Functional interpolation techniques

Jonas Klappert

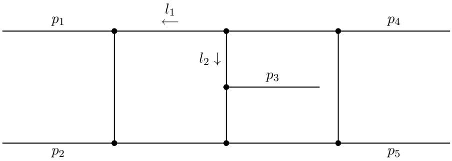
23.06.2020, Young Scientist Forum

Collaborators `FireFly` project: S.Y. Klein, F. Lange
Collaborators `Kira` project: F. Lange, P. Maierhöfer, J. Usovitsch

Problems in state-of-the-art calculations

Problem: Complexity becomes bottleneck for CAS-like approaches

For example: multi-scale problems



- Usually sparse, but

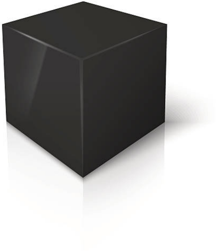
$$\#\text{nonzero terms} \propto \binom{D+n}{n}$$

- $D = 100$ for $n = 4 \rightarrow 4.6 \cdot 10^6$
- $D = 100$ for $n = 5 \rightarrow 9.7 \cdot 10^7$

Runtime and memory consumption can become critical

Black-box interpolation problem

$\vec{y} \longrightarrow$



$\longrightarrow f(\vec{y})$

$f(\vec{x}) = ?$

Black-box interpolation problem

$\vec{y} \longrightarrow$

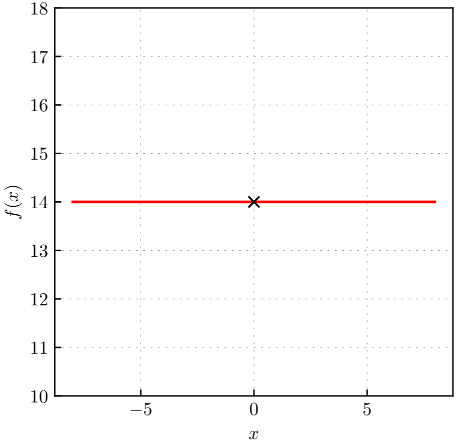
$\longrightarrow f(\vec{y})$

$$f(\vec{x}) = \frac{n_0 + n_1 x_1 + \dots}{d_0 + d_1 x_1 + \dots}$$

FireFly

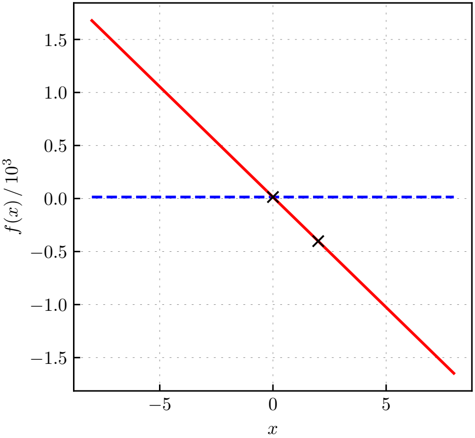
[JK, Lange 2019; JK, Klein, Lange 2020]

Interpolation



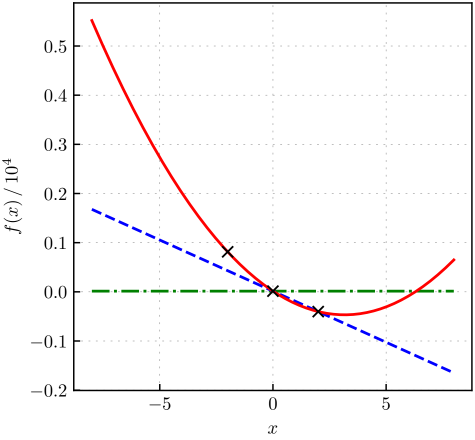
$$f(x) = 14$$

Interpolation



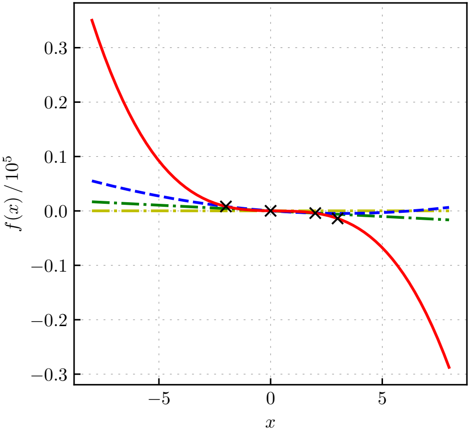
$$f(x) = 14 - 208\,x$$

Interpolation

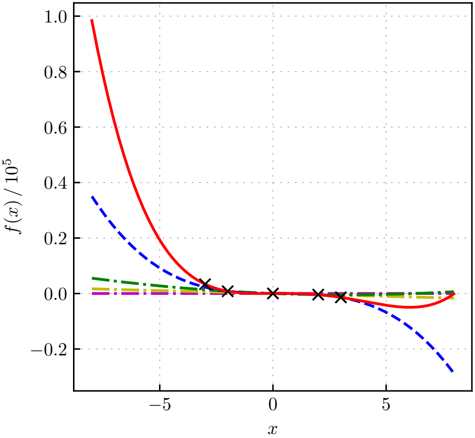


$$f(x) = 14 - 304\,x + 48\,x^2$$

Interpolation

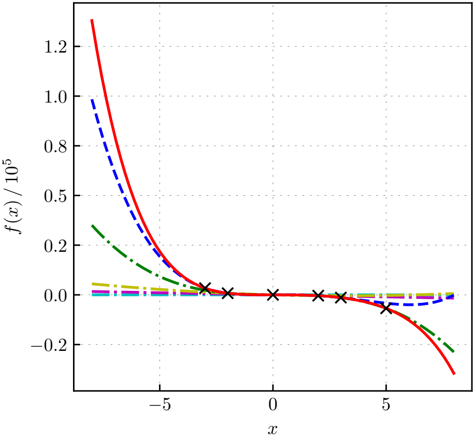


$$f(x) = 14 - 58\,x + 48\,x^2 - 62.5\,x^3$$



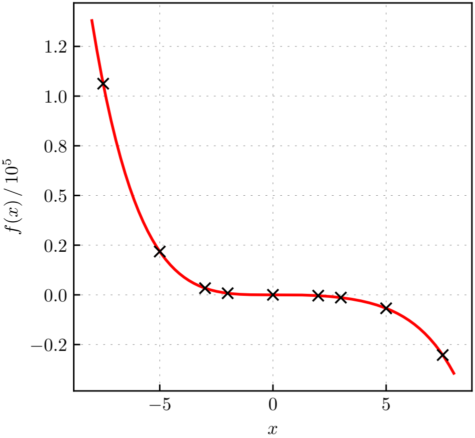
$$f(x) = 14 + 86\,x - 97.5\,x^3 + 12x^4$$

Interpolation



$$f(x) = 14 + 32\,x - 78\,x^3 + 12x^4 - 1.5\,x^5$$

Interpolation



$$f(x) = 14 + 32\,x - 78\,x^3 + 12x^4 - 1.5\,x^5$$

Finite fields

- Definition: A field with a finite number of elements
- All basic arithmetic operations are defined
- We use a prime field $\mathbb{Z}_p$, i.e. integers modulo a prime p
- Example $\mathbb{Z}_7$:

$$
\begin{aligned}
(3+5) \mod 7 &= 1 \\
(3-6) \mod 7 &= 4 \\
(4\cdot 3) \mod 7 &= 5
\end{aligned}
$$

- There is a unique multiplicative inverse for every element in $\mathbb{Z}_p$
- In $\mathbb{Z}_7$ the inverse of 2 is 4:

$$2\cdot 2^{-1} = 1 = 2\cdot 4 \mod 7$$

- Arithmetic is exact and fast (machine-size integer)

Multivariate interpolation over $\mathbb{Z}_p$

- Studied over the last decades in theoretical Computer Science
- Polynomial interpolation [Zippel 1979; Ben-Or,Tiwari 1988; Kaltofen, Lakshman 1988; Kaltofen, Lakshman, Wiley 1990; Kaltofen, Trager 1990; Diaz, Kaltofen 1998; Kaltofen, Lee, Lobo 2000; Kaltofen, Lee 2003; Kaltofen, Yang 2007; Javadi, Monagan 2010]
- Chance of failure (probabilistic) [Zippel 1979; Schwartz 1980; Zippel 1990]

$$\frac{nD^2T^2}{p}, \qquad T \leq \binom{D+n}{n}$$

Multivariate interpolation over $\mathbb{Z}_p$

- Studied over the last decades in theoretical Computer Science
- Polynomial interpolation [Zippel 1979; Ben-Or,Tiwari 1988; Kaltofen, Lakshman 1988; Kaltofen, Lakshman, Wiley 1990; Kaltofen, Trager 1990; Diaz, Kaltofen 1998; Kaltofen, Lee, Lobo 2000; Kaltofen, Lee 2003; Kaltofen, Yang 2007; Javadi, Monagan 2010]
- Chance of failure (probabilistic) [Zippel 1979; Schwartz 1980; Zippel 1990]

$$\frac{nD^2T^2}{p}, \qquad T \leq \binom{D+n}{n}$$

- Rational function interpolation [Grigoriev, Karpinski, Singer 1990; Kaltofen, Trager 1990; Grigoriev, Karpinski 1991; Grigoriev, Karpinski, Singer 1994; Monagan 2004; de Kleine, Monagan, Wittkopf 2005; Kaltofen, Yang 2007; Cuyt, Lee 2011; Huang, Gao 2017; JK, Lange 2019; Peraro 2019; JK, Klein, Lange 2020]

Strategy: Univariate rational function interpolation combined with multivariate polynomial interpolation as a hybrid racer

[JK, Klein, Lange 2020]

Brief history of finite fields in HEP

General ideas:

- Apply Laporta algorithm over $\mathbb{Z}_p$ [Kauers 2008]
- Remove linearly dependent equations from Laporta algorithm [Kant 2013; Maierhöfer, Usovitsch, Uwer 2017]
- Interpolate master integral coefficients [von Manteuffel, Schabinger 2014]
- Interpolate scattering amplitudes [Peraro 2016]

Some exemplary results:

- Four-loop contributions to gluon form factor [von Manteuffel, Schabinger 2017; von Manteuffel, Schabinger 2019]
- Four-loop cusp anomalous dimension [Henn et al. 2019]
- Two-loop five-gluon scattering [Badger et al. 2018; Abreu et al. 2018; Abreu et al. 2018; Badger et al. 2019; Abreu et al. 2019; Abreu et al. 2019; Badger et al. 2019]
- Three-loop gradient flow (for lattice) [Artz et al. 2019] ← Lange
- ...

Application: reduction to master integrals

Many (linear dependent) scalar integrals

$$F(\mathrm{id}; d, \{q_j\}, \{M_i\}, \{a_i\}) = \int_{l_1,\ldots,l_L} \frac{1}{P_1^{a_1} \cdots P_N^{a_N}}$$

with $P_i = k_i^2 - M_i^2 + \mathrm{i}0$

Solution: IBP reduction to a set of linear independent (master) integrals

[Tkachov 1981; Chetyrkin 1981]

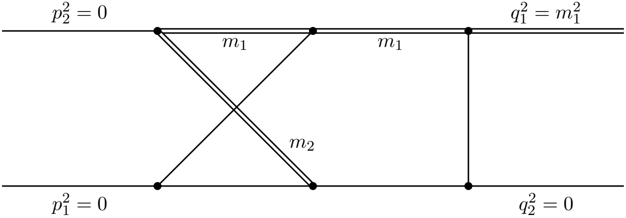
Perform derivative

$$\int_{l_1,\ldots,l_L} \frac{\partial}{\partial l_i^\mu} \left(\tilde{q}_j^\mu \frac{1}{P_1^{a_1} \cdots P_N^{a_N}} \right) = 0$$

to obtain linear relations between integrals

$$0 = \sum_n c_n F(\mathrm{id}; d, \{q_j\}, \{M_i\}, \{a_i^{(n)}\})$$

Example: single top production (4 scales + d)



Reduction of all integrals that could possibly appear in the amplitude

Kira		Kira + FireFly	
Runtime	Memory	Runtime	Memory
92 h	164 GiB	20 h	45 GiB

Used 48 cores on two Intel Xeon Platinum 8160

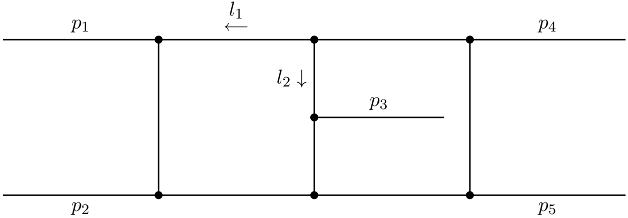
Example: single top production (4 scales + d) with factor scan

- In many physical applications, factors in the given set of variables occur, e.g. $(d-4)$
- Automatized univariate factor scan in `FireFly` before the multivariate interpolation starts
- Use these factors to cancel contributions to the black box

	No factor scan			With factor scan		
s_{max}	Runtime	Memory	Probes	Runtime	Memory	Probes
4	33 h	67 GiB	2745200	20 h	45 GiB	1624200

Used 48 cores on two Intel Xeon Platinum 8160

Example: five-light-parton scattering (5 scales + d) preliminary



Reduction of all integrals that appear in the amplitude

- Use method developed by [Guan, Liu, Ma 2019] → block-triangular system
- Trick: set MIs to zero and perform reduction sectorwise
- Runtime is less than 6 days and memory consumption is below 160 GiB
- Roughly $36 \cdot 10^6$ probes required

Used 40 cores on two Intel Xeon Gold 6138

Summary and Outlook

- Functional interpolation techniques offer competitive alternative to common algebraic approach
- Numerous applications in physical contexts (e.g. `ff_insert`)
- `FireFly` as general purpose interpolation library
- Outlook: Merge of `Kira` and `FireFly` is complete, need to write the paper...

Beta available at (today or in the next few days)

`https://gitlab.com/kira-pyred/kira`

Backup

Shifted Vandermonde system

$$\begin{pmatrix} v_{\alpha_1} & v_{\alpha_2} & \cdots & v_{\alpha_m} \\ v_{\alpha_1}^2 & v_{\alpha_2}^2 & \cdots & v_{\alpha_m}^2 \\ \vdots & \vdots & \ddots & \vdots \\ v_{\alpha_1}^m & v_{\alpha_2}^m & \cdots & v_{\alpha_m}^m \end{pmatrix} \begin{pmatrix} c_{\alpha_1} \\ c_{\alpha_2} \\ \vdots \\ c_{\alpha_m} \end{pmatrix} = \begin{pmatrix} f(\vec{y}^{\,1}) \\ f(\vec{y}^{\,2}) \\ \vdots \\ f(\vec{y}^{\,m}) \end{pmatrix}$$

Scales as $\mathcal{O}(m^2)$ in time and $\mathcal{O}(m)$ in space.

Interpolation of multivariate polynomials I

[Zippel: 1989; Kaltofen, Lee, Lobo: 2000]

- Try to interpolate

$$f(z_1, z_2, z_3) = z_1^5 + z_1 z_2^4 + z_1 z_2 z_3^3 + z_2^5$$

  using the Zippel algorithm with Newton interpolation.

Stage 1: Insert y_2, y_3 for z_2, z_3 and just interpolate z_1 dependence using Newton:

$$f(z_1, y_2, y_3) = k_0 + k_1 \cdot z_1 + k_5 \cdot z_1^5$$

- k_0, k_1, k_5 are multivariate polynomials in z_2, z_3 evaluated at y_2, y_3

Interpolation of multivariate polynomials II

[Zippel: 1989; Kaltofen, Lee, Lobo: 2000]

Stage 2: Interpolate dependence of z_2 using $k_{i,j}$ as input with Newton

$$f_j(z_1, y_2^j, y_3) = k_{0,j} + k_{1,j} \cdot z_1 + k_{5,j} \cdot z_1^5$$

- Assumption: $k_{2,j} = k_{3,j} = k_{4,j} = 0$ for this and all other stages!
- Instead of using Newton for z_1 to get f_j, build system of equations (Vandermonde)

$$\Rightarrow f(z_1, z_2, y_3) = \tilde{k}_0 \cdot z_2^5 + \tilde{k}_1 \cdot z_1 z_2 + \tilde{k}_2 \cdot z_1 z_2^4 + \tilde{k}_5 \cdot z_1^5$$

Proceed with z_3 analogously using $\tilde{k}_i(y_3)$

Benefits: Bound on evaluations of f to fully interpolate the function is $\binom{n+R}{R} \Leftrightarrow$ all possible coefficients, Vandermonde systems $O(m^2)$ in time

Interpolating rational functions

- Much more involved than polynomial interpolation (normalization issues) but doable [Cuyt, Lee: 2011]

$$f(\vec{z}) = \frac{3z_1 + 7z_2}{z_1 + z_2 + 4z_1 z_2}$$

- Idea: map multivariate function to dense univariate function by homogenizing with a new variable and perform a variable shift to normalize
- Note: Coefficients are multivariate polynomials in $\vec{z}$! Use previous strategies to get their analytical form!

$$f\left(t\vec{y} + \vec{s}\right) = \frac{316 + 464t}{1 + 178t + 317t^2}$$

Interpolating rational functions - formulae

How to get $f\,(t\vec{y} + \vec{s})$?

- Thiele interpolation (no knowledge about f needed)

$$T(t) = b_0 + (t - t_1)\left(b_1 + (t - t_2)\left(b_2 + (t - t_3)\left(\cdots + \frac{t - t_r}{b_r}\right)^{-1}\right)^{-1}\right)^{-1}$$

- Better: System of equations (degree bound needed)

$$f(z) = \frac{\sum_{i=0}^{R} n_i(z)}{1 + \sum_{j=1}^{R'} d_j(z)} \quad \Rightarrow \quad \sum_{i=0}^{R} n_i(z) - f(z) \sum_{j=1}^{R'} d_j(z) = f(z)$$

Extended Euclidean Algorithm

Q: How to find general solution of

$$a\,s + b\,t = \gcd(a, b)\,,$$

where a, b is input?

A: Extended Euclidean Algorithm (also used for inverse in finite field)

```
function extended_gcd(a, b)
    s := 0;    old_s := 1
    t := 1;    old_t := 0
    r := b;    old_r := a

    while r != 0
        quotient := old_r / r
        (old_r, r) := (r, old_r - quotient * r)
        (old_s, s) := (s, old_s - quotient * s)
        (old_t, t) := (t, old_t - quotient * t)

    output "Bezout coefficients:", (old_s, old_t)
    output "greatest common divisor:", old_r
```

From finite fields back to rational numbers I

Rational Reconstruction algorithm [Wang 1981]:

- Input: Integers $p > a \geq 0$
- Output: Rational number $e \equiv r/t$ such that

$$a = e \bmod p \quad \text{or} \quad \texttt{FAIL}$$

- Succeeds if $|r|, |t| \leq \sqrt{p/2}$
- e is unique [Wang, Guy, Davenport 1982]

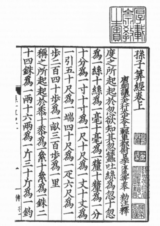
Alternative: Maximal Quotient Rational Reconstruction [Monagan 2004]:

- MQRR performs better in the average case but can perform worse
- Race Wang and Monagan [JK, Lange 2019]

What if p is too small to get e?

From finite fields back to rational numbers II

Q: Can one reuse images in multiple $\mathbb{Z}_{p_i}$,



$$a_1 = e \mod p_1\,,$$
$$a_2 = e \mod p_2\,,$$

of a rational number e?

A: Yes, Chinese Remainder Theorem

- Input: Two pairs of integers a_i and p_i such that

$$a_i = e \mod p_i$$

- Output: New pair a and $p = p_1 \cdot p_2$ such that

$$a = e \mod p$$

- Allows to combine the results of the interpolation over several $\mathbb{Z}_{p_i}$
- Interpolate over $\mathbb{Z}_{p_i}$ until the rational reconstruction succeeds