

Solving the computational bottlenecks of perturbative QFT

Ben Ruijl

ETH Zurich

January 26, 2021

About me

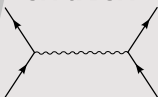
- Theoretical physicist specialised in computational problems
- PhD at Nikhef & Leiden University, 2017, *cum laude*
 - Search algorithms, AI
 - Simplifications of huge algebraic expressions
 - New methods: FORM, Forcer, R^*
 - Five-loop QCD beta function and the Higgs decay width
- PostDoc at ETH:
 - reFORM: next-generation computer algebra
 - Havana: new Monte Carlo integrator
 - Loop-Tree Duality, Local Unitarity
 - Project co-leader, mentoring four PhD students

Why better predictions?

Towards a better understanding of the Standard Model and Beyond

THEORY

SM & BSM



Predictions

- Differential cross sections

EXPERIMENT

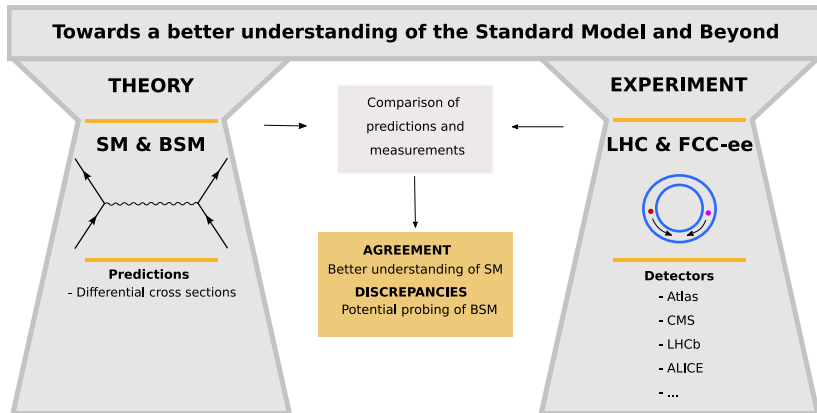
LHC & FCC-ee



Detectors

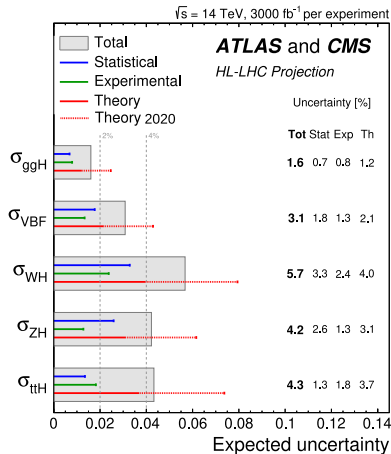
- Atlas
- CMS
- LHCb
- ALICE
- ...

Why better predictions?



Keeping up with experiment

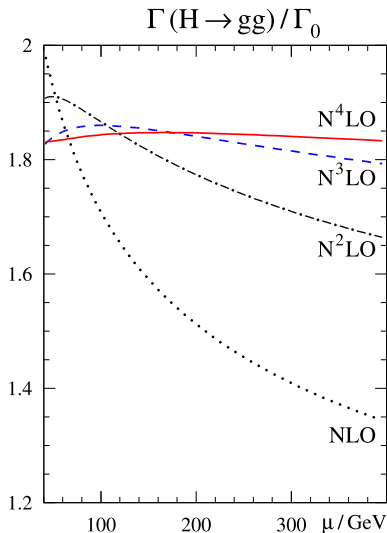
- High Luminosity LHC is expected to drastically reduce statistical uncertainty
- Theory uncertainty will be the **dominating contribution**
- We need to compute more higher-order corrections



[CERN-LPCC-2018-04]

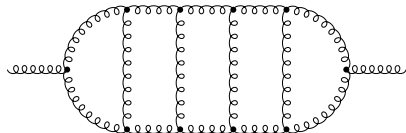
The effect of higher-order corrections

- Large reduction of uncertainty
- **Red line** is an *NNNNLO* computation [Herzog, Ruijl, Ueda, Vermaseren, Vogt, JHEP 2017]
- NLO cross sections are available with the push of a button
- Why no automation at NNLO?




Computational blow-up

- Each Feynman diagram represents a complicated integral



- Represents 12 029 521 scalar integrals!
- Can be reduced to 23 master integrals with algebraic identities
[\[Ruijl, Ueda, Vermaseren, CPC 2017\]](#)
- This reduction requires a terabyte of disk space and is time-consuming

FORM and symbolic manipulation

- Form is used in almost every cutting-edge QCD computation for 30 years [FORM 4.2 [Ruijl, Ueda, Vermaseren, '19](#)]
- Versatile: e.g., check if the graph  is connected:

```
1 L F = v(e1,e3,e4)*v(e1,e2,e5)*v(e2,e3,e6)*v(e4,e5,e6);  
2  
3 repeat id v(?a,e?,?b)*v(?c,e?,?d) = v(?a,?b,?c,?d);  
4 if count(v) <= 1 { print "Connected"; }
```


reFORM

reFORM: a new computer algebra system

- State-of-the-art gcd and factorization algorithms [\[Ruijl '19\]](#)
- Improved pattern matching algorithm
- Improved parallel sorting
- Improved parallel performance
- First-class finite field support
- Term and graph canonicalization algorithms
- No memory bugs, even when multi-threading (guaranteed!)

reFORM's Python API

```
1 import reform
2 vi = reform.VarInfo()
3 a = reform.Expression("x+x^2+3*x^4", vi)
4 expr = a * reform.Expression("x^2+5", vi)
5
6 expr = expr.expand().id("x^n?", "n? * x^(n?-1)", vi)
7
8 print("Result: ", expr)
```

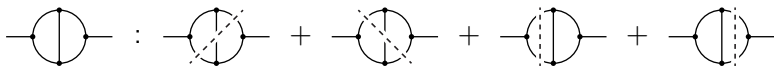
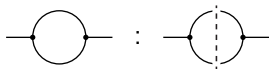
Listing: Polynomial derivative

The major bottlenecks of higher-order corrections

- The main challenge is that amplitudes are divergent
- It takes years to compute a single process!
- The methods used are hard to generalize

A paradigm shift

- Do not consider the amplitude as the fundamental component anymore, but elements of the cross section with the same *supergraph*: [Capatti, Hirschi, Kermanschah, Pelloni, Ruijl, JHEP 2020]



- Each of these collections can be made *locally* finite
- They can then be Monte Carlo integrated

Local Unitarity

Wishes

Differential cross sections

No restrictions on mass scales

No reduction to master integrals

No complicated function evaluations

No dimensional regularisation

No IR counterterms for loop degrees of freedom

No IR counterterms for real degrees of freedom

Integration in lower dimensions

Contour deformation only when needed

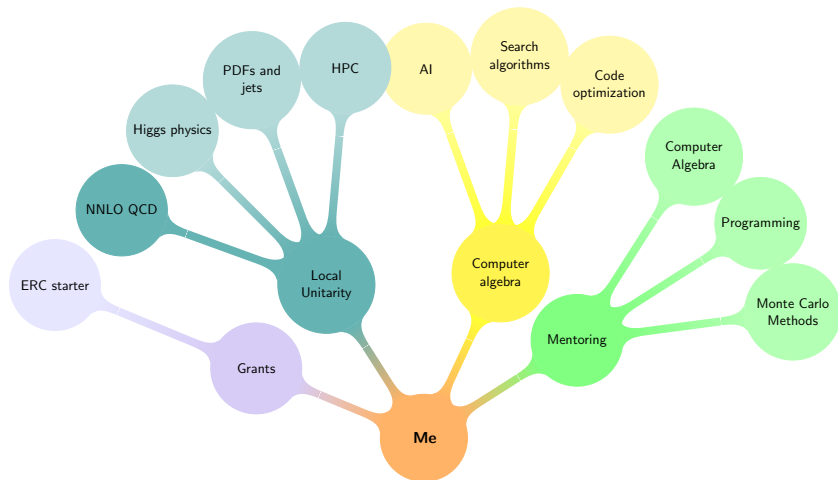
Method generic for any process to any order

Local Unitarity

Local Unitarity

Wishes	Local Unitarity
Differential cross sections	✓
No restrictions on mass scales	✓
No reduction to master integrals	✓
No complicated function evaluations	✓
No dimensional regularisation	✓
No IR counterterms for loop degrees of freedom	✓
No IR counterterms for real degrees of freedom	✓
Integration in lower dimensions	✓
Contour deformation only when needed	✓
Method generic for any process to any order	✓

Opportunities at MPI



Programming resumé (I)

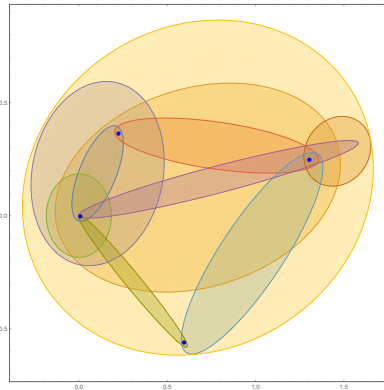
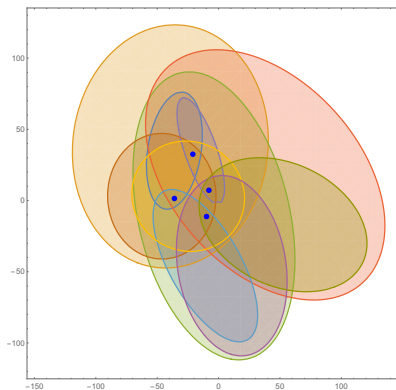
- 3 years Delphi (Pascal) experience
- 2 years Java (gamedev, cryptography)
- 1 year Intel assembly (wrote own OS)
- 6 years C/C++ (wrote award-winning game)
- 1 year Haskell (wrote own programming language)
- 15 years Python experience (solved > 150 project Euler problems)
- 6 years of FORM (computer algebra + dev)
- 5 years of Rust experience

Programming resumé (II)

- Taught programming for 3 years at Leiden University
- Lectured on symbolic manipulation at PhD School CALC2018
- Specialized in writing fast software
- Multi-threaded programming
- Async programming

Solving complicated combinatorial problems

- Find points in maximally overlapping sets of ellipsoids
- Code takes less than 1ms for cases with 40 ellipsoids!



Expression simplification

- Simplification of huge polynomials using Monte Carlo Tree Search, Simulated Annealing [[Ruijl, Vermaseren, Plaat, van den Herik '13](#)]
- Find the optimal Horner scheme and remove common subexpressions
- > 20 times reduction of real-world polynomials
- Essential for one-loop tools such as goSAM

Word-class in computing polynomial GCDs

	REFORM	FORM	Rings	Mathematica	Fermat
D	58.27	82.87	86.77	> 9000	1241.82
D_{simple}	0.31	0.20	0.13	1.39	0.03
$R(2, 50, 1000, 50)$	0.32	0.88	0.91	0.25	0.19
$R(5, 30, 50, 50)$	0.17	0.80	0.20	1.98	10.60
$R(5, 30, 100, 50)$	0.54	0.81	1.02	11.50	22.65
$R(10, 10, 100, 50)$	0.83	0.85	3.12	9.62	37.12

Table: Time in seconds to compute GCD of two large polynomials [Ruijl ACAT 2019]

Havana: Monte Carlo integrator

- Nesting of discrete and continuous layers, e.g.:

$$I = \int dx dy l_1(x, y) + \int dx l_1(x) + \int dx dy dz l_3(x, y, z)$$

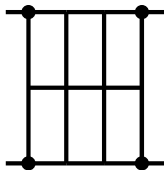
- First discrete sample I_a with probability $\bar{l}_a / (\bar{l}_1 + \bar{l}_2 + \bar{l}_3)$
- Next do VEGAS on the integration variables of I_a
- Very fast integration over multiple Feynman diagrams and multi-channels

Five-loop computations

- Five-loop QCD beta function in 5 days on 1 pc (1.5 years on cluster for competitors) [Herzog, [Ruijl](#), Ueda, Vermaseren, Vogt JHEP '17]
- Five-loop Higgs decay to gluons in two months [Herzog, [Ruijl](#), Ueda, Vermaseren, Vogt JHEP '17]
- Uses Forcer and R^* code written by me
- Billions of terms had to be processed

Pushing the boundaries of numerical integration

6-loop four-point function:



- Analytic: $8.4044862640909 \cdot 10^{-19}$ [Basso, Dixon 2017]
- Numerical: $8.38533 \cdot 10^{-19} \pm 2.99674 \cdot 10^{-21}$ [Hirschi, Ruijl '19]
- Rust code with numerical instability detection, upgrading to f128, dual numbers

Thank you for your attention.