Interpolation, Rational Reconstruction and Modular Algorithms

Claus Fieker

February 14, 2023

Many problems in computational algebra suffer the same problem:

- small input
- rapid growth
- small result

A common remidy is thus to solve a different problem by projecting s.w. where all objects are small - and hope that this is enough.

Many problems in computational algebra suffer the same problem:

- small input
- rapid growth
- small result

A common remidy is thus to solve a different problem by projecting s.w. where all objects are small - and hope that this is enough.

Many problems in computational algebra suffer the same problem:

- small input
- rapid growth
- small result

A common remidy is thus to solve a different problem by projecting s.w. where all objects are small - and hope that this is enough.

- Exact determinants over rings with large objects, e.g.
    - the ring of integers
    - polynomials over some (finite) field
    - field of (univariate) rationals functions
- roots of (univariate) polynomials over the same rings

Instead of computing in a ring R, we can try R/A for some ideal A. E.g. $\mathbb{Z}/n\mathbb{Z}$: all numbers are small, if n is prime, then we get a field. To get back: a natural candidate is the unique representative in $-n/2 \ldots n/2$.
Or: $R = k[x]$ and n any (linear) polynomial.
Note: $f = q(x-a) + r$ (euclidean division) iff $r = f(a)$.
Chinese remainder theorem allows to combine results in both cases, allowing a large n to be made up of small p - or a large degree n out of many linear ones. (Called evaluation and interpolation)

Interpolation works **o**nly if the result is unique - for all p one can compute the correct matching result.
Problem: not all problems have unique solutions....
$f = (x - 10^6 + 1)(x - 10^6 - 1)$ has 2 roots modulo every prime:
$f \equiv (x)(x+1) \bmod 3$, $f = (x+1)(x+4) \bmod 5$,
$f \equiv (t+927)(t+929) \bmod 1009$, and
$f \equiv (t+843)(t+845) \bmod 1013$. Which pairs should be combined?
f has 4 solutions modulo every product of 2 primes.

Interpolation works **o**nly if the result is unique - for all p one can compute the correct matching result.
Problem: not all problems have unique solutions....
$f = (x - 10^6 + 1)(x - 10^6 - 1)$ has 2 roots modulo every prime:
$f \equiv (x)(x+1) \bmod 3$, $f = (x+1)(x+4) \bmod 5$,
$f \equiv (t+927)(t+929) \bmod 1009$, and
$f \equiv (t+843)(t+845) \bmod 1013$. Which pairs should be combined?
f has 4 solutions modulo every product of 2 primes.

Interpolation works **o**nly if the result is unique - for all p one can compute the correct matching result.
Problem: not all problems have unique solutions....
$f = (x-10^6+1)(x-10^6-1)$ has 2 roots modulo every prime:
$f \equiv (x)(x+1) \bmod 3$, $f = (x+1)(x+4) \bmod 5$,
$f \equiv (t+927)(t+929) \bmod 1009$, and
$f \equiv (t+843)(t+845) \bmod 1013$. Which pairs should be combined?
f has 4 solutions modulo every product of 2 primes.

We have 2 recipies to obtain large n from small ones

- Chinese remaindering/ interpolation
- lifting

Lifting generically takes a solution $\bmod p^k$ and computes from there the solution $\bmod p^l$ for $l > k$ - avoiding the recombination problem.

For the rest of this talk we focus on details for (multivariate) polynomial rings: the hidden problem of finding a canonical, nice representative in R of an element given implicitly in R/A.
Here A is only implicit as it is defined by evaluation at strategically chosen points.
This is "trivial" for some rings - and hard to unkown for others.

Note

Modular techniques apply whenever the projection/ lift is effective - and a unique solution can be obtained.

Let $R = K(x_1, \ldots, x_d)$ and

$$h(\underline{x}) = \frac{f(\underline{x})}{g(\underline{x})}$$

for some f, $g \in K[x_1, \ldots, x_d]$.

Task (Rational Multivariate Interpolation)

Given (suitable) $\underline{\alpha}_i \in K^d$ *and* $y_i = h(\underline{\alpha}_i)$ *find* f_i, $g_i \in K$ *and* m_i, $n_i \in \mathbb{N}^d$ *s.th.*

$$f = \sum f_i \underline{x}^{m_i} \quad \text{and} \quad g = \sum g_i \underline{x}^{n_i}.$$

For this talk we assume that we can choose $\underline{\alpha}_i$ freely and have access to an oracle (black-box representation of h) computing y_i on demand.

We will use $K = \mathbb{Q}$, $\mathbb{F}_p$.

Let $R = K(x_1, \ldots, x_d)$ and

$$h(\underline{x}) = \frac{f(\underline{x})}{g(\underline{x})}$$

for some f, $g \in K[x_1, \ldots, x_d]$.

Task (Rational Multivariate Interpolation)

Given (suitable) $\underline{\alpha}_i \in K^d$ *and* $y_i = h(\underline{\alpha}_i)$ *find* f_i, $g_i \in K$ *and* m_i, $n_i \in \mathbb{N}^d$ *s.th.*

$$f = \sum f_i \underline{x}^{m_i} \quad \textit{and} \quad g = \sum g_i \underline{x}^{n_i}.$$

For this talk we assume that we can choose $\underline{\alpha}_i$ freely and have access to an oracle (black-box representation of h) computing y_i on demand.
We will use $K = \mathbb{Q}$, $\mathbb{F}_p$.

Polynomials

Classical interpolation:

- given

$$\alpha_i, y_i \in K, \quad 1 \le i \le n,\ \alpha_i \text{ pairwise distinct}$$

- find the (unique)

$$f \in K[x],\ \deg f < n \text{ s.th. } f(\alpha_i) = y_i.$$

There are many (explicit) formulas known.

Polynomials

Classical interpolation:

- given

$$\alpha_i, y_i \in K, \quad 1 \leq i \leq n, \ \alpha_i \text{ pairwise distinct}$$

- find the (unique)

$$f \in K[x], \ \deg f < n \text{ s.th. } f(\alpha_i) = y_i.$$

There are many (explicit) formulas known.

Polynomials

Important

- *f can be found with $\tilde{O}(n)$ operations in K only.*
- *$\tilde{O}(n)$: We ignore $\log^?(n)$ factors in the analysis.*

Note

- *The obvious, classical, solutions take $O(n^2)$ or even $O(n^3)$ operations in K.*
- *Assuming $n \gg 0$: fast methods are practical.*
- *This comparison is not fair and omits lots of important details.*
- *It is possible to add more information afterwards.*

Fast Methods - Products

It is well known that products can be computed using Karatsuba's trick or even using FFTs.
We need for the product of univariate polynomials f and $g \in K[x]$ of degrees n and m operations in K:

- Classical: $O(nm)$
- Karatsuba, if $n = m$: $O(n^{\log_2 3})$
- FFT, if $n = m$: $O(n \log n \log^* \log n) =: \tilde{O}(n)$

Why does this matter?
Multiplication is **not** time-associative!
The order of operations matters - the time can vary by magnitudes.

Fast Methods - Products

It is well known that products can be computed using Karatsuba's trick or even using FFTs.
We need for the product of univariate polynomials f and $g \in K[x]$ of degrees n and m operations in K:

- Classical: $O(nm)$
- Karatsuba, if $n = m$: $O(n^{\log_2 3})$
- FFT, if $n = m$: $O(n \log n \log^* \log n) =: \tilde{O}(n)$

Why does this matter?
Multiplication is **not** time-associative!
The order of operations matters - the time can vary by magnitudes.

Fast Methods - Products

It is well known that products can be computed using Karatsuba's trick or even using FFTs.

We need for the product of univariate polynomials f and $g \in K[x]$ of degrees n and m operations in K:

- Classical: $O(nm)$
- Karatsuba, if $n = m$: $O(n^{\log_2 3})$
- FFT, if $n = m$: $O(n \log n \log^* \log n) =: \tilde{O}(n)$

Why does this matter?

Multiplication is **not** time-associative!

The order of operations matters - the time can vary by magnitudes.

Product - Tree

Warming up: given $f_i \in K[x]$, s.th. $\deg f_i = n$ for all i. Task: compute the product

$$\prod f_i$$

Iterative: $(\ldots(((f_1 f_2) f_3) f_4) \ldots f_r)$
Clever: $(((f_1 f_2)(f_3 f_4)) \ldots)$

Fact

The total number of K operations for the iterative method is $O(r^2 n^2)$, while it is $\tilde{O}(rn)$ in the 2nd case!

Why?

Classical: $f_1 f_2$ takes $O(n^2)$ ops, result has degree $2n$.
$((f_1 f_2) f_3)$ takes $O(2n^2)$ ops, result is degree $3n$.
Total: $\sum_{i=1}^{r} O(in^2) = O(r^2 n^2)$.
Clever: all products are of polys of same degree.
$r/2\tilde{O}(n)$ for the initial products, $r/4\tilde{O}(2n)$ for the next level, ...,
total: $\sum_{i=1}^{\log_2 r} r/2^i \tilde{O}(2^{i-1} n) = \tilde{O}(nr)$
This matters!

Why?

Classical: $f_1 f_2$ takes $O(n^2)$ ops, result has degree $2n$.
$((f_1 f_2) f_3)$ takes $O(2n^2)$ ops, result is degree $3n$.
Total: $\sum_{i=1}^{r} O(in^2) = O(r^2 n^2)$.
Clever: all products are of polys of same degree.
$r/2\tilde{O}(n)$ for the initial products, $r/4\tilde{O}(2n)$ for the next level, …,
total: $\sum_{i=1}^{\log_2 r} r/2^i \tilde{O}(2^{i-1} n) = \tilde{O}(nr)$
This matters!

Why?

Classical: $f_1 f_2$ takes $O(n^2)$ ops, result has degree $2n$.
$((f_1 f_2) f_3)$ takes $O(2n^2)$ ops, result is degree $3n$.
Total: $\sum_{i=1}^{r} O(in^2) = O(r^2 n^2)$.
Clever: all products are of polys of same degree.
$r/2\tilde{O}(n)$ for the initial products, $r/4\tilde{O}(2n)$ for the next level, $\ldots$,
total: $\sum_{i=1}^{\log_2 r} r/2^i \tilde{O}(2^{i-1} n) = \tilde{O}(nr)$
This matters!

Why?

Classical: $f_1 f_2$ takes $O(n^2)$ ops, result has degree $2n$.
$((f_1 f_2) f_3)$ takes $O(2n^2)$ ops, result is degree $3n$.
Total: $\sum_{i=1}^{r} O(in^2) = O(r^2 n^2)$.
Clever: all products are of polys of same degree.
$r/2\tilde{O}(n)$ for the initial products, $r/4\tilde{O}(2n)$ for the next level, ...,
total: $\sum_{i=1}^{\log_2 r} r/2^i \tilde{O}(2^{i-1} n) = \tilde{O}(nr)$
This matters!

Why?

Classical: $f_1 f_2$ takes $O(n^2)$ ops, result has degree $2n$.
$((f_1 f_2) f_3)$ takes $O(2n^2)$ ops, result is degree $3n$.
Total: $\sum_{i=1}^{r} O(in^2) = O(r^2 n^2)$.
Clever: all products are of polys of same degree.
$r/2\tilde{O}(n)$ for the initial products, $r/4\tilde{O}(2n)$ for the next level, $\ldots$,
total: $\sum_{i=1}^{\log_2 r} r/2^i \tilde{O}(2^{i-1} n) = \tilde{O}(nr)$
This matters!

Why?

Classical: $f_1 f_2$ takes $O(n^2)$ ops, result has degree $2n$.
$((f_1 f_2) f_3)$ takes $O(2n^2)$ ops, result is degree $3n$.
Total: $\sum_{i=1}^{r} O(in^2) = O(r^2 n^2)$.
Clever: all products are of polys of same degree.
$r/2\tilde{O}(n)$ for the initial products, $r/4\tilde{O}(2n)$ for the next level, ...,
total: $\sum_{i=1}^{\log_2 r} r/2^i \tilde{O}(2^{i-1}n) = \tilde{O}(nr)$
This matters!

Why?

Classical: $f_1 f_2$ takes $O(n^2)$ ops, result has degree $2n$.
$((f_1 f_2) f_3)$ takes $O(2n^2)$ ops, result is degree $3n$.
Total: $\sum_{i=1}^{r} O(in^2) = O(r^2 n^2)$.
Clever: all products are of polys of same degree.
$r/2\tilde{O}(n)$ for the initial products, $r/4\tilde{O}(2n)$ for the next level, ...,
total: $\sum_{i=1}^{\log_2 r} r/2^i \tilde{O}(2^{i-1} n) = \tilde{O}(nr)$
This matters!

Product - Tree

A different way of looking at this:
The expression $\prod f_i$ can be evaluated on a computer using an evaluation tree, parsing tree, ...
Classical: corresponds to a narrow, deep tree, degrading into a line
Clever: is a binary tree of minimal depth.
In either case, the size of the intermediate results correspond to the level of the tree: growing from leaf to root.
However, the clever method needs more storage, minimally $\log_2 r$, typically $r/2$.

Interpolation = Chinese Remainder Theorem

Interpolation: $f(a_i) = b_i$ $(1 \le i \le n)$ and $\deg f < n$.
Division with remainder: $f = q(x - a_i) + b_i$, so

$$f \equiv b_i \bmod x - a_i$$

So: CRT will find f s.th. $f \equiv b_i \bmod x - a_i$ and f is modulo $\prod(x - a_i)$ unique, so $\deg f < n$.
Why? CRT can use product trees!

Interpolation = Chinese Remainder Theorem

Interpolation: $f(a_i) = b_i$ $(1 \leq i \leq n)$ and $\deg f < n$.
Division with remainder: $f = q(x - a_i) + b_i$, so

$$f \equiv b_i \bmod x - a_i$$

So: CRT will find f s.th. $f \equiv b_i \bmod x - a_i$ and f is modulo $\prod(x - a_i)$ unique, so $\deg f < n$.
Why? CRT can use product trees!

CRT - Tree

Given a_i, b_i in K, find $f \in K[x]$ s.th. $f(a_i) = b_i$ or, equivalently $f \equiv b_i \bmod x - a_i$.

Define $g_i := x - a_i$ and find $f_{2i-1,2i}$ s.th. $f_{2i-1,2i} \equiv b_{2i-1}$ and $f_{2i} \equiv b_{2i}$, set $g_{2i-1,2i} = g_{2i-1}g_{2i}$ for $i = 1, \ldots, r/2$.

Then iterate: find $f_{4i-3,4i-2,4i-1,4i} \equiv f_{4i-3,4i-2} \bmod g_{4i-3,4i-2}$ and $f_{4i-3,4i-2,4i-1,4i} \equiv f_{4i-1,4i} \bmod g_{4i-1,4i}$ and $g_{4i-3,4i-2,4i-1,4i} = g_{4i-3,4i-2}g_{4i-1,4i}$

...

Clearly, this works, but why bother?

CRT - Tree

Given a_i, b_i in K, find $f \in K[x]$ s.th. $f(a_i) = b_i$ or, equivalently $f \equiv b_i \bmod x - a_i$.

Define $g_i := x - a_i$ and find $f_{2i-1,2i}$ s.th. $f_{2i-1,2i} \equiv b_{2i-1}$ and $f_{2i} \equiv b_{2i}$, set $g_{2i-1,2i} = g_{2i-1} g_{2i}$ for $i = 1, \ldots, r/2$.

Then iterate: find $f_{4i-3,4i-2,4i-1,4i} \equiv f_{4i-3,4i-2} \bmod g_{4i-3,4i-2}$ and $f_{4i-3,4i-2,4i-1,4i} \equiv f_{4i-1,4i} \bmod g_{4i-1,4i}$ and

$g_{4i-3,4i-2,4i-1,4i} = g_{4i-3,4i-2} g_{4i-1,4i}$

...

Clearly, this works, but why bother?

CRT - Tree

Given a_i, b_i in K, find $f \in K[x]$ s.th. $f(a_i) = b_i$ or, equivalently $f \equiv b_i \bmod x - a_i$.
Define $g_i := x - a_i$ and find $f_{2i-1,2i}$ s.th. $f_{2i-1,2i} \equiv b_{2i-1}$ and $f_{2i} \equiv b_{2i}$, set $g_{2i-1,2i} = g_{2i-1}g_{2i}$ for $i = 1, \ldots, r/2$.
Then iterate: find $f_{4i-3,4i-2,4i-1,4i} \equiv f_{4i-3,4i-2} \bmod g_{4i-3,4i-2}$ and $f_{4i-3,4i-2,4i-1,4i} \equiv f_{4i-1,4i} \bmod g_{4i-1,4i}$ and $g_{4i-3,4i-2,4i-1,4i} = g_{4i-3,4i-2}g_{4i-1,4i}$
...
Clearly, this works, but why bother?

Single CRT

Given a, $b \in K[t]$, $\deg a$, $\deg b = n-1$, f, $g \in K[t]$, coprime, $\deg f$. $\deg g = n$, solve the CRT problem:
Find $h \equiv a \bmod f$ and $h \equiv b \bmod g$.
Find u and v s.th. $1 = \gcd(f,g) = uf + vg$ using the Euclidean algorithm.
Then $h \equiv vga + ufb$ (Note: $vg = 1 - uf$, saving a multiplication).
So, this needs

- 1 $\gcd$ degree n
- 4 products: 2 degree n by n and 2 degree $2n$ by n
- 1 division: degree $3n$ by $2n$

All can be done **fast**, ie $\tilde{O}(n)$
Doing this iteratively: same problem as the product.

(univariate) Interpolation: Summary

Given n points, the interpolation polynomial can be found using

$$\tilde{O}(n)$$

operations in K.
If neccessary, points can be added later - without starting from scratch.
In reality, I do not use fast methods until the degree is large (enough) of course.
The "same" tree can be used for multi-point evaluation.

(univariate) Interpolation: Summary

Given n points, the interpolation polynomial can be found using

$$\tilde{O}(n)$$

operations in K.

If neccessary, points can be added later - without starting from scratch.

In reality, I do not use fast methods until the degree is large (enough) of course.

The "same" tree can be used for multi-point evaluation.

(univariate) Interpolation: Summary

Given n points, the interpolation polynomial can be found using

$$\tilde{O}(n)$$

operations in K.
If neccessary, points can be added later - without starting from scratch.
In reality, I do not use fast methods until the degree is large (enough) of course.
The “same” tree can be used for multi-point evaluation.

Rational Interpolation

This is an application of rational reconstruction or, in $\mathbb{Q}$, Farey lifting.

Task

Given

$$y_i = f(\alpha_i)/g(\alpha_i) \quad , 1 \le i \le n$$

find f, $g \in K[x]$.

Here we need additional restrictions: $\deg f \le n_f$, $\deg g \le n_g$ and $n_f + n_g < n$.

Rational Interpolation

Theorem

There exist "unique" $f, g \in K[x]$ solving the interpolation problem:

$$y_i = \frac{f(\alpha_i)}{g(\alpha_i)}$$

subject to $\deg f \leq n_f$, $\deg g \leq n_g$.
Furthermore, f *and* g *can be found in* $\tilde{O}(n)$ *operations in* K.

Rational Interpolation

Idea:

- First find $\tilde{f} \in K[x]$ s.th. $\tilde{f}(\alpha_i) = y_i$,
- then find f, g s.th. $f \equiv g\tilde{f} \bmod \prod x - \alpha_i$

The first is (just) univariate interpolation, the second step is using (essentially) the extended Euclidean algorithm, stopping when the remainder is small enough.

Note: implicit here is $g(\alpha_i) \neq 0$

EEA

Simplifying: $a := \prod x - a_i$ and $b \in K[x]$ sth. $b(a_i) = b_i$, we want $f, g \in K[x]$ sth.

$$\frac{f}{g}(a_i) = b(a_i) = b_i$$

This implies:

$$f \equiv gb \bmod a$$

Task

Given a, b find f and g sth.

$$\frac{f}{g} = b \iff f \equiv bg \bmod a$$

Also known as rational reconstruction or, Farey lifting.

EEA

Simplifying: $a := \prod x - a_i$ and $b \in K[x]$ sth. $b(a_i) = b_i$, we want $f, g \in K[x]$ sth.

$$\frac{f}{g}(a_i) = b(a_i) = b_i$$

This implies:

$$f \equiv gb \bmod a$$

Task

Given a, b find f and g sth.

$$\frac{f}{g} = b \iff f \equiv bg \bmod a$$

Also known as rational reconstruction or, Farey lifting.

Monagan

Given a, $b \in K[x]$, Monagan defines the extended euclidean algorithm via $R_0 = (a, 0)$, $R_1 = (b, 1)$, $R_i = (r_i, t_i)$ and then $q_i = r_{i-1} \operatorname{div} r_i$ and $R_{i+1} = (r_{i-1} - q_i r_i, t_{i-1} - q_i t_i)$.

Fact

- *If* $r_{i+1} = 0$, *then* $r_i = \gcd(a, b)$
- $\forall i : \deg r_i + \deg t_i + \deg q_i = \deg a$
- $\sum \deg q_i = \deg a$
- $\forall i : bt_i \equiv r_i \bmod a \iff b \equiv \frac{r_i}{t_i} \bmod a$

Monagan ctd.

Generically, $\deg q_i = 1$, Monagan suggests using i sth. $\deg q_i$ is maximal as "the" solution:

$$\frac{f}{g} = \frac{r_i}{t_i}$$

If $\deg a$ is large enough $(\deg a > 2(\deg f + \deg g))$ this i is unique and all works.
If the degrees of f and g are known, then this can be used as a stopping condition as well and all works.
Monagan uses the fast $\gcd$ methods to achieve a runtime $\tilde{O}(n)$ again.

Monagan ctd.

Generically, $\deg q_i = 1$, Monagan suggests using i sth. $\deg q_i$ is maximal as "the" solution:

$$\frac{f}{g} = \frac{r_i}{t_i}$$

If $\deg a$ is large enough ($\deg a > 2(\deg f + \deg g)$) this i is unique and all works.
If the degrees of f and g are known, then this can be used as a stopping condition as well and all works.
Monagan uses the fast $\gcd$ methods to achieve a runtime $\tilde{O}(n)$ again.

Rational Reconstruction

A similar construction is applied to supplement the CRT for rational solutions.
Given prime numbers p_i and values y_i, find f, $g \in \mathbb{Z}$ s.th.

$$g \bmod p_i \neq 0 \quad \text{and} \quad gy_i \equiv f \bmod p_i.$$

If $2|f| < A$, $0 < g < B$ and $AB \leq M = \prod p_i$ then this is unique.
This can be phrased as a lattice problem, solved using LLL or using continued fractions via the extended Euclidean algorithm.
Again, the runtime is $\tilde{O}(\log M)$.

Polynomials I

To warm up: $f = \sum f_i \underline{x}_i^{m_i}$ for $m_i \in \mathbb{N}^d$.
Given $S \subset \mathbb{N}^d$, $|S| = n < \infty$, $m_i \in S$ (so S is a superset of the support of f).

Theorem

Then, given pairwise distinct $\underline{\alpha}_i \in K^d$ and $y_i \in K$ we (mostly) can find the unique f s.th.

$$f(\underline{\alpha}_i) = y_i$$

using linear algebra in time $O(n^\omega)$.

(The mostly refers to things like $f(x, y) = xy$ where choosing $\underline{\alpha}_i = (0, i)$ is not going to work. If the evaluation points are "random" the Schwartz-Zippel Lemma implies the "mostly")
If only the degree b (or a bound) is known, we need $n = b^d$. . .

Polynomials II

Using the (unique) univariate case, we can obtain a different algorithm - with a sometimes better complexity.
We illustrate this in 2 variables.
Choosing $\underline{\alpha}_{i,j} = (\mu_i, \nu_j)$ we can, fixing j, use the univariate case to find $f_j \in K[x_1]$ s.th. $f_j(\mu_i) = f(\mu_i, \nu_j)$.
Now using the interpolation over $K(x_1)$ to solve $f(x_1, \mu_j) = f_j$ we can find the unique solution.
Initially $f \in K(x_1)[x_2]$ only, but since by assumption the solution $f \in K[x_1, x_2]$ is unique, we're done.
This takes $\tilde{O}(d)$ operations in K to find f_j and then $\tilde{O}(d)$ operations in $K(x_1)$ to find f.

Polynomials III

A hybrid approach: choosing $\underline{\alpha}_i = (\mu_i, \nu_2, \dots, \nu_d)$ we can find $f_1(x_1) = f(x_1, \nu_2, \dots, \nu_d)$ giving, generically, the degrees $D_1 \subset \mathbb{N}$ in which x_1 occurs in f.
Repeating this with $\underline{\alpha}_i = (\nu_1, \dots, \mu_i, \dots, \nu_d)$ we can find all degree sets D_i for x_i, this then gives a superset for the support of f as $S \subseteq \prod D_i$.
This can be much smaller than the generic case. Using the linear algebra then is efficient.

Polynomials IV - Linear Recurrence

Choosing clever evaluation points we can obtain a sparse algorithm. Let $\underline{\alpha}_{i,j} = \beta_j(p_1^i, \ldots, p_d^i)$ for suitable numbers p_i and $\beta_j \in K$. Then $y_{i,j} = f(\underline{\alpha}_{i,j})$, $1 \leq j \leq d$ defines many univariate interpolation problems. We find $f_i \in \mathbb{Q}[z]$ s.th. $f_i(\beta_j) = y_{i,j}$, so $f_i(z) = f(zp_1^i, \ldots, zp_d^i)$. Analysing the coefficients $f_{i,l}$ of f_i we see that

$$\begin{aligned} f_{i,l} &= \sum_{|m_t|=l} c_t \prod_k (p_k^i)^{m_{t,k}} \\ &= \sum_{|m_t|=l} c_t \prod_k (p_k^{m_{t,k}})^i =: \sum_{|m_t|=l} c_t \beta_t^i \end{aligned}$$

Here we have 2 sets of unknowns: the c_t and the m_t. The degrees l however are known from the f_i!

Polynomials IV - Linear Recurrence

$$f_{i,l} = \sum_{|m_t|=l} c_t \beta_t^i$$

For each l, this is well known to be a linear recurrence (of unknown length). Using the Berlekamp-Massey algorithm we can obtain a recurrence of degree $< n$ from $2n$ terms. This finds an auxiliary polynomial $T \in K[z]$ s.th.

$$T(z) = \prod (z - \beta_t)$$

Problem: find m_t from $\beta_t \ldots$

Ben-Or, Tiwari

Using pairwise coprime (or distinct primes) $p_i \in \mathbb{Z}$ (for $K = \mathbb{Q}$), the exponents m_t can be recovered from the β_t using factorisation!
The number of evaluation points depends on the degree of f_i, hence the total degree, and the number of m_t of the same degree.
We need $\deg f_i$ many β_j and $2\#\{m_t \mid |m_t| = l\}$ many i, so $2\deg f_i \#\{m_t\}$ many in total.
We note that, due to the high powers of p_i used, the rational coefficients will be huge.
Once the exponents, the monomials, are known, linear algebra will find the coefficients.
This can be done degree-by-degree.

Soo Go

To combine Ben-Or/ Tiwari with modular algorithms, Soo Go came up with a trick:
Let b_i be a bound on the degree of x_i in f. Let $p = k\prod_{i=1}^{d} p_i + 1$ be a prime where p_i are pairwise coprime, $p_i \geq b_i$ and $k > 0$ suitable. Primes in arithmetic progressions imply k can be found.
Now let $\mathbb{F}_p^* = \langle z \rangle$ for some (arbitrary) generator z. Choosing $\alpha_i = z^{(p-1)/p_i}$ we can recover the exponents m_t from the roots:

Soo Go

$$\alpha_i = z^{(p-1)/p_i}$$

Since z is primitive, $\beta_t = z^{a_t}$ and

$$\beta_t = \prod (z^{(p-1)/p_i})^{m_i} = z^{\sum m_i(p-1)/p_i} = z^{a_t}$$

so

$$\sum m_i(p-1)/p_i \equiv a_t \bmod p-1$$

and

$$\sum m_i(p-1)/p_i \equiv a_t \bmod p_i$$

but $(p-1)/p_i \equiv 0 \bmod p_j$, so m_i can trivially be found!

Rational Interpolation I

Warming up, using linear algebra again: given $y_i = h(\underline{\alpha}_i)$ for $h = f/g$, $f = \sum f_i \underline{x}^{m_i}$ and $g = \sum g_i \underline{x}^{n_i}$, we again get a linear equation:

$$\sum f_i \underline{\alpha}_j^{m_i} = y_j \sum g_i \underline{\alpha}_j^{n_i}$$

if supersets for the support $\{m_i | i\}$ for f and $\{n_i | i\}$ for g are known. The cost is (cubic) in the size of the supersets. Thus, as before, if only degree bounds are used, this is inefficient - unless the problem is really dense.

Note: the solution is not unique - we can normalise the rational function as we want.

Rational II - Recursive, dense

Assume $h(0)$ is defined, then $g(0) \neq 0$ and wlog. $g(0) = 1$.
Let $\alpha_i = (\mu_i, \nu_2, \ldots, \nu_d)$.
Use the univariate rational to get

$$\frac{f_{\underline{\nu}}(x_1)}{g_{\underline{\nu}}(x_1)} = h(x_1, \nu_2, \ldots, \nu_d).$$

Normalise $g_{\underline{\nu}}(0) = 1$, then $g_{\underline{\nu}} = g(x_1, \nu_2, \ldots, \nu_d)$.
This now is a "simple" multivariate polynomial interpolation problem for f and g, to be solved by any means.
Similarly to the hybrid approach for polynomials, we can use this too to find the degree sets for each variable (at cost $\tilde{O}(\sum \deg_{x_i} h)$).

Rational II - Recursive, dense, shift

To achieve $g(0) \neq 0$, we apply the algorithm to $h(\underline{x} + \underline{\beta})$ for any $\underline{\beta}$ s.th. $h(\underline{\beta})$ is defined.
This “shift” destroys the sparsity of h.
Depending on the overall algorithm, the sparsity can be recovered in the polynomial interpolation step.
We need $2 \deg f_{\underline{\nu}}$ evaluation points for $f_{\underline{\nu}}$ and then more for the rest.

Rational III - Sparse

Similar to the Ben Or, Tiwari, Soo Go method, we can operate here.
Assume first $h(0)$ is defined, thus $g(0) \neq 0$. As above, wlog. $g(0) = 1$.
Evaluating at $\alpha_{i,j} = \beta_j(p_1^i, \ldots, p_d^i)$ for i fixed, using the rational univariate case, we find $h_i(z) = f_i(z)/g_i(z)$ and then proceed as in the multivariate polynomial case for f and g separately.
However, if $g(0) = 0$ we cannot do this (directly) and shifting destroys the sparsity.

Rational III - Sparse

Observation: The leading monomial in $f(\underline{x})$ and $f(\underline{x}+\underline{\beta})$ is identical! In fact, the entire homogenous component of highest degree is unchanged.
Thus we can use Ben Or, Tiwari, Soo Go to find the maximal homogenous component H - and then proceed to recover $f(\underline{x}+\underline{\beta}) - H(\underline{x}+\underline{\beta})$. Recursively, we can recover the sparse f and g.
Let D be (a bound for) the largest number of homogenous parts. The costs are $O(4\deg hD)$ evaluation points, and $D\tilde{O}(2\deg h)$ to find all f_i, then $\tilde{O}(2D)$ for each Berlekamp-Massey, $O(D^\omega)$ to find the coefficients as well as the univariate factorisation.

Final Remarks

- Unless bounds/ properties are known, reconstruction is not guaranteed to find the "correct" result
- Methods can be nested: using modular methods to compute rational reconstructions over $\mathbb{Q}$ or $\mathbb{F}_p(\underline{x})$
- Each level in the product trees can be evaluated in parallel
- The lifting can be extended to deal with "wrong" evaluation values, coming from bad primes
- The univariate case can be extended to allow addition of more points - until we are happy with the result.