some settings, interesting answers to them

Two main steps:

1. Construct a differential equation satisfied by all the close enough polynomials

2. Solve the equation to recover all such polynomials

Overview of the algorithm: main steps

Overview of the algorithm: main steps

Input: a function $R : S \to F^s$ and a parameter d

Overview of the algorithm: main steps

Input: a function $R : S \to F^s$ and a parameter d

L : list of all degree d polynomials P with large agreement with R

Overview of the algorithm: main steps

Input: a function $R : S \to F^s$ and a parameter d

L : list of all degree d polynomials P with large agreement with R

1. Construct a differential equation satisfied by all close enough polynomials

Overview of the algorithm: main steps

Input: a function $R : S \to F^s$ and a parameter d

L : list of all degree d polynomials P with large agreement with R

1. Construct a differential equation satisfied by all close enough polynomials

Construct a non-zero polynomial $Q(x, y_1, \cdots, y_m) = A_0(x) + A_1(x)y_1 + \cdots + A_m(x)y_m$

such that for all P in L, $Q(x, P(x), P^{(1)}(x) \cdots, P^{(m-1)}(x)) = 0$

Overview of the algorithm: main steps

Input: a function $R : S \to F^s$ and a parameter d

L : list of all degree d polynomials P with large agreement with R

1. Construct a differential equation satisfied by all close enough polynomials

Construct a non-zero polynomial $Q(x, y_1, \cdots, y_m) = A_0(x) + A_1(x)y_1 + \cdots + A_m(x)y_m$

such that for all P in L, $Q(x, P(x), P^{(1)}(x) \cdots, P^{(m-1)}(x)) = 0$

Why does such a Q exist ?

Overview of the algorithm: main steps

Input: a function $R : S \to F^s$ and a parameter d

L : list of all degree d polynomials P with large agreement with R

1. Construct a differential equation satisfied by all close enough polynomials

Construct a non-zero polynomial $Q(x, y_1, \cdots, y_m) = A_0(x) + A_1(x)y_1 + \cdots + A_m(x)y_m$

such that for all P in L, $Q(x, P(x), P^{(1)}(x) \cdots, P^{(m-1)}(x)) = 0$

Why does such a Q exist ?

How do we find it in near linear time ?

Overview of the algorithm: main steps

But suppose step 1 can be done….

Overview of the algorithm: main steps

But suppose step 1 can be done….

2. Solve the differential equation to recover such polynomials

Overview of the algorithm: main steps

But suppose step 1 can be done….

2. Solve the differential equation to recover such polynomials

Solve for degree d P such that

Overview of the algorithm: main steps

But suppose step 1 can be done….

2. Solve the differential equation to recover such polynomials

Solve for degree d P such that

$$Q(x, P(x), \cdots, P^{(m-1)}(x)) = A_0(x) + \sum_i A_i(x) P^{(i-1)}(x) \equiv 0$$

Overview of the algorithm: main steps

But suppose step 1 can be done….

2. Solve the differential equation to recover such polynomials

Solve for degree d P such that

$$Q(x, P(x), \cdots, P^{(m-1)}(x)) = A_0(x) + \sum_i A_i(x) P^{(i-1)}(x) \equiv 0$$

How many such solutions are there ?

Overview of the algorithm: main steps

But suppose step 1 can be done….

2. Solve the differential equation to recover such polynomials

Solve for degree d P such that

$$Q(x, P(x), \cdots, P^{(m-1)}(x)) = A_0(x) + \sum_i A_i(x) P^{(i-1)}(x) \equiv 0$$

How many such solutions are there ?

We could be in trouble if there are more than constantly many solutions.

Overview of the algorithm: main steps

But suppose step 1 can be done….

2. Solve the differential equation to recover such polynomials

Solve for degree d P such that

$$Q(x, P(x), \cdots, P^{(m-1)}(x)) = A_0(x) + \sum_i A_i(x) P^{(i-1)}(x) \equiv 0$$

How many such solutions are there ?

We could be in trouble if there are more than constantly many solutions.

In general, how do we go about solving such equations…and that too in near linear time ?

Step 1: constructing the differential equation

Step 1: constructing the differential equation

Construct a differential equation satisfied by all close enough polynomials

Step 1: constructing the differential equation

Construct a differential equation satisfied by all close enough polynomials

Construct a non-zero polynomial $Q(x, y_1, \cdots, y_m) = A_0(x) + A_1(x)y_1 + \cdots + A_m(x)y_m$

such that for all P in L, $Q(x, P(x), P^{(1)}(x) \cdots, P^{(m-1)}(x)) = 0$

Step 1: constructing the differential equation

Construct a differential equation satisfied by all close enough polynomials

Construct a non-zero polynomial $Q(x, y_1, \cdots, y_m) = A_0(x) + A_1(x)y_1 + \cdots + A_m(x)y_m$

such that for all P in L, $Q(x, P(x), P^{(1)}(x) \cdots, P^{(m-1)}(x)) = 0$

Set up a system of homogeneous linear equations in the coefficients of Q, with more variables than constraints (based on R).

Step 1: constructing the differential equation

Construct a differential equation satisfied by all close enough polynomials

Construct a non-zero polynomial $Q(x, y_1, \cdots, y_m) = A_0(x) + A_1(x)y_1 + \cdots + A_m(x)y_m$

such that for all P in L, $Q(x, P(x), P^{(1)}(x) \cdots, P^{(m-1)}(x)) = 0$

Set up a system of homogeneous linear equations in the coefficients of Q, with more variables than constraints (based on R).

Solve the system to obtain the Q.

Step 1: constructing the differential equation

Construct a differential equation satisfied by all close enough polynomials

Construct a non-zero polynomial $Q(x, y_1, \cdots, y_m) = A_0(x) + A_1(x)y_1 + \cdots + A_m(x)y_m$

such that for all P in L, $Q(x, P(x), P^{(1)}(x) \cdots, P^{(m-1)}(x)) = 0$

Set up a system of homogeneous linear equations in the coefficients of Q, with more variables than constraints (based on R).

Solve the system to obtain the Q.

Guruswami-Wang showed a way of doing this in poly time. We need to do this faster.

Step 1: constructing the differential equation

Construct a differential equation satisfied by all close enough polynomials

Construct a non-zero polynomial $Q(x, y_1, \cdots, y_m) = A_0(x) + A_1(x)y_1 + \cdots + A_m(x)y_m$

such that for all P in L, $Q(x, P(x), P^{(1)}(x) \cdots, P^{(m-1)}(x)) = 0$

Set up a system of homogeneous linear equations in the coefficients of Q, with more variables than constraints (based on R).

Solve the system to obtain the Q.

Guruswami-Wang showed a way of doing this in poly time. We need to do this faster.

We rely on ideas of Alekhnovich to view this as a case of finding shortest vector (in the degree norm) in a lattice over the univariate polynomial ring.

Step 2: solving the differential equation

Step 2: solving the differential equation

Solve $A_0(x) + A_1(x)P(x) + \cdots + A_m(x)P^{(m-1)}(x) \equiv 0$

Step 2: solving the differential equation

Solve $A_0(x) + A_1(x)P(x) + \cdots + A_m(x)P^{(m-1)}(x) \equiv 0$

Structure of degree d solutions: set of degree d solutions forms an affine subspace of dimension m

Step 2: solving the differential equation

Solve $A_0(x) + A_1(x)P(x) + \cdots + A_m(x)P^{(m-1)}(x) \equiv 0$

Structure of degree d solutions: set of degree d solutions forms an affine subspace of dimension m

Guruswami-Wang: an efficient algorithm that takes Q as input and outputs a basis of this affine space

Step 2: solving the differential equation

Solve $A_0(x) + A_1(x)P(x) + \cdots + A_m(x)P^{(m-1)}(x) \equiv 0$

Structure of degree d solutions: set of degree d solutions forms an affine subspace of dimension m

Guruswami-Wang: an efficient algorithm that takes Q as input and outputs a basis of this affine space

Technical statement

Step 2: solving the differential equation

Solve $A_0(x) + A_1(x)P(x) + \cdots + A_m(x)P^{(m-1)}(x) \equiv 0$

Structure of degree d solutions: set of degree d solutions forms an affine subspace of dimension m

Guruswami-Wang: an efficient algorithm that takes Q as input and outputs a basis of this affine space

Technical statement

A fast algorithm that takes Q as input and outputs a basis of this linear subspace.

Step 2: solving the differential equation

Step 2: solving the differential equation

m = 2 for the rest of this discussion

Step 2: solving the differential equation

m = 2 for the rest of this discussion

Q. Solve $A_0(x) + A_1(x)P(x) + A_2(x)P^{(1)}(x) \equiv 0$

Step 2: solving the differential equation

m = 2 for the rest of this discussion

Q. Solve $A_0(x) + A_1(x)P(x) + A_2(x)P^{(1)}(x) \equiv 0$

Simpler question : solve $A_0(x) + A_1(x)P(x) \equiv 0$

Step 2: solving the differential equation

m = 2 for the rest of this discussion

Q. Solve $A_0(x) + A_1(x)P(x) + A_2(x)P^{(1)}(x) \equiv 0$

Simpler question : solve $A_0(x) + A_1(x)P(x) \equiv 0$

Solution is unique (if one exists)

Step 2: solving the differential equation

m = 2 for the rest of this discussion

Q. Solve $A_0(x) + A_1(x)P(x) + A_2(x)P^{(1)}(x) \equiv 0$

Simpler question : solve $A_0(x) + A_1(x)P(x) \equiv 0$

Solution is unique (if one exists)

Can always find one using long division of univariate polynomials

Step 2: solving the differential equation

m = 2 for the rest of this discussion

Q. Solve $A_0(x) + A_1(x)P(x) + A_2(x)P^{(1)}(x) \equiv 0$

Simpler question : solve $A_0(x) + A_1(x)P(x) \equiv 0$

Solution is unique (if one exists)

Can always find one using long division of univariate polynomials

Can we do this in near linear time ?

Step 2: solving the differential equation

m = 2 for the rest of this discussion

Q. Solve $A_0(x) + A_1(x)P(x) + A_2(x)P^{(1)}(x) \equiv 0$

Simpler question : solve $A_0(x) + A_1(x)P(x) \equiv 0$

Solution is unique (if one exists)

Can always find one using long division of univariate polynomials

Can we do this in near linear time ?

Yes, but not trivial.

Step 2: solving the differential equation

m = 2 for the rest of this discussion

Q. Solve $A_0(x) + A_1(x)P(x) + A_2(x)P^{(1)}(x) \equiv 0$

Simpler question : solve $A_0(x) + A_1(x)P(x) \equiv 0$

Solution is unique (if one exists)

Can always find one using long division of univariate polynomials

Can we do this in near linear time ?

Yes, but not trivial.

Results of Sieveking, Kung, Strassen from 1970s. Beautiful application of FFT, Newton iteration, divide and conquer.

Step 2: solving the differential equation

Simpler question : solve $A_0(x) + A_1(x)P(x) \equiv 0$

Step 2: solving the differential equation

Simpler question : solve $A_0(x) + A_1(x)P(x) \equiv 0$

Write $P(x) = U(x) + x^{d/2}V(x)$ for U, V of degree d/2

Step 2: solving the differential equation

Simpler question : solve $A_0(x) + A_1(x)P(x) \equiv 0$

Write $P(x) = U(x) + x^{d/2}V(x)$ for U, V of degree d/2

Thus, $A_0 + A_1P = A_0 + A_1(U + x^{d/2}V(x)) = (A_0 + A_1U) + x^{d/2}(A_1V) \equiv 0$

Step 2: solving the differential equation

Simpler question : solve $A_0(x) + A_1(x)P(x) \equiv 0$

Write $P(x) = U(x) + x^{d/2}V(x)$ for U, V of degree d/2

Thus, $A_0 + A_1P = A_0 + A_1(U + x^{d/2}V(x)) = (A_0 + A_1U) + x^{d/2}(A_1V) \equiv 0$

This happens if and only if

Step 2: solving the differential equation

Simpler question : solve $A_0(x) + A_1(x)P(x) \equiv 0$

Write $P(x) = U(x) + x^{d/2}V(x)$ for U, V of degree d/2

Thus, $A_0 + A_1P = A_0 + A_1(U + x^{d/2}V(x)) = (A_0 + A_1U) + x^{d/2}(A_1V) \equiv 0$

This happens if and only if

$$A_0(x) + A_1(x)U(x) \equiv 0 \mod x^{d/2}$$

Step 2: solving the differential equation

Simpler question : solve $A_0(x) + A_1(x)P(x) \equiv 0$

Write $P(x) = U(x) + x^{d/2}V(x)$ for U, V of degree d/2

Thus, $A_0 + A_1P = A_0 + A_1(U + x^{d/2}V(x)) = (A_0 + A_1U) + x^{d/2}(A_1V) \equiv 0$

This happens if and only if

$A_0(x) + A_1(x)U(x) \equiv 0 \mod x^{d/2}$ and

Step 2: solving the differential equation

Simpler question : solve $A_0(x) + A_1(x)P(x) \equiv 0$

Write $P(x) = U(x) + x^{d/2}V(x)$ for U, V of degree d/2

Thus, $A_0 + A_1P = A_0 + A_1(U + x^{d/2}V(x)) = (A_0 + A_1U) + x^{d/2}(A_1V) \equiv 0$

This happens if and only if

$A_0(x) + A_1(x)U(x) \equiv 0 \mod x^{d/2}$ and $B(x) + x^{d/2}A_1(x)V(x) \equiv 0 \mod x^d$, where $B = A_0 + A_1U$

Step 2: solving the differential equation

Simpler question : solve $A_0(x) + A_1(x)P(x) \equiv 0$

Write $P(x) = U(x) + x^{d/2}V(x)$ for U, V of degree d/2

Thus, $A_0 + A_1P = A_0 + A_1(U + x^{d/2}V(x)) = (A_0 + A_1U) + x^{d/2}(A_1V) \equiv 0$

This happens if and only if

$A_0(x) + A_1(x)U(x) \equiv 0 \mod x^{d/2}$ and $B(x) + x^{d/2}A_1(x)V(x) \equiv 0 \mod x^d$, where $B = A_0 + A_1U$

For a 'correct' U, B is divisible by $x^{d/2}$.

Step 2: solving the differential equation

Simpler question : solve $A_0(x) + A_1(x)P(x) \equiv 0$

Write $P(x) = U(x) + x^{d/2}V(x)$ for U, V of degree d/2

Thus, $A_0 + A_1P = A_0 + A_1(U + x^{d/2}V(x)) = (A_0 + A_1U) + x^{d/2}(A_1V) \equiv 0$

This happens if and only if

$A_0(x) + A_1(x)U(x) \equiv 0 \mod x^{d/2}$ and $B(x) + x^{d/2}A_1(x)V(x) \equiv 0 \mod x^d$, where $B = A_0 + A_1U$

For a 'correct' U, B is divisible by $x^{d/2}$. Thus, $B(x) + x^{d/2}A_1(x)V(x) \equiv 0 \mod x^d$

Step 2: solving the differential equation

Simpler question : solve $A_0(x) + A_1(x)P(x) \equiv 0$

Write $P(x) = U(x) + x^{d/2}V(x)$ for U, V of degree d/2

Thus, $A_0 + A_1P = A_0 + A_1(U + x^{d/2}V(x)) = (A_0 + A_1U) + x^{d/2}(A_1V) \equiv 0$

This happens if and only if

$A_0(x) + A_1(x)U(x) \equiv 0 \mod x^{d/2}$ and $B(x) + x^{d/2}A_1(x)V(x) \equiv 0 \mod x^d$, where $B = A_0 + A_1U$

For a 'correct' U, B is divisible by $x^{d/2}$. Thus, $B(x) + x^{d/2}A_1(x)V(x) \equiv 0 \mod x^d$ if and only if

Step 2: solving the differential equation

Simpler question : solve $A_0(x) + A_1(x)P(x) \equiv 0$

Write $P(x) = U(x) + x^{d/2}V(x)$ for U, V of degree d/2

Thus, $A_0 + A_1P = A_0 + A_1(U + x^{d/2}V(x)) = (A_0 + A_1U) + x^{d/2}(A_1V) \equiv 0$

This happens if and only if

$A_0(x) + A_1(x)U(x) \equiv 0 \mod x^{d/2}$ and $B(x) + x^{d/2}A_1(x)V(x) \equiv 0 \mod x^d$, where $B = A_0 + A_1U$

For a 'correct' U, B is divisible by $x^{d/2}$. Thus, $B(x) + x^{d/2}A_1(x)V(x) \equiv 0 \mod x^d$ if and only if $\tilde{B}(x) + A_1(x)V(x) \equiv 0 \mod x^{d/2}$ for $\tilde{B} = B(x)/x^{d/2}$

Step 2: solving the differential equation

Simpler question : solve $A_0(x) + A_1(x)P(x) \equiv 0$

Write $P(x) = U(x) + x^{d/2}V(x)$ for U, V of degree d/2

Thus, $A_0 + A_1P = A_0 + A_1(U + x^{d/2}V(x)) = (A_0 + A_1U) + x^{d/2}(A_1V) \equiv 0$

This happens if and only if

$A_0(x) + A_1(x)U(x) \equiv 0 \mod x^{d/2}$ and $B(x) + x^{d/2}A_1(x)V(x) \equiv 0 \mod x^d$, where $B = A_0 + A_1U$

For a 'correct' U, B is divisible by $x^{d/2}$. Thus, $B(x) + x^{d/2}A_1(x)V(x) \equiv 0 \mod x^d$ if and only if $\tilde{B}(x) + A_1(x)V(x) \equiv 0 \mod x^{d/2}$ for $\tilde{B} = B(x)/x^{d/2}$

Thus, U and V must satisfy

Step 2: solving the differential equation

Simpler question : solve $A_0(x) + A_1(x)P(x) \equiv 0$

Write $P(x) = U(x) + x^{d/2}V(x)$ for U, V of degree d/2

Thus, $A_0 + A_1P = A_0 + A_1(U + x^{d/2}V(x)) = (A_0 + A_1U) + x^{d/2}(A_1V) \equiv 0$

This happens if and only if

$A_0(x) + A_1(x)U(x) \equiv 0 \mod x^{d/2}$ and $B(x) + x^{d/2}A_1(x)V(x) \equiv 0 \mod x^d$, where $B = A_0 + A_1U$

For a 'correct' U, B is divisible by $x^{d/2}$. Thus, $B(x) + x^{d/2}A_1(x)V(x) \equiv 0 \mod x^d$ if and only if $\tilde{B}(x) + A_1(x)V(x) \equiv 0 \mod x^{d/2}$ for $\tilde{B} = B(x)/x^{d/2}$

Thus, U and V must satisfy

$A_0(x) + A_1(x)U(x) \equiv 0 \mod x^{d/2}$ and $\tilde{B}(x) + A_1(x)V(x) \equiv 0 \mod x^{d/2}$

Step 2: solving the differential equation

Simpler question : solve $A_0(x) + A_1(x)P(x) \equiv 0$

Step 2: solving the differential equation

Simpler question : solve $A_0(x) + A_1(x)P(x) \equiv 0$

Write $P(x) = U(x) + x^{d/2}V(x)$ for U, V of degree d/2

Step 2: solving the differential equation

Simpler question : solve $A_0(x) + A_1(x)P(x) \equiv 0$

Write $P(x) = U(x) + x^{d/2}V(x)$ for U, V of degree d/2

Thus, U and V must satisfy

Step 2: solving the differential equation

Simpler question : solve $A_0(x) + A_1(x)P(x) \equiv 0$

Write $P(x) = U(x) + x^{d/2}V(x)$ for U, V of degree d/2

Thus, U and V must satisfy

$A_0(x) + A_1(x)U(x) \equiv 0 \mod x^{d/2}$

Step 2: solving the differential equation

Simpler question : solve $A_0(x) + A_1(x)P(x) \equiv 0$

Write $P(x) = U(x) + x^{d/2}V(x)$ for U, V of degree d/2

Thus, U and V must satisfy

$A_0(x) + A_1(x)U(x) \equiv 0 \mod x^{d/2}$ and $\tilde{B}(x) + A_1(x)V(x) \equiv 0 \mod x^{d/2}$

Step 2: solving the differential equation

Simpler question : solve $A_0(x) + A_1(x)P(x) \equiv 0$

Write $P(x) = U(x) + x^{d/2}V(x)$ for U, V of degree d/2

Thus, U and V must satisfy

$A_0(x) + A_1(x)U(x) \equiv 0 \mod x^{d/2}$ and $\tilde{B}(x) + A_1(x)V(x) \equiv 0 \mod x^{d/2}$

Two sub-problems of about half the size, together with pre and post processing that is nearly linear time using FFT

Step 2: solving the differential equation

Simpler question : solve $A_0(x) + A_1(x)P(x) \equiv 0$

Write $P(x) = U(x) + x^{d/2}V(x)$ for U, V of degree d/2

Thus, U and V must satisfy

$A_0(x) + A_1(x)U(x) \equiv 0 \mod x^{d/2}$ and $\tilde{B}(x) + A_1(x)V(x) \equiv 0 \mod x^{d/2}$

Two sub-problems of about half the size, together with pre and post processing that is nearly linear time using FFT

Recurse…..

Step 2: solving the differential equation

Extension to solving $A_0(x) + A_1(x)P(x) + A_2(x)P^{(1)}(x) \equiv 0$

Step 2: solving the differential equation

Extension to solving $A_0(x) + A_1(x)P(x) + A_2(x)P^{(1)}(x) \equiv 0$

Natural attempt: Try and decompose P into polynomials of half the degree as before and obtain differential equations that these lower degree polynomials satisfy

Step 2: solving the differential equation

Extension to solving $A_0(x) + A_1(x)P(x) + A_2(x)P^{(1)}(x) \equiv 0$

Natural attempt: Try and decompose P into polynomials of half the degree as before and obtain differential equations that these lower degree polynomials satisfy

Technical issue: Solutions are no longer unique and we cannot afford to branch at every recursive step

Step 2: solving the differential equation

Extension to solving $A_0(x) + A_1(x)P(x) + A_2(x)P^{(1)}(x) \equiv 0$

Natural attempt: Try and decompose P into polynomials of half the degree as before and obtain differential equations that these lower degree polynomials satisfy

Technical issue: Solutions are no longer unique and we cannot afford to branch at every recursive step

Guruswami and Wang: Give an efficient way of getting our hands on a basis of the affine space of solutions

Step 2: solving the differential equation

Extension to solving $A_0(x) + A_1(x)P(x) + A_2(x)P^{(1)}(x) \equiv 0$

Natural attempt: Try and decompose P into polynomials of half the degree as before and obtain differential equations that these lower degree polynomials satisfy

Technical issue: Solutions are no longer unique and we cannot afford to branch at every recursive step

Guruswami and Wang: Give an efficient way of getting our hands on a basis of the affine space of solutions

We try to construct each of these basis elements one by one

Step 2: solving the differential equation

Extension to solving $A_0(x) + A_1(x)P(x) + A_2(x)P^{(1)}(x) \equiv 0$

Natural attempt: Try and decompose P into polynomials of half the degree as before and obtain differential equations that these lower degree polynomials satisfy

Technical issue: Solutions are no longer unique and we cannot afford to branch at every recursive step

Guruswami and Wang: Give an efficient way of getting our hands on a basis of the affine space of solutions

We try to construct each of these basis elements one by one

For each such basis vector, we construct a proxy differential equation with a unique solution

Step 2: solving the differential equation

Extension to solving $A_0(x) + A_1(x)P(x) + A_2(x)P^{(1)}(x) \equiv 0$

Natural attempt: Try and decompose P into polynomials of half the degree as before and obtain differential equations that these lower degree polynomials satisfy

Technical issue: Solutions are no longer unique and we cannot afford to branch at every recursive step

Guruswami and Wang: Give an efficient way of getting our hands on a basis of the affine space of solutions

We try to construct each of these basis elements one by one

For each such basis vector, we construct a proxy differential equation with a unique solution

And, a version of divide and conquer as we saw it works there

Open questions

Open questions

- Generalization to multivariate polynomials ? Many technical issues to overcome, including a multivariate generalization of the differential equation solver.

Open questions

- Generalization to multivariate polynomials ? Many technical issues to overcome, including a multivariate generalization of the differential equation solver.

- Understanding the state of standard noisy polynomial interpolation beyond the Johnson bound

Open questions

- Generalization to multivariate polynomials ? Many technical issues to overcome, including a multivariate generalization of the differential equation solver.

- Understanding the state of standard noisy polynomial interpolation beyond the Johnson bound
    - explicit construction of evaluation points for which the solution size is small

Open questions

- Generalization to multivariate polynomials ? Many technical issues to overcome, including a multivariate generalization of the differential equation solver.

- Understanding the state of standard noisy polynomial interpolation beyond the Johnson bound
    - explicit construction of evaluation points for which the solution size is small
    - efficient algorithms for noisy interpolation for such point sets

Open questions

- Generalization to multivariate polynomials ? Many technical issues to overcome, including a multivariate generalization of the differential equation solver.

- Understanding the state of standard noisy polynomial interpolation beyond the Johnson bound
    - explicit construction of evaluation points for which the solution size is small
    - efficient algorithms for noisy interpolation for such point sets
    - eventually, near linear time algorithms for noisy interpolation in this regime

Thank You!