Integrand Reduction Reloaded
Algebraic Geometry and Finite Fields

Ray D. Sameshima

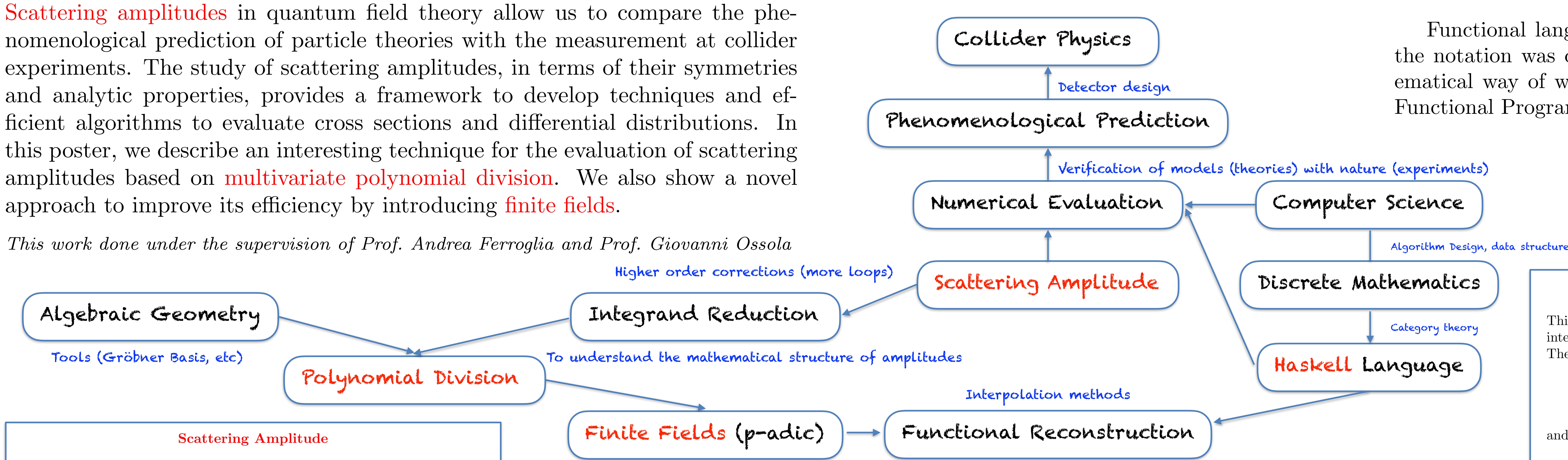
Department of Physics – New York City College of Technology

Graduate School and University Center, City University of New York

ABSTRACT

Scattering amplitudes in quantum field theory allow us to compare the phenomenological prediction of particle theories with the measurement at collider experiments. The study of scattering amplitudes, in terms of their symmetries and analytic properties, provides a framework to develop techniques and efficient algorithms to evaluate cross sections and differential distributions. In this poster, we describe an interesting technique for the evaluation of scattering amplitudes based on multivariate polynomial division. We also show a novel approach to improve its efficiency by introducing finite fields.

This work done under the supervision of Prof. Andrea Ferroglia and Prof. Giovanni Ossola



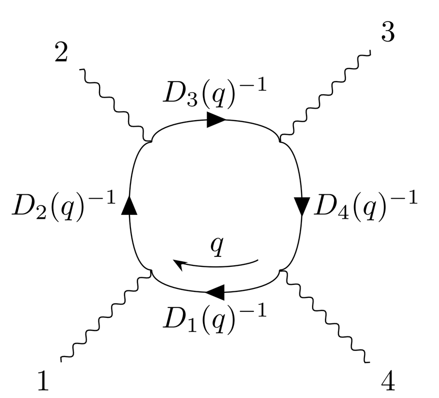
Functional languages can be seen as 'executable mathematics'; the notation was designed to be as close as possible to the mathematical way of writing. [F. Rabhi, G. Lapalme, "Algorithms: A Functional Programming Approach" 1999]

```
Haskell :: Pure, Lazy, Functional Programming language
-- "a monad is a monoid object in the category of endofunctors, what's the problem?"
```

Scattering Amplitude

A scattering amplitude is given by Feynman diagrams, for example, one-loop $n = 4$ diagram is

$$iI_4(q) \quad :=$$



$$= \quad \frac{N_4(q)}{D_1(q)D_2(q)D_3(a)D_4(q)}.$$

Integrand Reduction

Our target, n-leg l-loop integrand has the following form of rational function

$$\begin{aligned} I &= \frac{N(q)}{D_1(q)\cdots D_n(q)} \\ q &= (q_1, \cdots, q_l) \end{aligned}$$

where $D_i(q)$ is a propagator and $N(q)$ is a polynomial of loop momenta q's.

Introducing the ideal generated by denominators, we recursively take polynomial division of numerators

$$\begin{aligned} N(q) &= \Gamma(q) + \Delta(q) \\ \Gamma(q) &\in \langle D_1(q), \cdots D_n(q)\rangle \end{aligned}$$

modulo a Gröbner basis [B. Buchberger 1985] of the ring $\langle D_1(q), \cdots D_n(q)\rangle$ with some fixed monomial order.

For $l = 1$ loop case, we have the completely decomposed expression [P. Mastrolia, E. Mirabella, G. Ossola, T. Peraro]

$$I = \sum_{i=1}^{5}\left(\sum_{1=i_1<\cdots<i_k}^{n-1} \frac{\Delta_{i_1\cdots i_k}}{D_{i_1}\cdots D_{i_k}}\right).$$

Finite Fields $\mathbb{Z}_p$

Let p be a prime number. $\mathbb{Z}_p$ essentially is the quotient ring with modulo arithmetic. The underlying set is just a finite set of numbers:

$$\{0, 1\cdots, (p-1)\}$$

With Chinese Remainder Theorem, we can fully reconstruct any rational number from its images in $\mathbb{Z}_p$ with high probability in the machine size integer

```
[-2^63,2^63-1]
```

Modulo arithmetics

For arbitrary rational number

$$q = \frac{n}{d}$$

of $n \in \mathbb{Z}, d(>0) \in \mathbb{Z}$, we map it to $\mathbb{Z}_p$ by

$$\frac{n}{d} \mod p := (n * (d^{-1} \mod p)) \mod p.$$

We call $q \mod p$ the image of $q \in \mathbb{Q}$ in $\mathbb{Z}_p$.

We can determine the original rational number q from its image in $\mathbb{Z}_p$ uniquely if $n^2, d^2 \lessapprox p$ [P. S. Wang 1981].

Functional reconstruction

Let us specify our problem. Our target is the numerator function of the integrand which is given as a "black box". That is, what we can access is a graph of its input-output.

$$q \overset{N}{\mapsto} N(q)$$

For univariate polynomial reconstruction, we can apply Newton interpolation method to determine the coefficients.

$$N(q) = \sum_\alpha c_\alpha q^\alpha$$

The base field is usually the rational field $\mathbb{Q}$. Here we combine these reconstruction techniques with finite fields $\mathbb{Z}_p$ [A. Cuyt, W. S. Lee, 2011, T. Peraro, 2016].

Haskell implementation,
given by a function composition

```
> list2pol :: (Integral a) => [Ratio a] -> [Ratio a]
> list2pol = npol2pol . list2npol
```

Several checks on an interpreter (GHCi)

```
Reconstruction as curve fitting
  *Univariate> let f x = 2*x^3 + 3*x + 1%5
  *Univariate> take 10 $ map f [0..]
  [1 % 5,26 % 5,111 % 5,316 % 5,701 % 5,1326 % 5,2251 % 5,3536 % 5,5241 % 5,7426 % 5]
  *Univariate> list2npol it
  [1 % 5,5 % 1,6 % 1,2 % 1]
  *Univariate> list2npol $ map f [0..]
  [1 % 5,5 % 1,6 % 1,2 % 1]
  *Univariate> list2pol $ map (\n -> 1%3 + (3%5)*n + (5%7)*n^2) [0..]
  [1 % 3,3 % 5,5 % 7]
  *Univariate>  list2pol [0,1,5,14,30,55]
  [0 % 1,1 % 6,1 % 2,1 % 3]
  *Univariate> map (p2fct $ list2pol [0,1,5,14,30,55]) [0..6]
  [0 % 1,1 % 1,5 % 1,14 % 1,30 % 1,55 % 1,91 % 1]
```

Extended Euclidean algorithm

The function `exGCD` takes two integer a, b and returns their gcd and two integers x, y which satisfy so called Bëzout identity:

$$a * x + b * y = \gcd(a, b).$$

Its type annotation in Haskell language is the following.

```
> exGCD :: Integral t => t -> t -> (t, t, t)
> exGCD a b = (g, x, y)
>   where ...
```

For arbitrary $a \in \mathbb{Z}_p$ of a prime field, we have

$$\begin{aligned} a * x + p * y &= \gcd(a, p) \\ &= 1 \end{aligned}$$

Once we take modulo p of both sides, we have

$$a * x = 1 \mod p,$$

and

$$x = a^{-1} \mod p.$$

Chinese Remainder Theorem

Let $p_1, p_2 \in \mathbb{Z}$ be coprime integers, then for arbitrary $a_1, a_2 \in \mathbb{Z}$, the following a system of equations

$$\begin{aligned} x &= a_1 \mod p_1 \\ x &= a_2 \mod p_2 \end{aligned}$$

have a unique solution modular $p_1 * p_2$. That is, we can enlarge the unique region for reconstruction.

$$n^2, d^2 \lessapprox p \Rightarrow n^2, d^2 \lessapprox (p_1 * p_2 * \cdots)$$

Taking several big primes, we can determine any rational number given by two machine size `Int` uniquely.

The solution
for Chinese Remainder Theorem

Its implementation

From Bëzout identity, there are $m_1, m_2 \in \mathbb{Z}$ s.t.

$$p_1 * m_1 + p_2 * m_2 = 1.$$

Now we have

$$\begin{aligned} m_1 &= p_1^{-1} \mod p_2 \\ m_2 &= p_2^{-1} \mod p_1. \end{aligned}$$

Then the solution is given by

$$a := a_1 * p_2 * m_2 + a_2 * p_1 * m_1 \mod (p_1 * p_2).$$

```
> crtRec' :: Integral t => (t, t) -> (t, t) -> (t, t)
> crtRec' (a1,p1) (a2,p2) = (a,p)
>   where
>     a = (a1*p2*m2 + a2*p1*m1) `mod` p
>     m1 = fromJust (p1 `inversep` p2)
>     m2 = fromJust (p2 `inversep` p1)
>     p = p1*p2
```

Extended Euclidean Algorithm

This algorithm is a constructive proof for Bëzout identity. The inputs are two integers a, b and the out puts are $\gcd(a, b)$ and four lists $\{q_i\}_i, \{r_i\}_i, \{s_i\}_i, \{t_i\}_i$. The base cases are

$$\begin{aligned} (r_0, s_0, t_0) &:= (a, 1, 0) \\ (r_1, s_1, t_1) &:= (b, 0, 1) \end{aligned}$$

and inductively, for $i \geq 2$,

$$\begin{aligned} q_i &:= \mathrm{quot}(r_{i-2}, r_{i-1}) \\ r_i &:= r_{i-2} - q_i * r_{i-1} \\ s_i &:= s_{i-2} - q_i * s_{i-1} \\ t_i &:= t_{i-2} - q_i * t_{i-1}. \end{aligned}$$

The termination condition is

$$r_k = 0$$

for some $k \in \mathbb{N}$ and

$$\begin{aligned} \gcd(a, b) &= r_{k-1} \\ x &= s_{k-1} \\ y &= t_{k-1}. \end{aligned}$$

```
> exGCD' :: (Integral n) => n -> n -> ([n], [n], [n], [n])
> exGCD' a b = (qs, rs, ss, ts)
>   where
>     qs = zipWith quot rs (tail rs)
>     rs = takeUntil (==0) r'
>     r' = steps a b
>     ss = steps 1 0
>     ts = steps 0 1
>     steps a b = rr
>       where
>         rr@(_:rs) = a:b: zipWith (-) rr (zipWith (*) qs rs)
>
> takeUntil :: (a -> Bool) -> [a] -> [a]
> takeUntil p = foldr func []
>   where
>     func x xs
>       | p x = []
>       | otherwise = x : xs
```

We describe "what"
in functional language

References

1. Pierpaolo Mastrolia, Edoardo Mirabella, Giovanni Ossola, Tiziano Peraro, "Scattering Amplitudes from Multivariate Polynomial Division", arXiv:1205.7087 [hep-ph]
2. Tiziano Peraro, "Scattering amplitudes over finite fields and multivariate functional reconstruction", arXiv:1608.01902 [hep-ph]
3. Paul S. Wang, "A p-adic algorithm for univariate partial fractions", Proceeding SYMSAC '81 Proceedings of the fourth ACM symposium on Symbolic and algebraic computation (1981)
4. Bruno Buchberger, "Grobner bases: An algorithmic method in polynomial ideal theory" Multidimensional systems theory (1985)
5. Annie Cuyt, Wen-shin Lee, "Sparse interpolation of multivariate rational functions" Theoretical Computer Science (2011)

This project has been supported by the **National Science Foundation** under **Grant PHY-1417354**, and performed in collaboration with Pierpaolo Mastrolia, William J. Torres Bobadilla, and Amedeo Primo from Universita' degli Studi di Padova.

Thoughts, Suggestions, and Comments