ACAT 2019, Saas Fee, Switzerland 12 March 2019

GPU Usage in Theoretical Calculations

Stephen Jones

Borowka, Greiner, Heinrich, Jahn, Kerner, Luisoni,
Schlenk, Schubert, Zirke

CPC (Accepted) [1811.11720]
PRL 120 (2018) 162001 [1802.00349]
CPC 22 (2018) 313 [1703.09692]
JHEP 10 (2016) 107 [1608.04798]
PRL 117 (2016) 012001, Erratum 079901 [1604.06447]

Higher Order Computations

Focus of this talk will be on the computation of higher order perturbative corrections to scattering processes

QCD Factorisation Formula

$$\mathrm{d}\sigma = \int \mathrm{d}x_a \mathrm{d}x_b f(x_a) f(x_b) \mathrm{d}\hat{\sigma}_{ab}(x_a, x_b) F_J + \mathcal{O}\left((\Lambda/Q)^m\right)$$

PDFs/ Input parameters

Hard Scattering Matrix Element

Non-perturbative effects ~ few %

With $\alpha_s \sim 0.1$

Typically: NLO ~ 10% correction, NNLO ~ 1% correction

However, there are important exceptions:

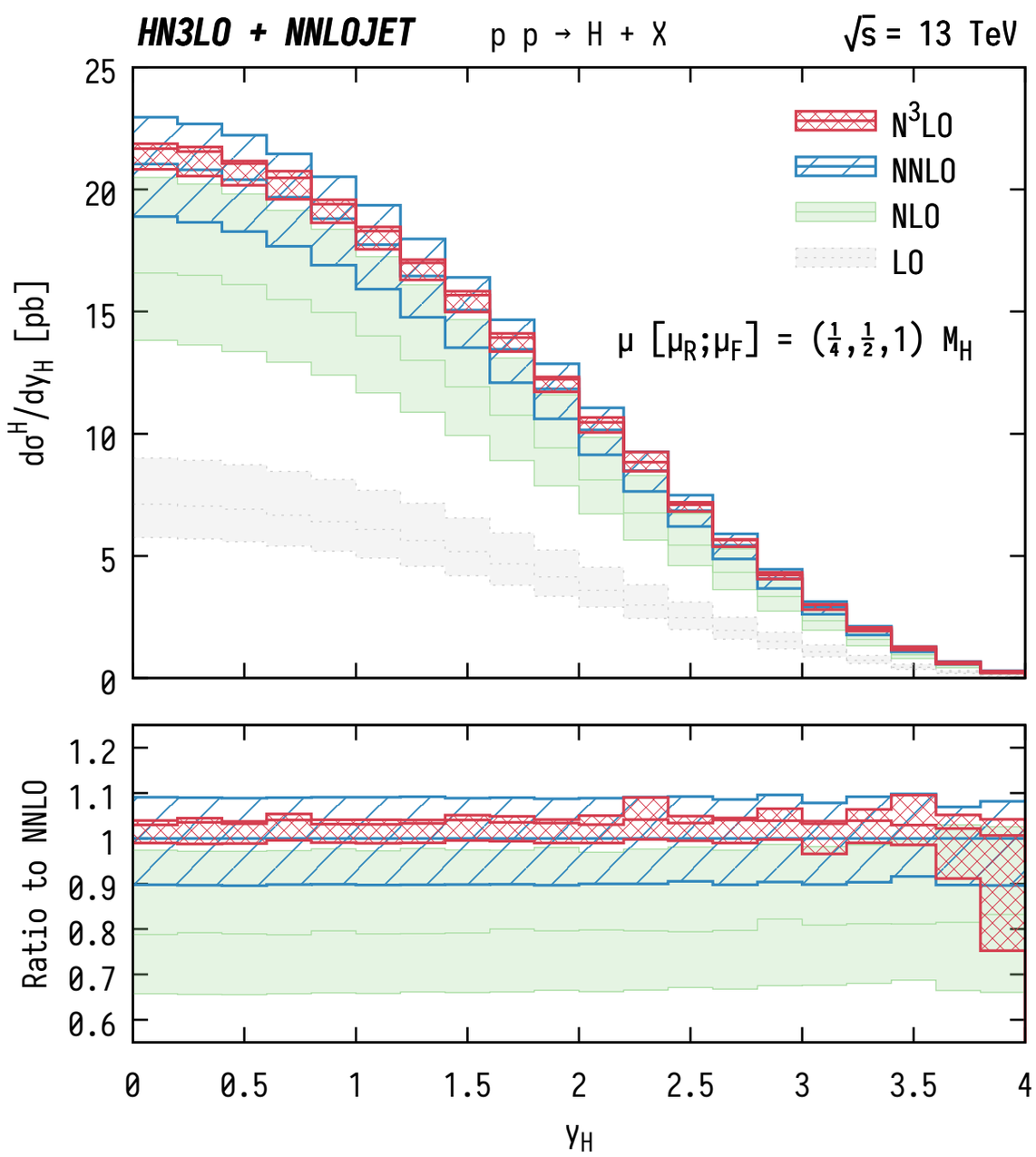
- Higgs Boson production (NLO ~100%, NNLO ~10%, N3LO ~ 2%)
- New partonic channels can open (e.g. di-boson production)

and

- Distributions can be modified substantially (even if σ_{tot} is stable)

HPC for HEP-PH

Some need for HPC (CPUs) for higher order calculations



Cieri, Chen, Gehrmann, Glover, Huss 18

Event generation:

LO		$\mathcal{O}(1\ \mathrm{CPU\ min})$
NLO	[Antenna]	$\mathcal{O}(30\ \mathrm{CPU\ min})$
NNLO	[Antenna]	$\mathcal{O}(100\ \mathrm{CPU\ h})$
N³LO	[q_T sub.]	$\mathcal{O}(7\mathrm{M\ CPU\ h})$

A. Huss, Physics Event Generator Workshop 2018

Most time usually spent integrating real-type contributions

Non-local subtraction schemes can lead to large numerical cancellations (slow)

Outline

(Multi) Loop amplitudes

Numerically Computing Feynman Integrals

- Sector Decomposition
- Numerical Integration
- Rank 1 Shifted Lattices (Quasi-Monte Carlo)

Using GPUs to Compute Feynman Integrals

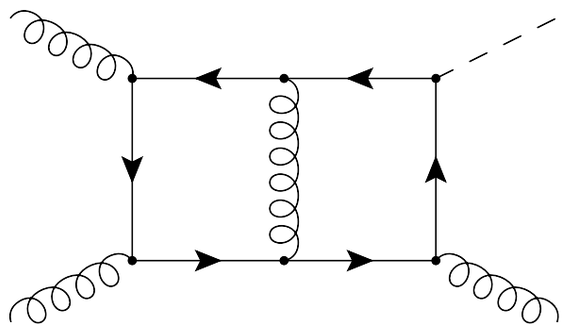
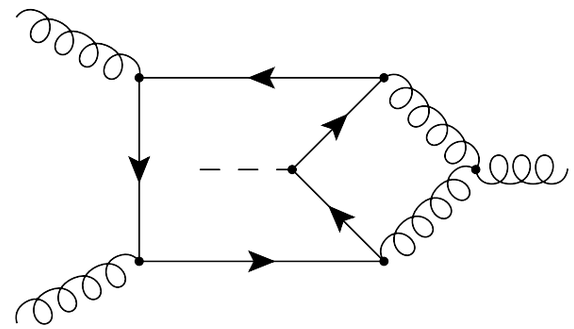
- Integration Package: qmc
- All-in-one Program: pySecDec

Physics Applications

- Higgs Boson Pair Production
- Higgs Boson + Jet Production

Multi-loop Amplitudes

Let us consider (multi-)loop amplitudes themselves:

$\mathcal{M}^{\mu\nu\rho} \sim$  $+$  $+\ldots$

Feynman diagrams → Projectors ($F = P_{\mu\nu\rho}\mathcal{M}^{\mu\nu\rho}$) → Integral Reduction

(or other methods e.g. Numerical unitarity)

$$F = \sum_i c_i I_i$$

Large (#terms/degree) coefficient
Rational function of kinematics
Handled with specialist symbolic manipulation programs

→Talks of Poslavsky, Ueda, Ruijl

Feynman integrals
Sometimes can be computed analytically ($\Longrightarrow$ fast)
We will discuss recent success in computing them numerically

Feynman Integrals

Applying Feynman rules produces integrals in momentum space:

$$I = \int \prod_{l=1}^{L} \left[\mathrm{d}^D k_l\right] \frac{1}{\prod_{j=1}^{N} P_j^{\nu_j}}$$

L – loops
N – propagators
D – spacetime dimension

(Textbook) Feynman Parametrise and compute momentum integrals

$$I = (-1)^{N_\nu} \frac{\Gamma(N_\nu - LD/2)}{\prod_{j=1}^{N} \Gamma(\nu_j)} \int_0^\infty \prod_{j=1}^{N} \mathrm{d}x_j \; x_j^{\nu_j - 1} \delta(1 - \sum_{i=1}^{N} x_i) \frac{\mathcal{U}^{N_\nu - (L+1)D/2}(\vec{x})}{\mathcal{F}^{N_\nu - LD/2}(\vec{x}, s_{ij})}$$

Here $\mathcal{U}, \mathcal{F}$ are simply polynomials

Have exchanged L D-dim. momentum integrals for N param. integrals

Case:

1) Integral I is finite as $D \to 4$, may compute numerically*

2) I is ill-defined as $D \to 4$, regulate $D = 4 - 2\epsilon$, disentangle singular behaviour and expand integral about $\epsilon \to 0$

Sector Decomposition

One technique **Iterated Sector Decomposition** repeat:

Binoth, Heinrich 00

$$\int_0^1 \mathrm{d}x_1 \int_0^1 \mathrm{d}x_2 \frac{1}{(x_1+x_2)^{2+\epsilon}}$$ ← **Overlapping singularity for** $x_1, x_2 \to 0$

$$= \int_0^1 \mathrm{d}x_1 \int_0^1 \mathrm{d}x_2 \frac{1}{(x_1+x_2)^{2+\epsilon}} (\theta(x_1 - x_2) + \theta(x_2 - x_1))$$

$$= \int_0^1 \mathrm{d}x_1 \int_0^{x_1} \mathrm{d}x_2 \frac{1}{(x_1+x_2)^{2+\epsilon}} + \int_0^1 \mathrm{d}x_2 \int_0^{x_2} \mathrm{d}x_1 \frac{1}{(x_1+x_2)^{2+\epsilon}}$$

$$= \int_0^1 \mathrm{d}x_1 \int_0^1 \mathrm{d}t_2 \frac{x_1}{(x_1+x_1 t_2)^{2+\epsilon}} + \int_0^1 \mathrm{d}x_2 \int_0^1 \mathrm{d}t_1 \frac{x_2}{(x_2 t_1+x_2)^{2+\epsilon}}$$

$$= \int_0^1 \mathrm{d}x_1 \int_0^1 \mathrm{d}t_2 \frac{x_1^{-1-\epsilon}}{(1+t_2)^{2+\epsilon}} + \int_0^1 \mathrm{d}x_2 \int_0^1 \mathrm{d}t_1 \frac{x_2^{-1-\epsilon}}{(t_1+1)^{2+\epsilon}}$$ ← **Singularities factorised**

If this procedure terminates depends on order of decomposition steps

An alternative strategy **Geometric Sector Decomposition** always terminates; both strategies are implemented in pySecDec.

Kaneko, Ueda 10; See also: Bogner, Weinzierl 08; Smirnov, Tentyukov 09

Sector Decomposition (II)

3) Expand in ϵ (simple case $a = -1$ "Logarithmic Divergence"):

$$\int_0^1 \mathrm{d}x^{-1-b\epsilon} g(x) = \frac{g(0)}{-b\epsilon} + \int_0^1 \mathrm{d}x x^{-b\epsilon} \left[\frac{g(x) - g(0)}{x} \right]$$

Finite

Poles

`subtraction' of $g(0)$

By Definition: $g(0) \neq 0, g(0)$ finite

4) Numerically integrate

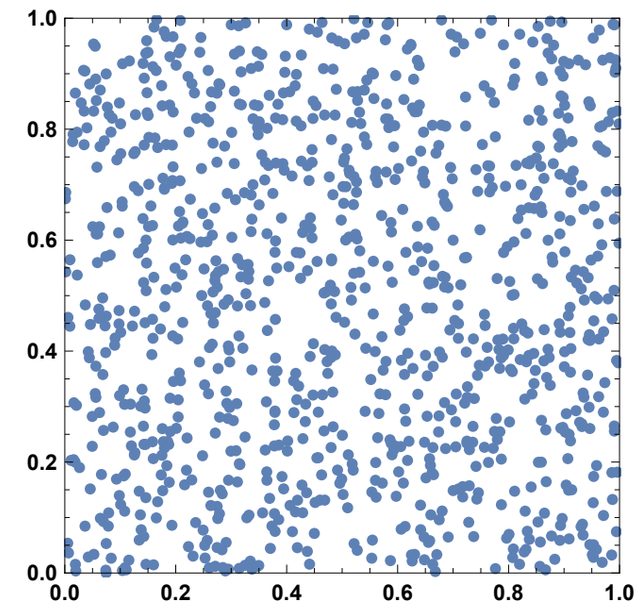
Key Point: Sector Decomposed integrals can be expanded in ϵ and numerically integrated

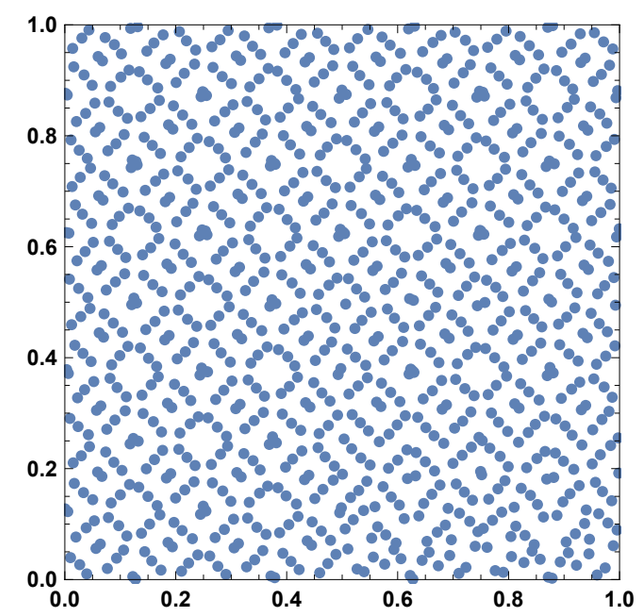
Numerical Integration

$$I[f] \equiv \int_{[0,1]^d} \mathrm{d}\mathbf{x}\ f(\mathbf{x}) \quad \approx \quad Q[f] = \frac{1}{N}\sum_{i=1}^{N} w_i\ f(\mathbf{x}_i)$$

Goal: select points to minimise integration error

$$\varepsilon \equiv |I[f] - Q[f]|$$





Monte Carlo:

Randomly select sampling points

$$\varepsilon \approx \mathrm{Var}[f]/\sqrt{N}, \quad \varepsilon \sim \mathcal{O}(N^{-1/2})$$

Improves slowly with N

Quasi-Monte Carlo

Select points with low discrepancy D_N

$$\varepsilon \leq D_N \cdot \mathrm{V}[f], \quad \varepsilon \sim \mathcal{O}(\log^d(N)/N)$$

Poor performance for large d

Both methods implemented in Cuba Hahn 04; Hahn14

Quasi-Monte Carlo (Rank 1 Lattices)

Quasi-Monte Carlo (QMC) in a Weighted Function Space

First applications to loop integrals, see: Li, Wang, Yan, Zhao 15; de Doncker, Almulihi, Yuasa 17, 18; de Doncker, Almulihi 17; Kato, de Doncker, Ishikawa, Yuasa 18

$\varepsilon \leq e_{\boldsymbol{\gamma}} \cdot ||f||_{\boldsymbol{\gamma}}, \quad \varepsilon \sim \mathcal{O}(N^{-1})$ or better

$$I[f] \approx \bar{Q}_{n,m}[f] \equiv \frac{1}{m} \sum_{k=0}^{m-1} Q_n^{(k)}[f], \quad Q_n^{(k)}[f] \equiv \frac{1}{n} \sum_{i=0}^{n-1} f\left(\left\{\frac{i\mathbf{z}}{n} + \boldsymbol{\Delta}_k\right\}\right)$$

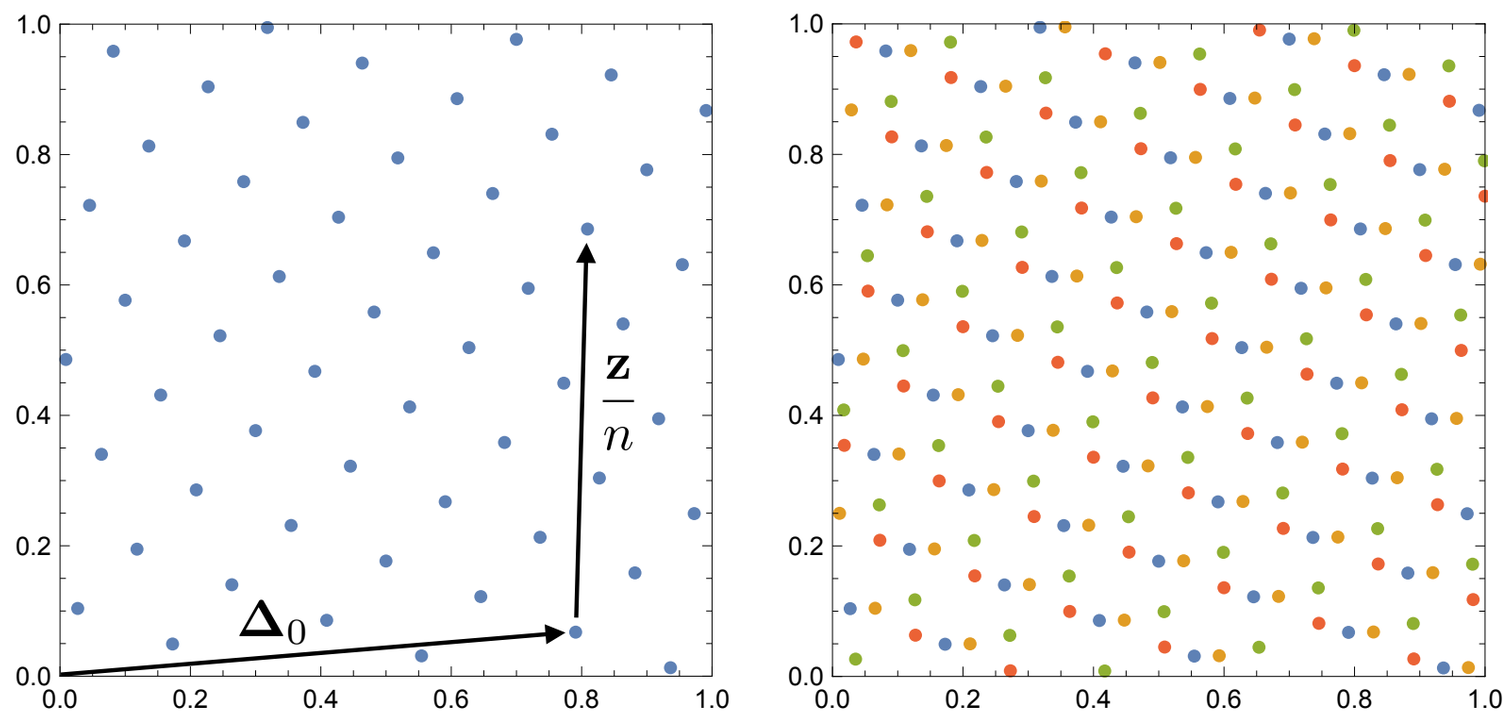
$\mathbf{z}$ - Generating vec.

$\boldsymbol{\Delta}_k$ - Random shift vec.

$\{\}$ - Fractional part

n - # Lattice points

m - # Random shifts



1 Shift 4 Shifts

Unbiased error estimate computed using (10-50) random shifts

Periodizing Transforms

Lattice rules work especially well for continuous, smooth and periodic functions
Functions can be periodized by a suitable change of variables: $\mathbf{x} = \phi(\mathbf{u})$

$$I[f] \equiv \int_{[0,1]^d} \mathrm{d}\mathbf{x}\ f(\mathbf{x}) = \int_{[0,1]^d} \mathrm{d}\mathbf{u}\ \omega_d(\mathbf{u}) f(\phi(\mathbf{u}))$$

Review: Dick, Kuo, Sloan 13

$$\phi(\mathbf{u}) = (\phi(u_1), \ldots, \phi(u_d)), \qquad \omega_d(\mathbf{u}) = \prod_{j=1}^{d} \omega(u_j) \qquad \text{and} \qquad \omega(u) = \phi'(u)$$

Korobov transform: $\omega(u) = 6u(1-u), \quad \phi(u) = 3u^2 - 2u^3$

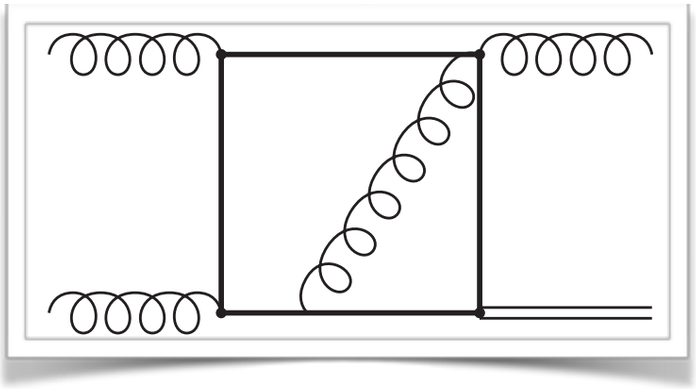
Sidi transform: $\omega(u) = \pi/2 \sin(\pi u), \quad \phi(u) = 1/2(1 - \cos \pi t)$

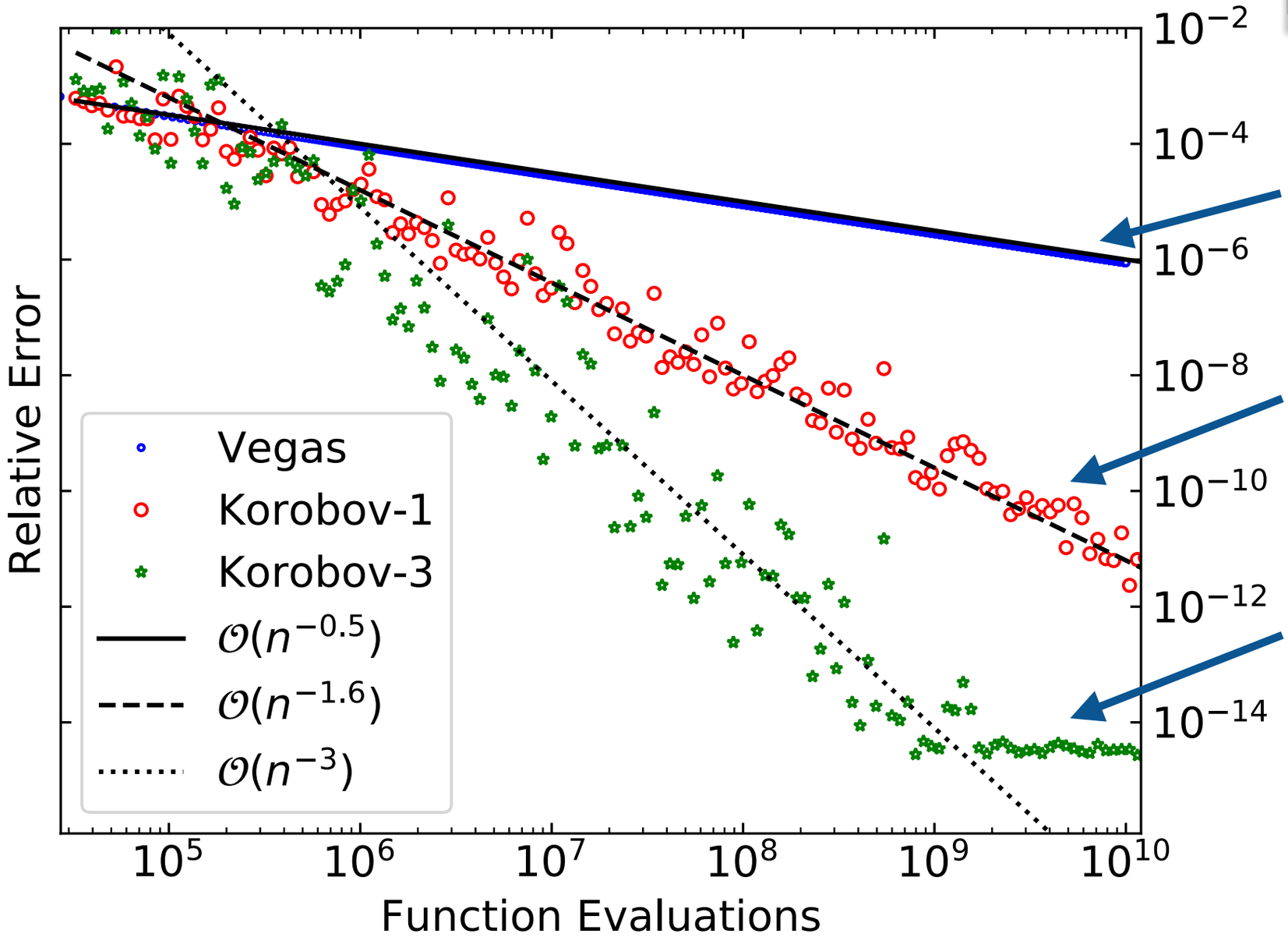
Baker transform: $\phi(u) = 1 - |2u - 1|$

Scaling

Example:

Sector Decomposed HJ Integral





Monte Carlo scaling

Better than ``guaranteed" n^{-1} scaling

Limited by machine precision (double)

Variance Reduction

Can improve performance of numerical integrator by flattening integrand via a variable transformation $y = p(x)$

$$I = \int_0^1 \mathrm{d}y \; f(y) = \int_0^1 \mathrm{d}x \; p'(x) f(p(x)) \qquad \text{s.t.} \qquad p'(x) \propto |f(p(x))|^{-1}$$

VEGAS:

Assume integrand separable $f(\mathbf{x}) = g(x_1)g(x_2)\cdots g(x_d)$

Iteratively approximate $p(x)$ with piecewise linear function

But: Spoils smoothness of integrand (bad for QMC)

Alternative:

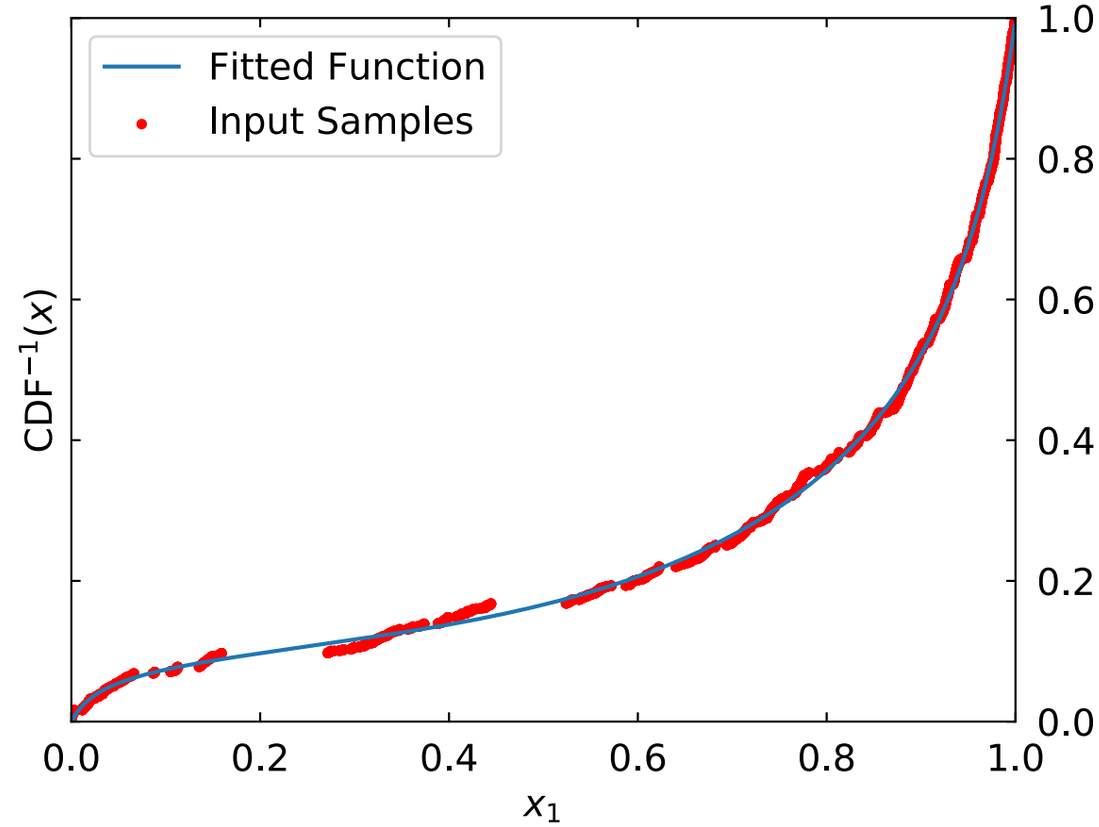
Choose a smooth function for $p(x)$, for example:

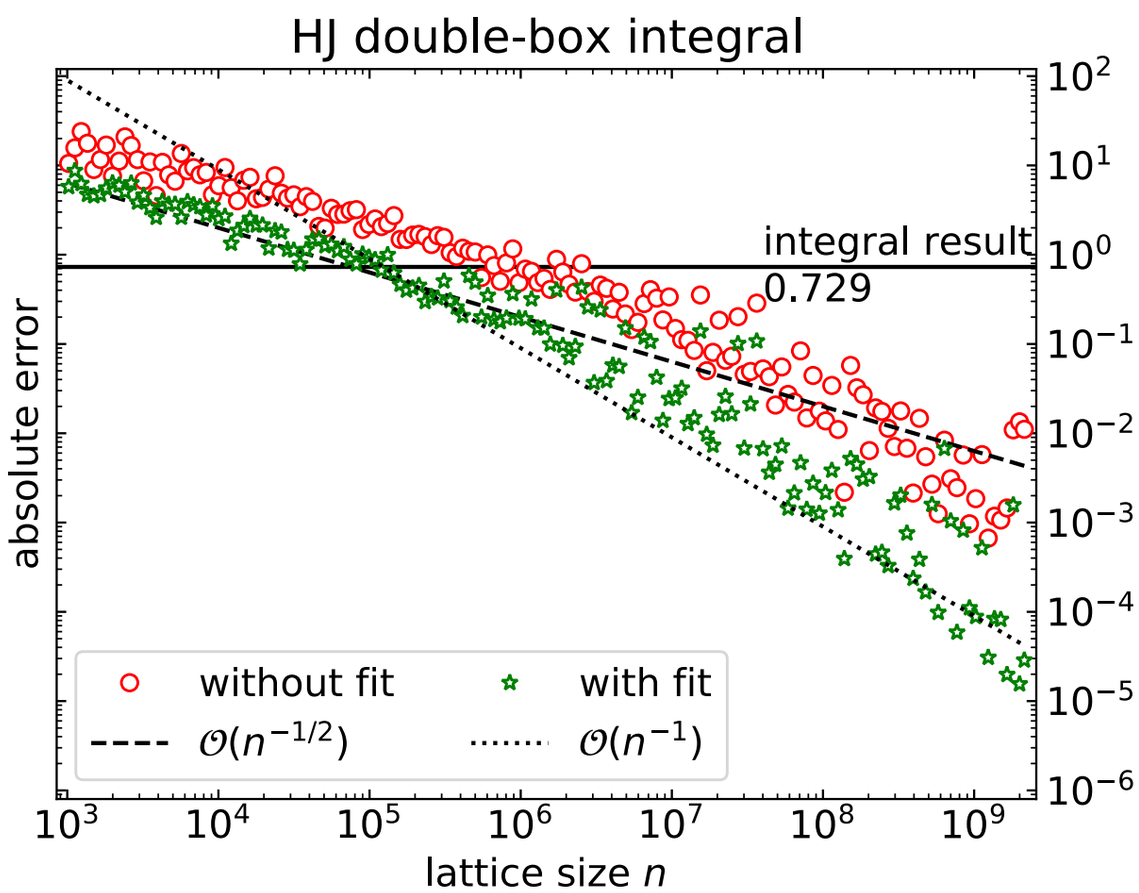
$$p(x) = a_2 \cdot x \, \frac{a_0 - 1}{a_0 - x} + a_3 \cdot x \, \frac{a_1 - 1}{a_1 - x} + a_4 \cdot x + a_5 \cdot x^2 + \left(1 - \sum_{i=2}^{5} a_i\right) \cdot x^3$$

Fit parameters a_i to inverse cumulative distribution function (CDF)

Variance Reduction (II)

1) Pre-sample integrand
2) In each dimension: fit inverse CDF
3) Use fit as variable transformation, obtain flatter integrand without spoiling smoothness





Example: HJ Integral

Small n: ~ 3x improvement
Large n: ~ 10x improvement

QMC Scaling preserved

GPU Usage for Calculating Feynman Integrals

FIESTA4 (Cuba with GPUs + Sector Decomposition) Smirnov 16

- Widely used in community, e.g. 4-loop $\overline{\mathrm{MS}}$-on-shell quark mass relation in QCD Marquard, Smirnov, Smirnov, Steinhauser, Wellmann 16

Lattice Rules + Sector Decomposition

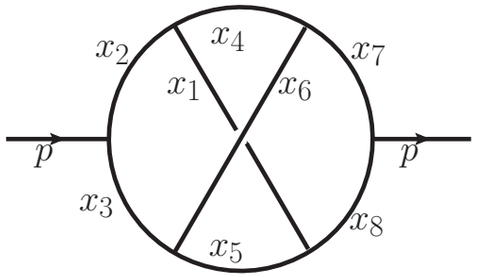
- 2-loop QCD Higgs Boson pair production
- 2-loop QCD Higgs Boson + Jet production

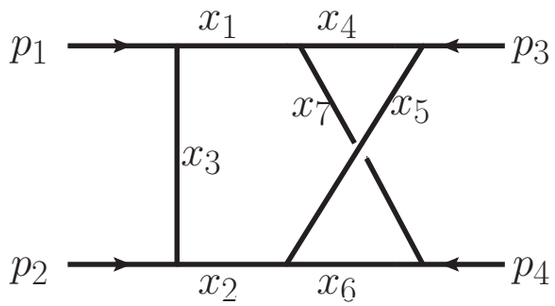
Li, Wang, Yan, Zhao 15; Borowka, Greiner, Heinrich, SPJ, Kerner, Schlenk, Schubert, Zirke 16; SPJ, Kerner, Luisoni 18

Lattice Rules + $\epsilon \to 0$ Extrapolation

- 2-3 loop box/self-energy diagrams

de Doncker, Almulihi, Yuasa 17, 18





→Talk of de Doncker

Monte Carlo Integration of Hepp Sectors Volkov 18

- 2-4 loop QED contributions to the electron magnetic moment
- 5-6 loop "ladder" form factor integrals

→Talk of Volkov

qmc: Design

QMC integration is embarrassingly parallel and simple to implement

We aimed to write a public package that is:

1) Fast
2) Easy to use
3) Supports multiple CPUs/GPUs (on a single machine)
4) Flexible and reasonably easy to develop and extend

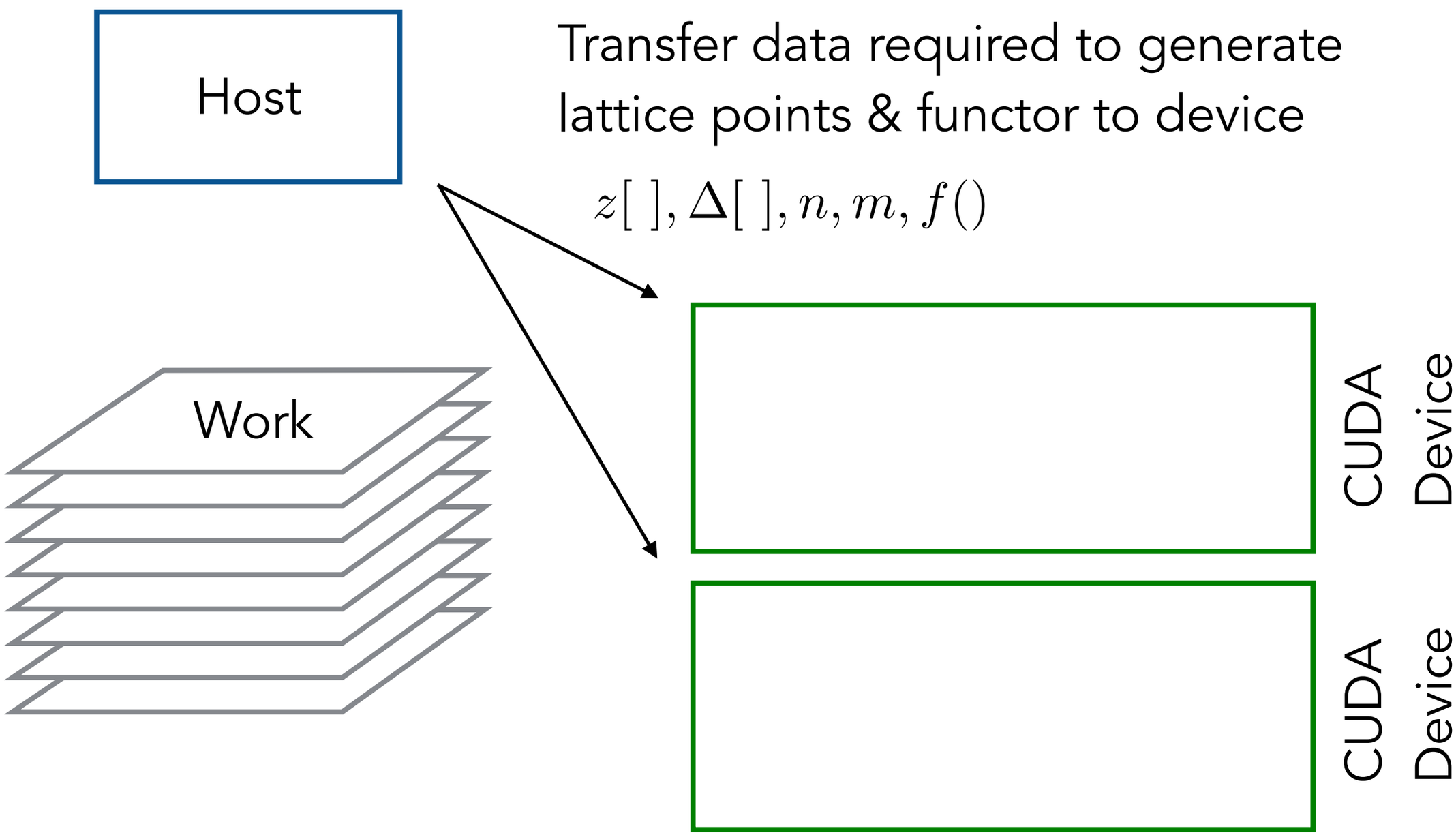
Language Choices:

CUDA (fairly developed language, decent compiler)

c++11 (object orientated, std::thread, templates, CUDA support)

qmc: Implementation

1) Devices are initialised

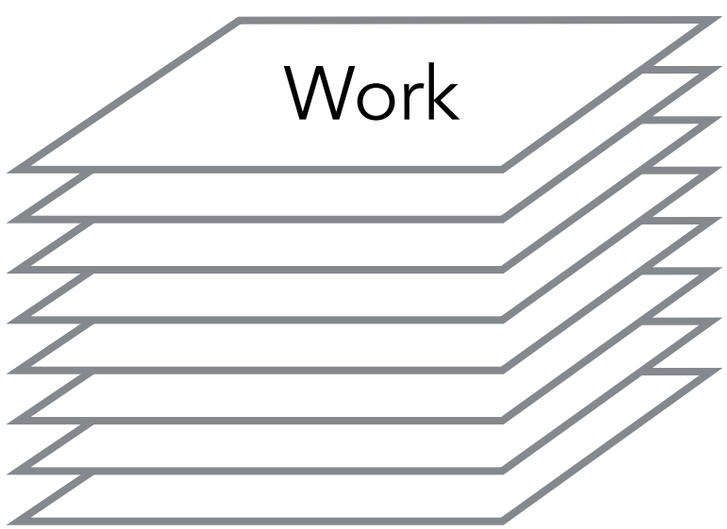


qmc: Implementation

2) (Loop) Work is requested from central work queue by each device

Host

Only index of point(s) to calculate needs to be transferred
Partial results are accumulated by each thread on each device



i →

$f\left(\left\{\frac{i\mathbf{z}}{n}+\mathbf{\Delta}_0\right\}\right)$	$f\left(\left\{\frac{i\mathbf{z}}{n}+\mathbf{\Delta}_1\right\}\right)$	$\dots$
$f\left(\left\{\frac{(i+1)\mathbf{z}}{n}+\mathbf{\Delta}_0\right\}\right)$	$f\left(\left\{\frac{(i+1)\mathbf{z}}{n}+\mathbf{\Delta}_1\right\}\right)$	$\dots$
$\dots$	$\dots$	

Thread 0
Thread 1
…

i' →

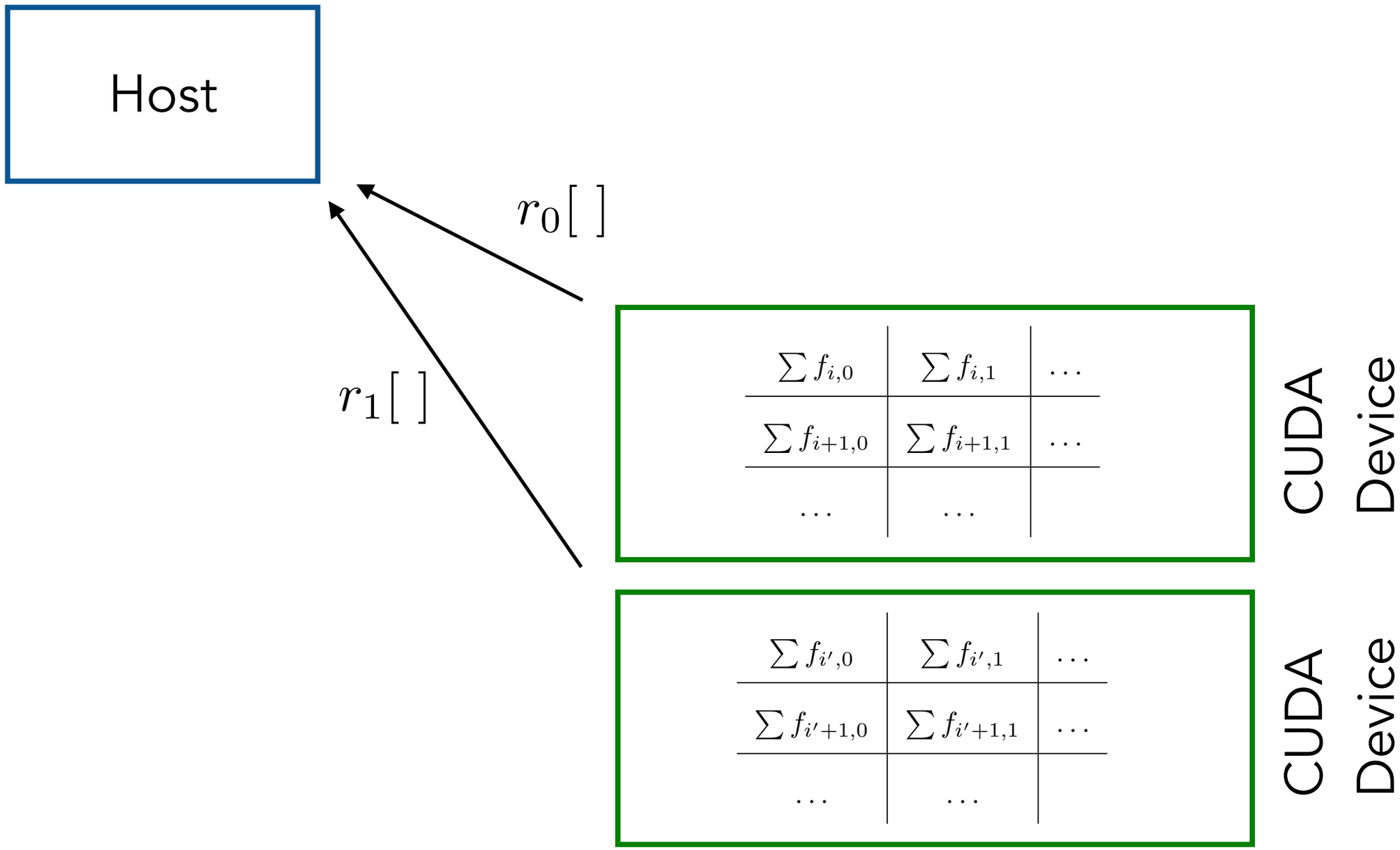
$f\left(\left\{\frac{i'\mathbf{z}}{n}+\mathbf{\Delta}_0\right\}\right)$	$f\left(\left\{\frac{i'\mathbf{z}}{n}+\mathbf{\Delta}_1\right\}\right)$	$\dots$
$f\left(\left\{\frac{(i'+1)\mathbf{z}}{n}+\mathbf{\Delta}_0\right\}\right)$	$f\left(\left\{\frac{(i'+1)\mathbf{z}}{n}+\mathbf{\Delta}_1\right\}\right)$	$\dots$
$\dots$	$\dots$	

Thread 0
Thread 1
…

Work is actually just an atomically locked counter

qmc: Implementation

3) Once all work is completed, partial results are transferred back to host for final reduction

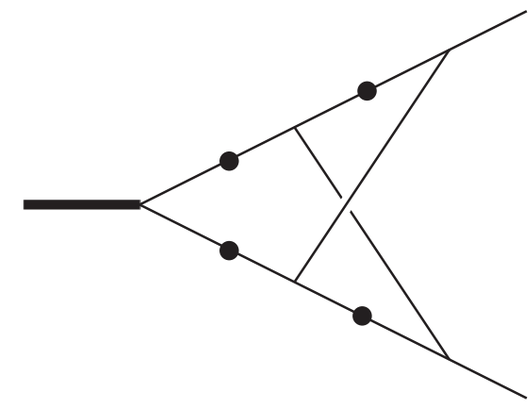


qmc: Usage

Let's compute a finite Feynman integral (à la de Doncker, Almulihi, Yuasa 17)

```
#include <iostream>
#include "qmc.hpp"
struct formfactor2L_t {
    const unsigned long long int number_of_integration_variables = 5;
#ifdef __CUDACC__
    __host__ __device__
#endif
    double operator()(const double arg[]) const
    {
        // Simplex to cube transformation
        double x0  = arg[0];
        double x1  = (1.-x0)*arg[1];
        double x2  = (1.-x0-x1)*arg[2];
        double x3  = (1.-x0-x1-x2)*arg[3];
        double x4  = (1.-x0-x1-x2-x3)*arg[4];
        double x5  = (1.-x0-x1-x2-x3-x4);
        double wgt =
        (1.-x0)*
        (1.-x0-x1)*
        (1.-x0-x1-x2)*
        (1.-x0-x1-x2-x3);
        if(wgt <= 0) return 0;
        // Integrand
        double u=x2*(x3+x4)+x1*(x2+x3+x4)+(x2+x3+x4)*x5+x0*(x1+x3+x4+x5);
        double f=x1*x2*x4+x0*x2*(x1+x3+x4)+x0*(x2+x3)*x5;
        double n=x0*x1*x2*x3;
        double d = f*f*u*u;
        return wgt*n/d;
    }
} formfactor2L;
```

$(8-2\epsilon)$



Note: can compile this code with or without CUDA

```
int main() {
    integrators::Qmc<double,double,5,integrators::transforms::Korobov<3>::type> integrator;
    integrator.minn = 100000000; // (optional) lattice size
    integrators::result<double> result = integrator.integrate(formfactor2L);
    std::cout << "integral = " << result.integral << std::endl;
    std::cout << "error    = " << result.error    << std::endl;
    return 0;
}
```

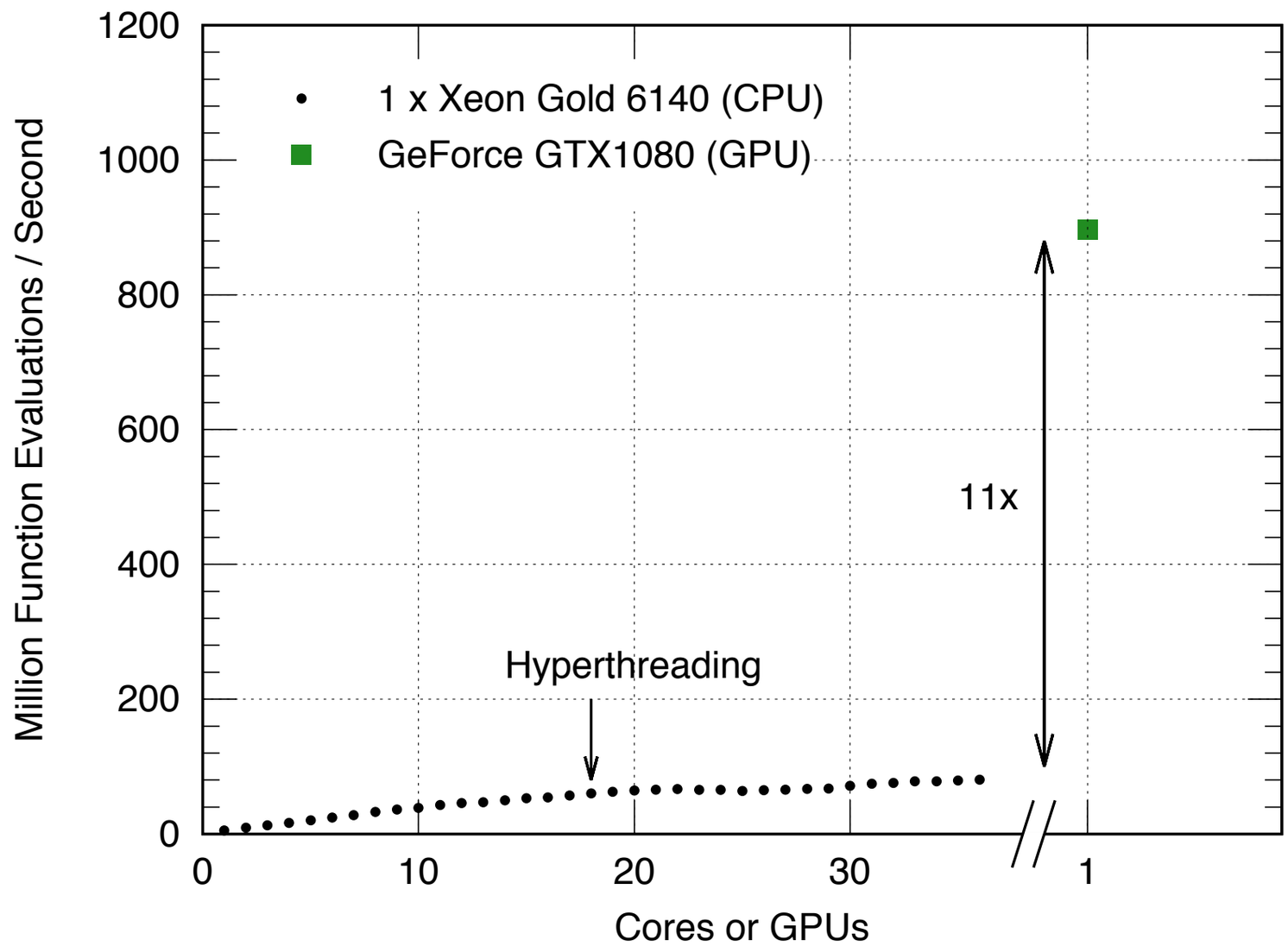
```
$ nvcc -O3 -arch=sm_70 -std=c++11 -x cu -I../src 102_ff2_demo.cpp.cpp -o 102_ff2_demo.out -lgsl -lgslcblas && ./102_ff2_demo.out
integral = 0.27621
error    = 4.49751e-07
```

(Agrees with analytic result)

qmc: Performance

Accuracy limited by number of function evaluations

Can accelerate this using Graphics Processing Units (GPUs)



Device	M Func. Evals/s	Speedup
Xeon 6140	80.6	-
GTX1080	897	11x

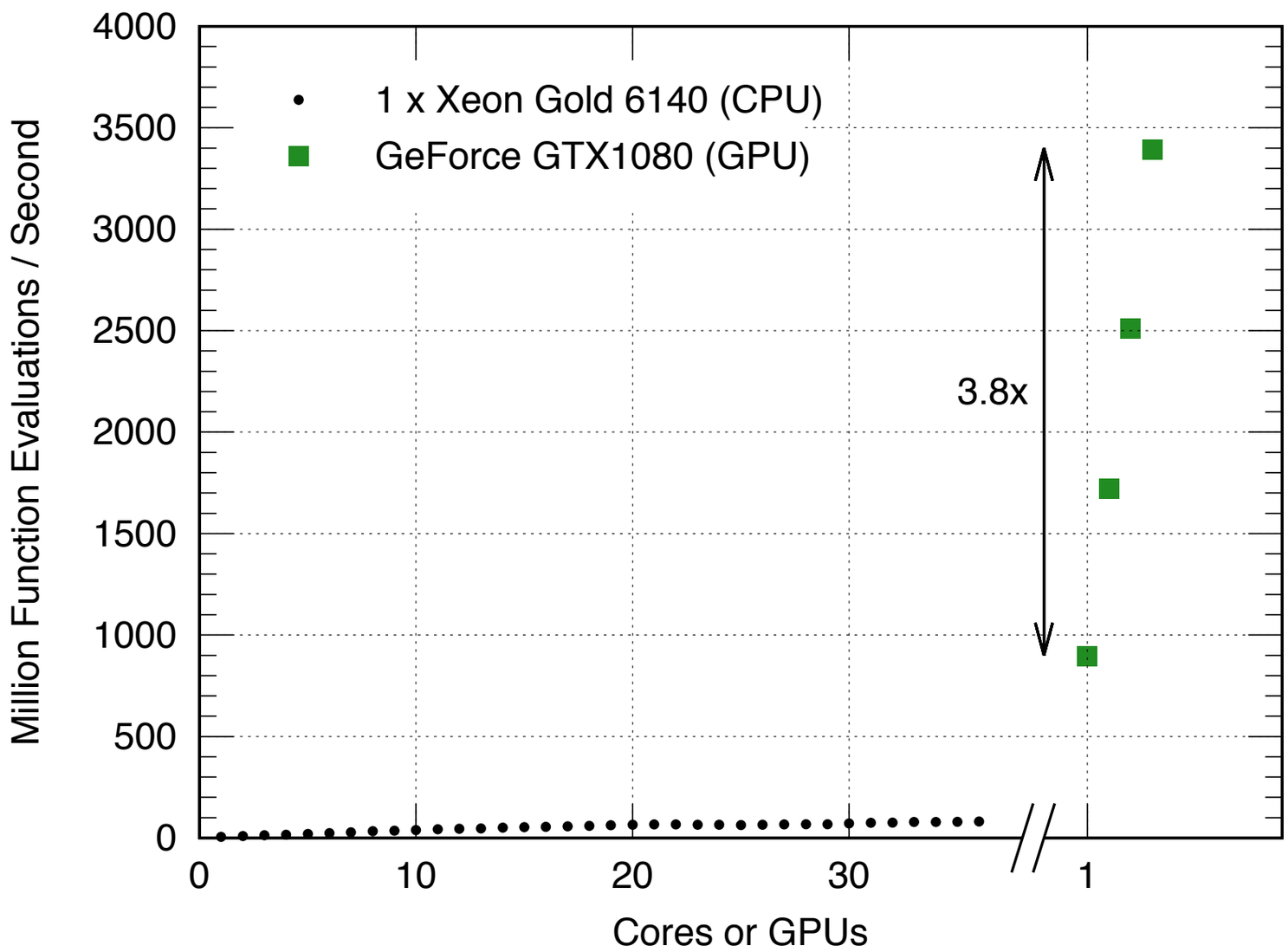
Note: Performance gain highly dependent on integrand & hardware!

Still room for further optimisations (both for CPU and GPU)

qmc: Performance

Accuracy limited by number of function evaluations

Can accelerate this using Graphics Processing Units (GPUs)



Device	M Func. Evals/s	Speedup
Xeon 6140	80.6	-
GTX1080	897	11x

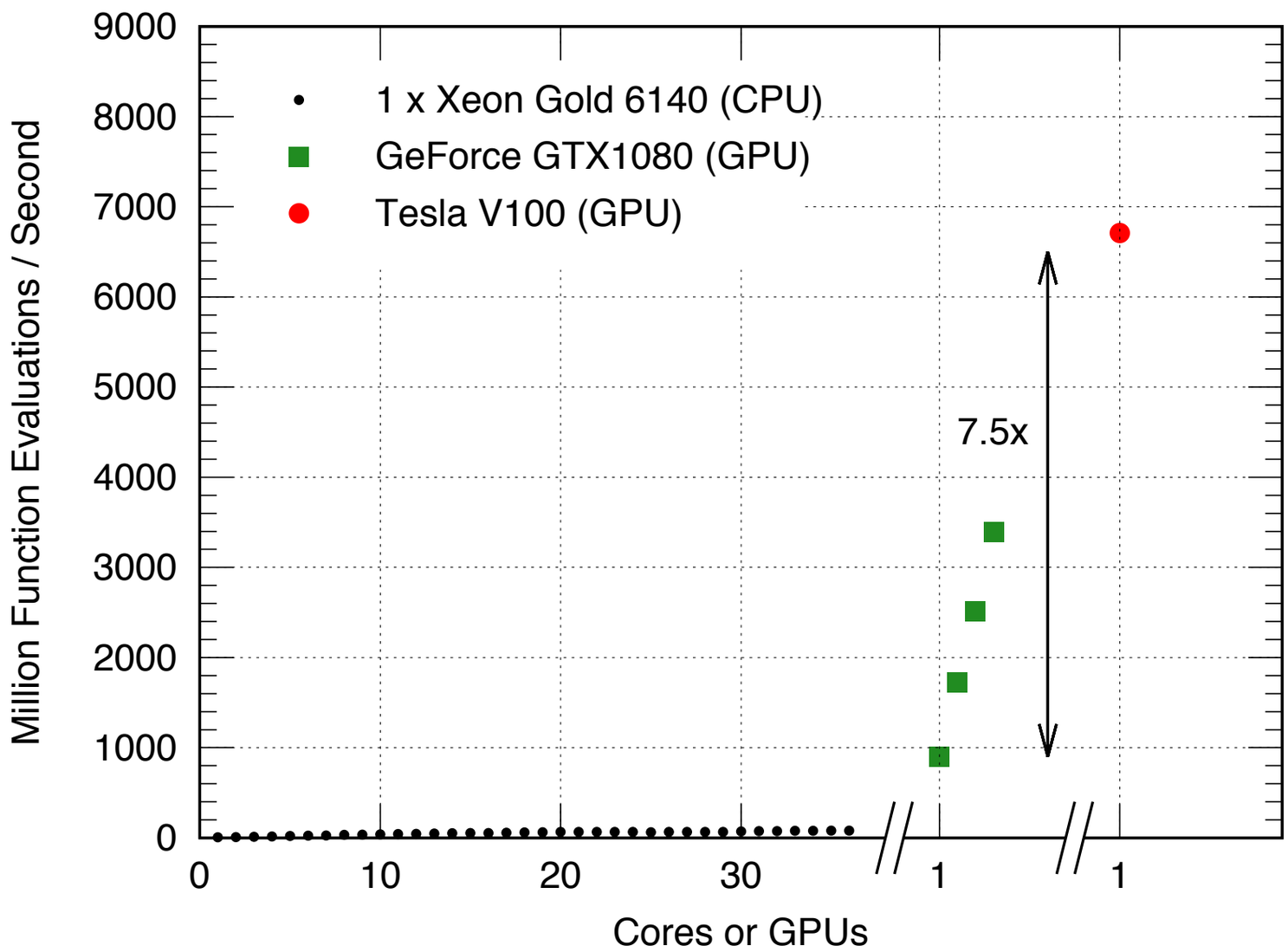
Note: Performance gain highly dependent on integrand & hardware!

Still room for further optimisations (both for CPU and GPU)

qmc: Performance

Accuracy limited by number of function evaluations

Can accelerate this using Graphics Processing Units (GPUs)



Device	M Func. Evals/s	Speedup
Xeon 6140	80.6	-
GTX1080	897	11x
Tesla V100	6710	83x

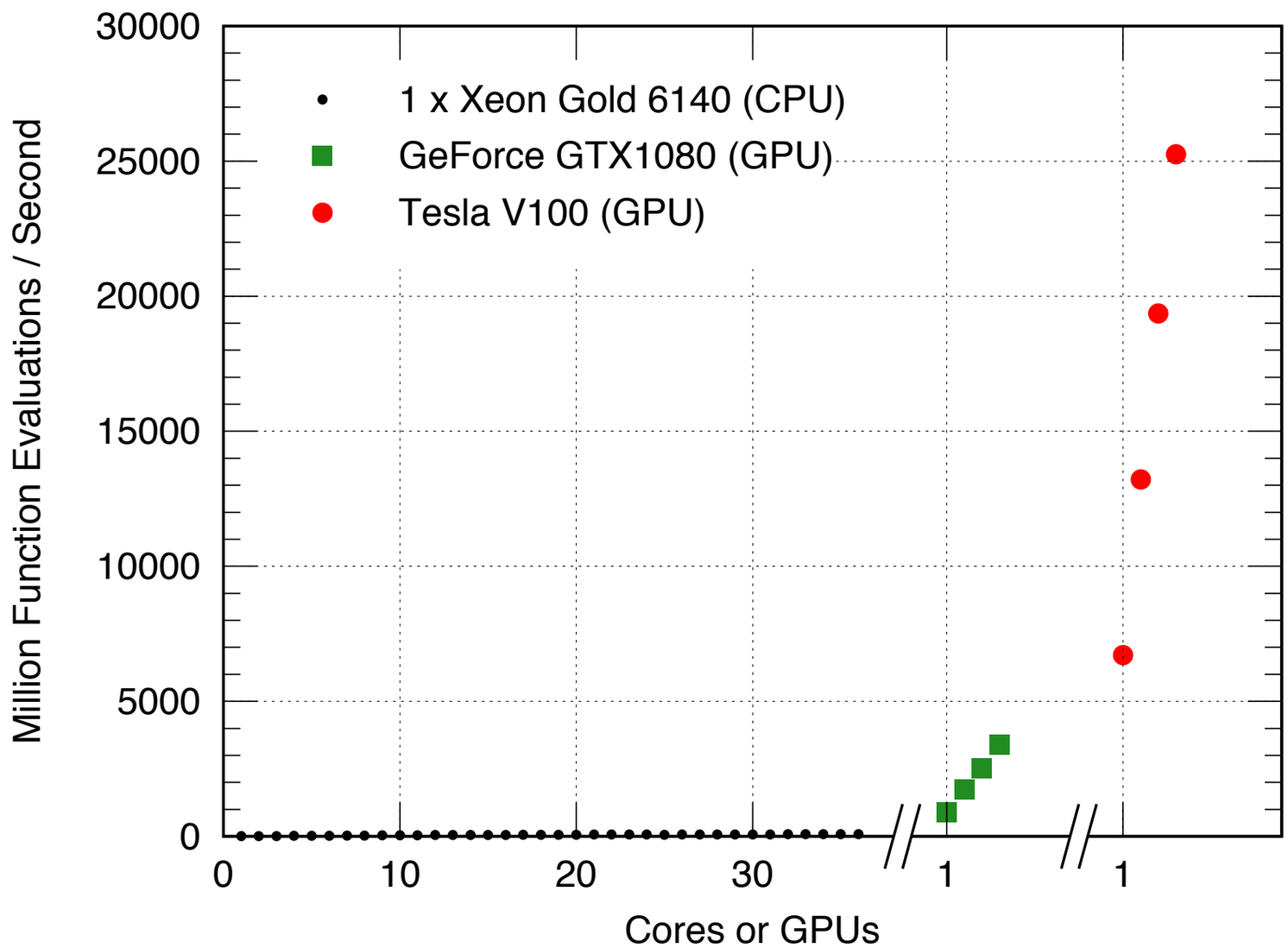
Note: Performance gain highly dependent on integrand & hardware!

Still room for further optimisations (both for CPU and GPU)

qmc: Performance

Accuracy limited by number of function evaluations

Can accelerate this using Graphics Processing Units (GPUs)



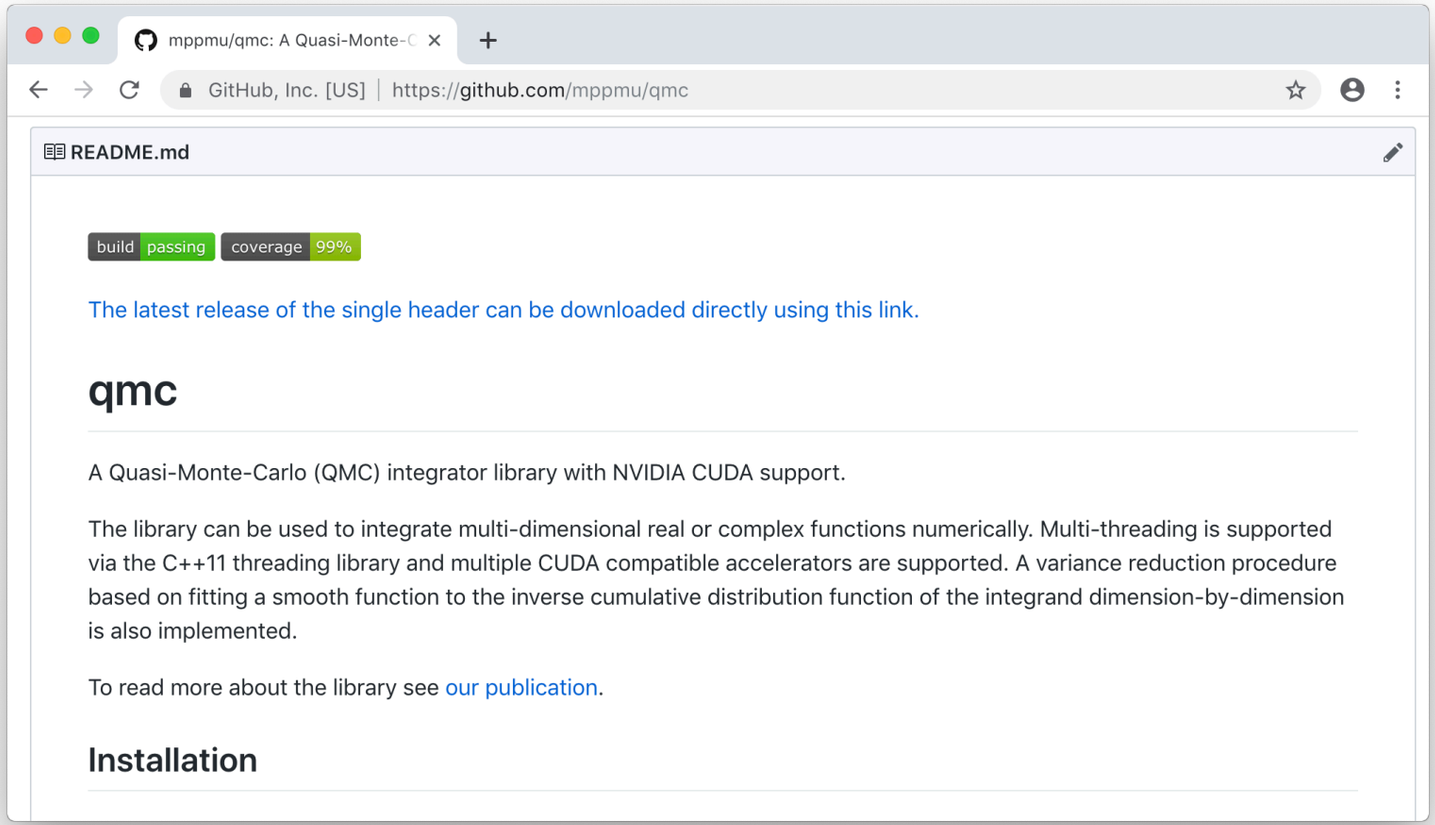
Device	M Func. Evals/s	Speedup
Xeon 6140	80.6	-
GTX1080	897	11x
Tesla V100	6710	83x

Note: Performance gain highly dependent on integrand & hardware!

Still room for further optimisations (both for CPU and GPU)

qmc: Installation

qmc: distributed as a standalone single header c++11 library
quite flexible: define your own lattices, integral transforms, floating-point types (e.g… Boost multi-precision)



Publicly available (Github)

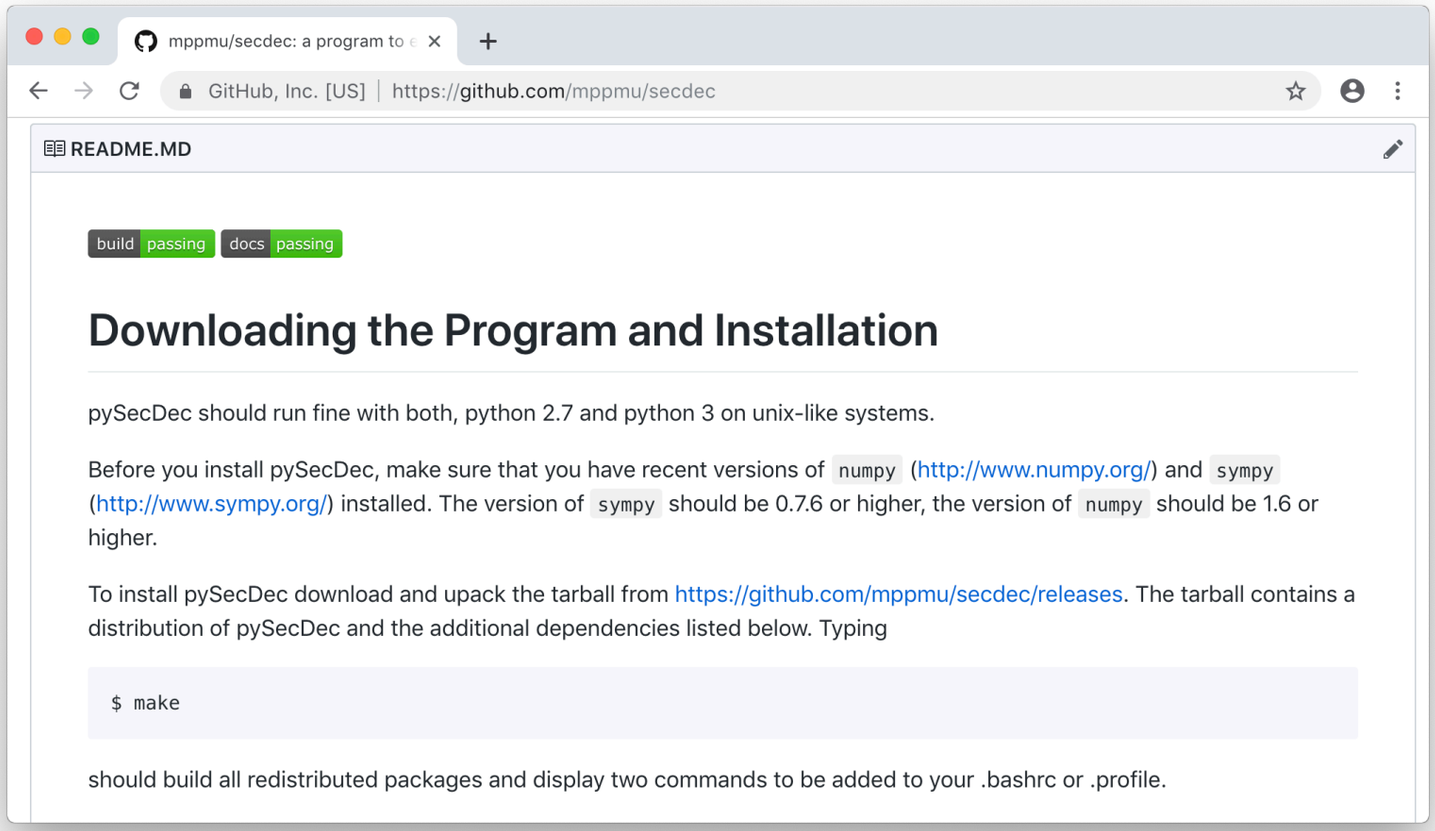
Extensive tests (CI) and coverage

Borowka, Heinrich, Jahn, SPJ, Kerner, Schlenk

pySecDec

pySecDec: a program to numerically evaluate dimensionally regulated parameter integrals (written in python, FORM & c++)

Vermaseren 00; Kuipers, Ueda, Vermaseren 13; Ruijl, Ueda, Vermaseren 17



Publicly available (Github)

Extensive tests (CI) and documentation

Borowka, Heinrich, Jahn, SPJ, Kerner, Schlenk, Zirke

New: Quasi-Monte Carlo integration & CUDA GPU Support

Borowka, Heinrich, Jahn, SPJ, Kerner, Schlenk 18

Other Sector Decomposition Tools

Several other codes implement sector decomposition

Public:

- sector_decomposition + CSectors (uses GiNaC)
  http://wwwthep.physik.uni-mainz.de/~stefanw/sector_decomposition
  Bogner, Weinzierl 07; Gluza, Kajda, Riemann, Yundin 10

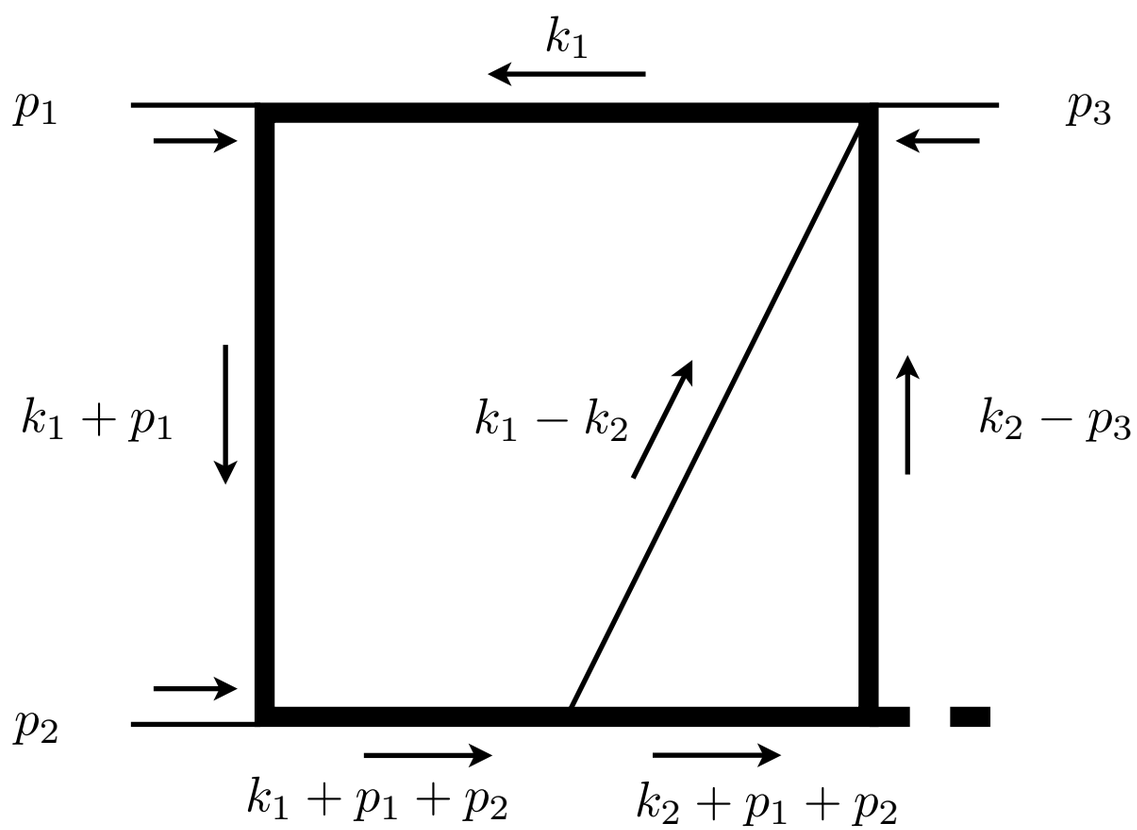
- FIESTA (uses Mathematica, C)
  http://science.sander.su/FIESTA.htm
  Supports integration on GPUs
  A. Smirnov, V. Smirnov, Tentyukov 08, 09, 13, 15; Smirnov 16

(Currently) Private:

- FORM & Python Implementations
  Fujimoto, Kaneko and Ueda 08, 10
- NIFT
  Zhao (in preparation)

Example: elliptic2L_euclidean

Massive 2-loop Box



Bonciani, Del Duca, Frellesvig, Henn, Moriello, Smirnov 16

Propagators:

$$(k_1)^2 - m^2$$
$$(k_1 + p_1 + p_2)^2 - m^2$$
$$(k_2 + p_1 + p_2)^2 - m^2$$
$$(k_1 + p_1)^2 - m^2$$
$$(k_1 - k_2)^2$$
$$(k_2 - p_3)^2 - m^2$$

Scalar Products:

$$p_1 \cdot p_1 = 0 \qquad p_2 \cdot p_2 = 0 \qquad p_3 \cdot p_3 = 0$$
$$p_1 \cdot p_2 = s/2 \qquad p_1 \cdot p_3 = t/2 \qquad p_2 \cdot p_3 = (m_H^2 - s - t)/2$$

Example: elliptic2L_euclidean (II)

Step 1: Write input files

generate_elliptic2L_euclidean.py

```
from pySecDec.loop_integral import loop_package
import pySecDec as psd

li = psd.loop_integral.LoopIntegralFromPropagators(
propagators = ['k1**2-msq', \
               '(k1+p1+p2)**2-msq', \
               '(k2+p1+p2)**2-msq', \
               '(k1+p1)**2-msq', \
               '(k1-k2)**2', \
               '(k2-p3)**2-msq'],
loop_momenta = ['k1','k2'],
replacement_rules = [
                        ('p1*p1',0),
                        ('p2*p2',0),
                        ('p3*p3',0),
                        ('p1*p2','s/2'),
                        ('p2*p3','pp4/2-s/2-t/2'),
                        ('p1*p3','t/2')
                    ]
)

Mandelstam_symbols = ['s','t','pp4']
mass_symbols = ['msq']

loop_package(
name = 'elliptic2L_euclidean',
loop_integral = li,
real_parameters = Mandelstam_symbols + mass_symbols,
additional_prefactor = '(-s/msq)**(3/2)',
requested_order = 0,
contour_deformation = False
)
```

Or: LoopIntegralFromGraph (Adjacency list)

Propagators

Scalar products

Result multiplied by this

Euclidean region

Example: elliptic2L_euclidean (III)

Step 1: Write input files

integrate_elliptic2L_euclidean.py

```
from pySecDec.integral_interface import IntegralLibrary
elliptic2L_euclidean = IntegralLibrary('elliptic2L_euclidean/elliptic2L_euclidean_pylink.so')
elliptic2L_euclidean.use_Qmc(minn=10000000,transform='korobov5')
s, t, pp4, msq = [-4./3.,-16./5.,-100./39.,1.]
_, _, result = elliptic2L_euclidean([s, t, pp4, msq])
print(result)
```

Select qmc integrator

Step 2: Generate & build c++ library, run integrator

```
$ python generate_elliptic2L_euclidean.py
$ CXX=nvcc SECDEC_WITH_CUDA=sm_70 make -C elliptic2L_euclidean
$ time python integrate_elliptic2L_euclidean.py
 + (2.4707419914073156 0e-01 +/- 4.3607559980517135 5e-16) + O(eps)
real        0m2.576s
```

Compile with nvcc

2 x Xeon Gold 6140 + 4 x Tesla V100

Integration performed on all CPU cores & GPUs on the system

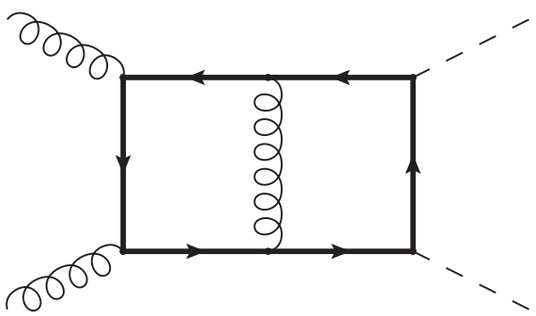
Agrees with analytic result: Bonciani, Del Duca, Frellesvig, Henn, Moriello, Smirnov 16

Physics Applications

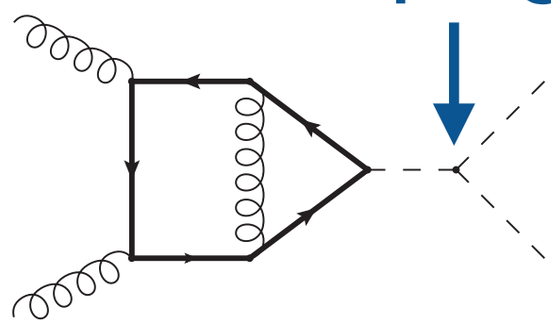
HH and HJ Production @ NLO

Di-Higgs Boson and Higgs Boson+Jet production are loop induced

HH Production



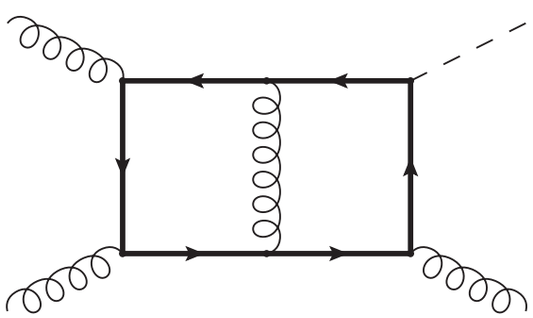
HHH coupling



Sensitive to Higgs self coupling, probe of EW symmetry breaking

$$\frac{m_H^2}{2}H^2 + \boxed{\frac{m_H^2}{2v}H^3} + \frac{m_H^2}{8v^2}H^4$$

HJ Production



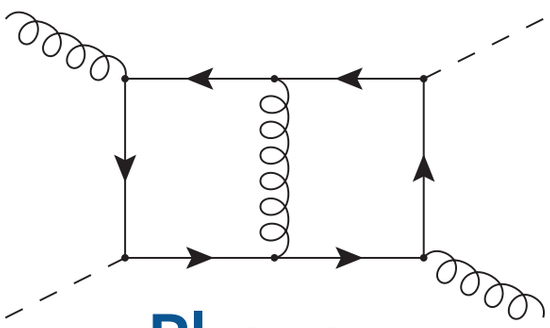
Contributes to Higgs p_T distribution
At large p_T can "probe inside the loop", distributions can be modified by heavy BSM particles

NLO corrections (2-loop) are currently known only numerically

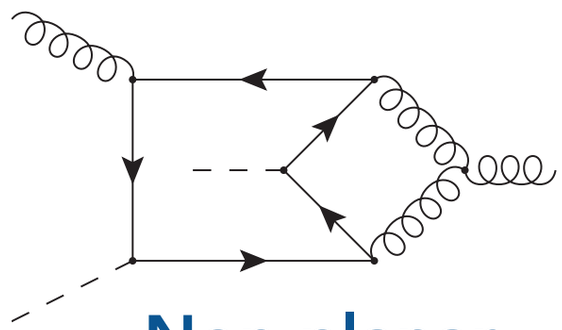
Higgs Boson Production: NLO QCD Results

NLO QCD corrections to HJ/HH production with full top quark mass:
≤7 propagator, 4-point, 2-loop diagrams, 4 mass scales (s, t, m_T, m_H)

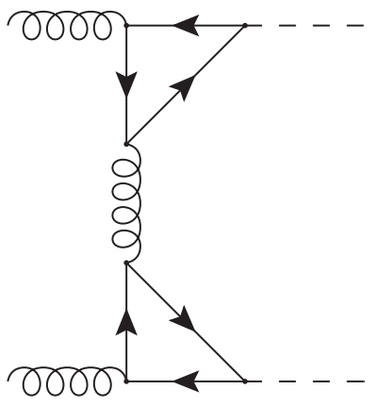
HH:



Planar

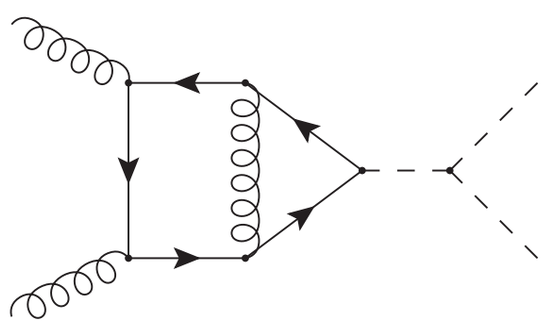


Non-planar



Reducible

Degrassi, Giardino, Gröber 16

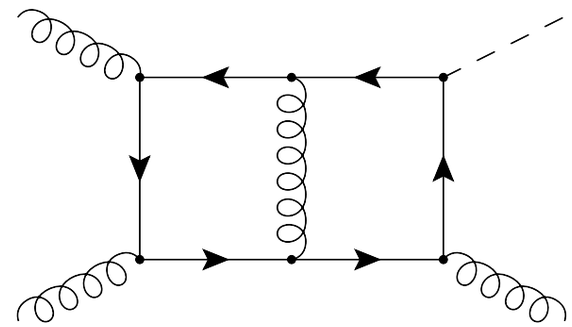


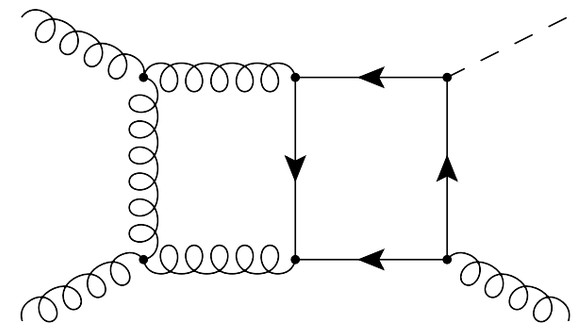
$gg \to H$

Spira, Djouadi et al. 93, 95; Bonciani, P. Mastrolia 03,04; Anastasiou, Beerli et al. 06;

$H \to Z\gamma$ Gehrmann, Guns, Kara 15;

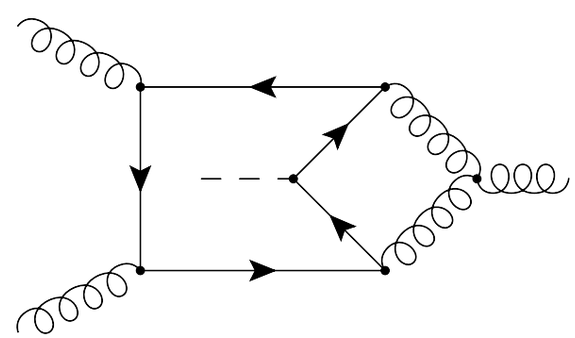
HJ:

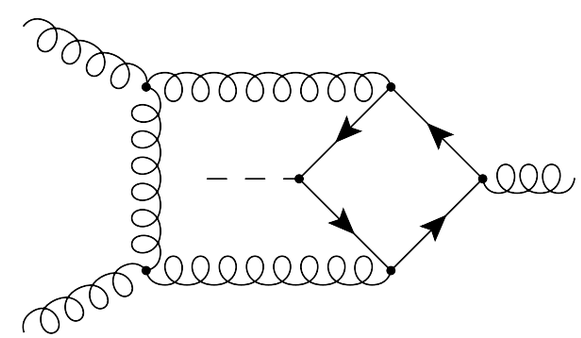




Planar

Bonciani, Del Duca, Frellesvig, Henn, Moriello, Smirnov 16; (See also Primo, Tancredi 16)





Non-planar

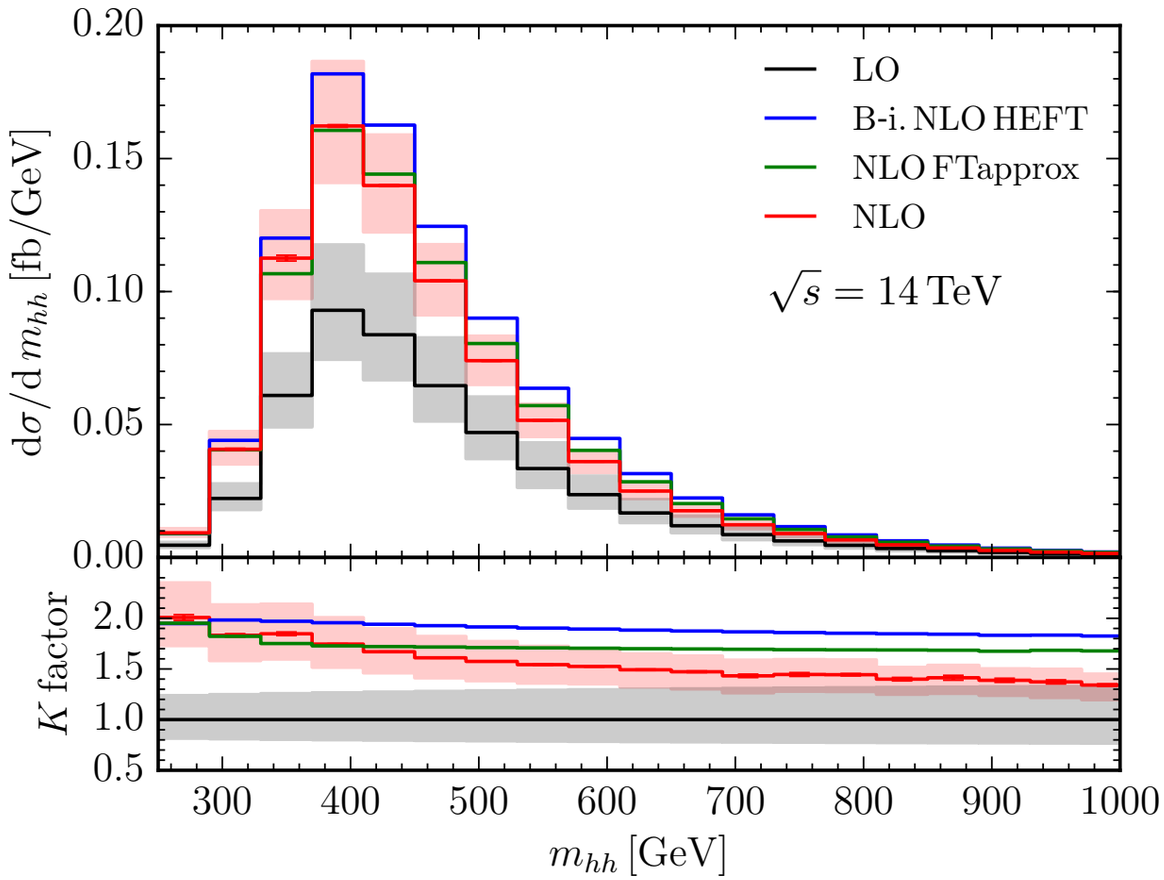
Comparison of HJ and HH

	HJ production	HH production
#Form factors	4+2	2
Full reduction	✓	only planar
(quasi-) finite basis	✓	only planar
#Master integrals including crossings	458	327*
#Master integrals neglecting crossings	120	215*
#Integrals after sector decomposition and expansion in ε	22675	11244
Code size coefficients	~340 MB	~80 MB
Code size integrals	~330 MB	~580 MB
Compile time coefficients	~2 weeks	few days
Compile time integrals	~4 hours	~1-2 days
Time for linking the program	~3-4 days	few hours

Slide: Matthias Kerner, Radcor 2017

* HH non-planar not fully reduced

Higgs Boson Pair Results



	σ_{LO} (fb)	σ_{NLO} (fb)
B.I. HEFT	$19.85^{+27.6\%}_{-20.5\%}$	$38.32^{+18.1\%}_{-14.9\%}$
FTapprox	$19.85^{+27.6\%}_{-20.5\%}$	$34.26^{+14.7\%}_{-13.2\%}$
Full Theory	$19.85^{+27.6\%}_{-20.5\%}$	$\mathbf{32.91^{+13.6\%}_{-12.6\%}}$

Borowka, Greiner, Heinrich, SPJ, Kerner, Schlenk, Schubert, Zirke 16;

Median 2 GPU hours per phase-space point (important to sample PS well)

Interfaced to public Monte-Carlo+Parton Shower codes, dependence on Higgs Boson self-coupling can be explored

→Talk of Scyboz

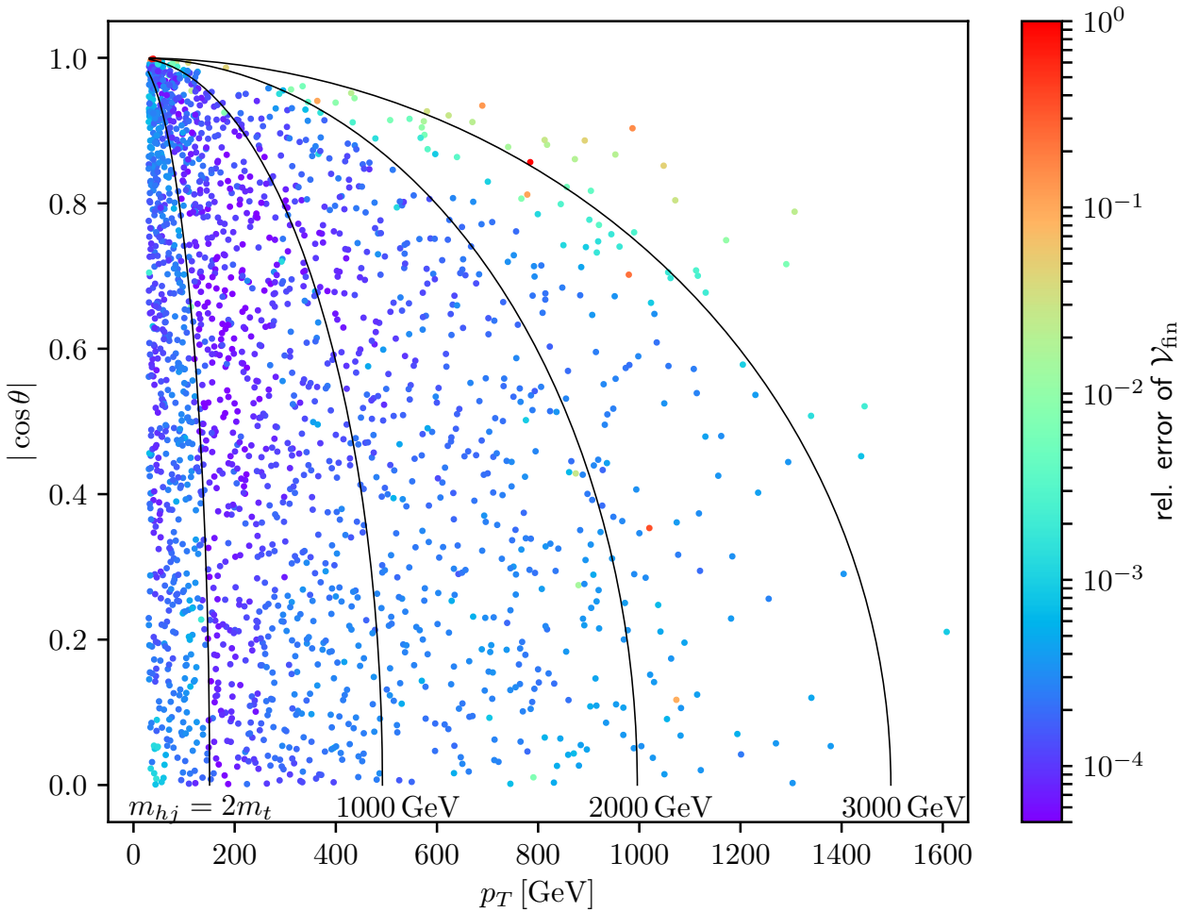
Complementary to analytic results (now known in high/low energy limit)

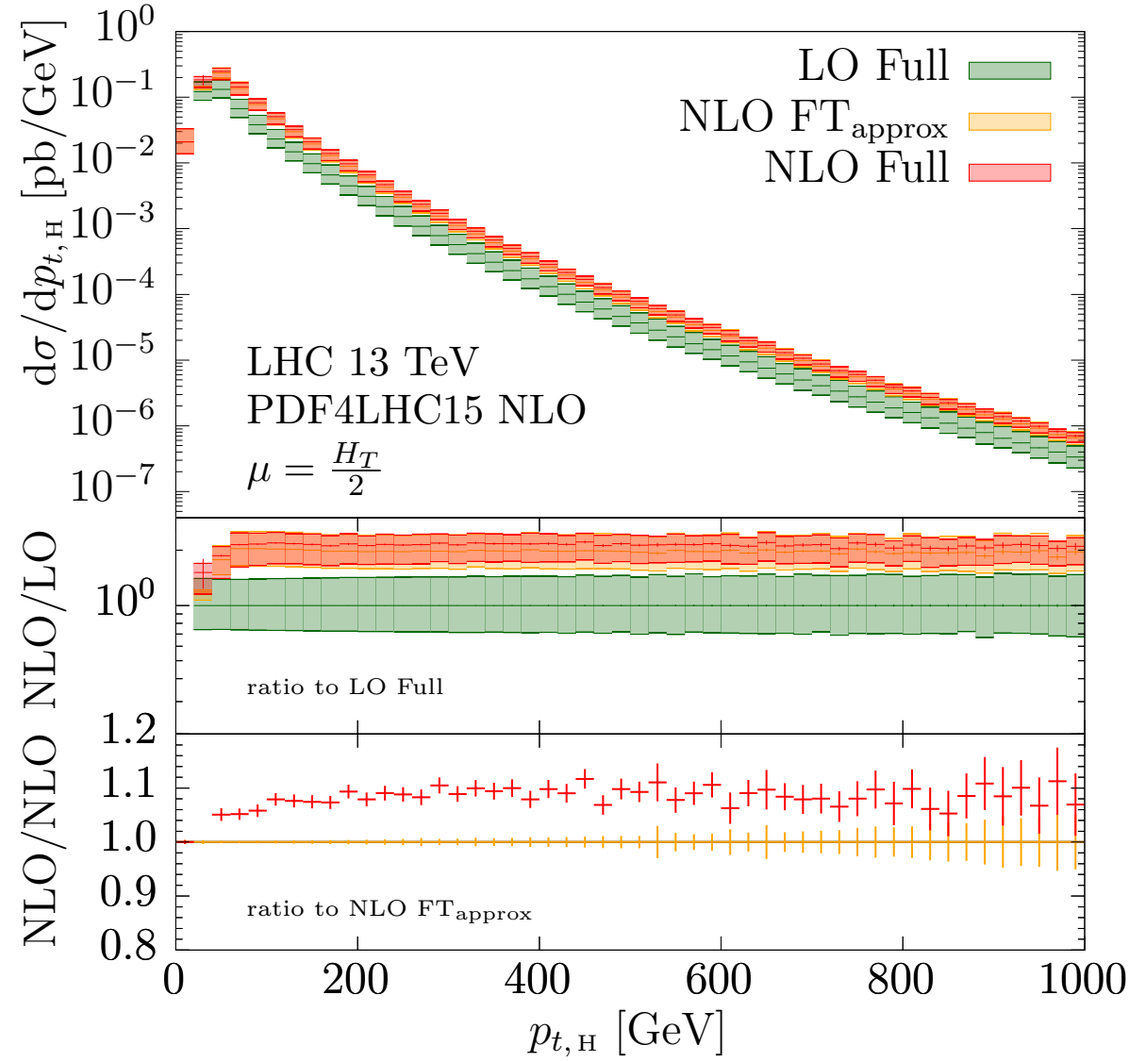
→Talk of Davies

HH recently recomputed by another group (also using numerical methods)

Baglio, Campanario, Glaus, Mühlleitner, Spira, Streicher 18

Higgs Boson + Jet Results





SPJ, Kerner, Luisoni 18

Must evaluate the amplitude for many phase-space points, significant numerical stability improvements from carefully choosing basis of integrals

Full theory predicts nearly flat K factor at large p_T for $\mu = H_T/2$

~8% increase in tail by including top quark mass dependence in virtuals

Conclusion

Currently in an era where GPUs can be extremely useful for producing state-of-the-art HEP predictions

Numerical Multi-loop Calculations

- Several well developed approaches to computing Feynman Integrals
- Significant progress in attacking the problem of numerical integration itself and making better use of modern hardware

Physics Applications

- Presented HH & HJ at NLO with full top quark mass dependence
- 2-loop virtual amplitude calculated numerically on GPUs

Future

- Several multi-scale 2→2 processes left to attack
- Likely significant room for improvement: writing better optimised integrand functions, improving variance reduction techniques (NN?), …

Thank you for listening!

Backup

Weighted Function Spaces

Review: Dick, Kuo, Sloan 13

Assign weights $\gamma_{\mathfrak{u}}$ to each subset of dimension $\mathfrak{u} \subseteq \{1, \ldots, d\}$

Sobolev Space

Functions with square integrable first derivatives

Norm $$||f||_{\boldsymbol{\gamma}}^2 = \sum_{\mathfrak{u} \subseteq \{1,\ldots,d\}} \frac{1}{\gamma_{\mathfrak{u}}} \int_{[0,1]^{|\mathfrak{u}|}} \left(\int_{[0,1]^{d-|\mathfrak{u}|}} \frac{\partial^{|\mathfrak{u}|} f(\mathbf{x})}{\partial \mathbf{x}_{\mathfrak{u}}} \mathrm{d}\mathbf{x}_{-\mathfrak{u}} \right)^2 \mathrm{d}\mathbf{x}_{\mathfrak{u}}$$

Worst-case error $$e_{\boldsymbol{\gamma}}^2 \leq \left(\frac{1}{\psi(n)} \sum_{\emptyset \neq \mathfrak{u} \subseteq \{1,\ldots,d\}} \gamma_{\mathfrak{u}}^{\lambda} \left(\frac{2\zeta(2\lambda)}{(2\pi^2)^{\lambda}} \right)^{|\mathfrak{u}|} \right)^{\frac{1}{\lambda}}$$

$\lambda \in (1/2, 1]$

$$\boxed{\varepsilon \sim \mathcal{O}(N^{-1})}$$

Korobov Space

Periodic functions which are α times differentiable in each variable

$$||f||_{\boldsymbol{\gamma}}^2 = \sum_{\mathbf{h} \in \mathbb{Z}^d} \frac{\prod_{j \in \mathfrak{u}(\mathbf{h})} |h_j|^{2\alpha}}{\gamma_{\mathfrak{u}(\mathbf{h})}} |\hat{f}(\mathbf{h})|^2$$

Fourier Coefficient

$$e_{\boldsymbol{\gamma}}^2 \leq \left(\frac{1}{\psi(n)} \sum_{\emptyset \neq \mathfrak{u} \subseteq \{1,\ldots,d\}} \gamma_{\mathfrak{u}}^{\lambda} \left(2\zeta(2\alpha\lambda)\right)^{|\mathfrak{u}|} \right)^{\frac{1}{\lambda}}$$

$\lambda \in (1/(2\alpha), 1]$, smoothness α

$$\boxed{\varepsilon \sim \mathcal{O}(N^{-\alpha})}$$

Generating vector $\mathbf{z}$ precomputed for a **fixed** number of lattice points, chosen to minimise worst-case error Nuyens 07

A Brief Introduction to Computing Feynman Integrals

Many methods of computing Feynman Integrals exist

Differential Equations have proven themselves to be a powerful tool

Kotikov 91; Remiddi 97; Gehrmann, Remiddi 00; Henn 13; …

In a nutshell:

1. Derive a system of differential equations

$$\partial_x \vec{g}(x,\epsilon) = A(x,\epsilon)\vec{g}(x,\epsilon)$$

2. Find transformation $\vec{f}(x,\epsilon) = T(x,\epsilon)\vec{g}$ that factors out ϵ

$$\partial_x \vec{f}(x,\epsilon) = \epsilon \hat{A}(x)\vec{f}(x,\epsilon)$$

Henn 13

3. Make change of variables that rationalise square roots (if present), express result in *canonical* form:

$$d\vec{f}(x,\epsilon) = \epsilon(d\tilde{A})\vec{f}(x,\epsilon) \qquad \tilde{A} = \sum_k A_k \log \alpha_k(x)$$

$$\alpha_k = x - x_k$$

4. Write down solution in (generalized) multiple polylogarithms (GPLs)
5. Fix boundary conditions

Goncharov; Remiddi, Vermaseren; Brown; Poincaré; Euler; ...

A Brief Introduction to Computing Feynman Integrals

Amazingly successful - look at *nearly* any modern loop calculation

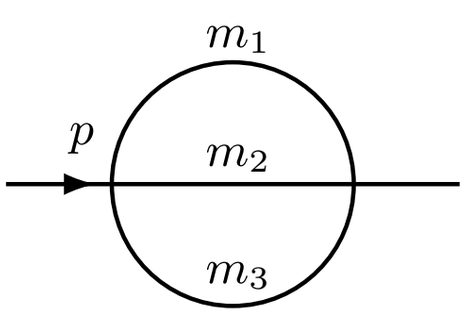
Importantly: the underlying mathematical and algebraic structures of MPLs/GPLs are well understood (symbol, co-product,…)

But: It can be difficult or impossible to perform these steps!

Early example: 2-loop sunset integral

Elliptic Integral

Sabry 62; Broadhurst 90



$$= 2\int_{(m_2+m_3)^2}^{\infty} \frac{\mathrm{d}x}{\sqrt{R_2(x,m_2^2,m_3^2)R_2(s,x,m_1^2)}} \log\left(\frac{x+m_1^2-s+\sqrt{R_2(s,x,m_1^2)}}{x+m_1^2-s-\sqrt{R_2(s,x,m_1^2)}}\right)$$

Recently, several promising developments in handling elliptic integrals:

- Connection to elliptic MPLs investigated (shuffle algebra, symbol,…)

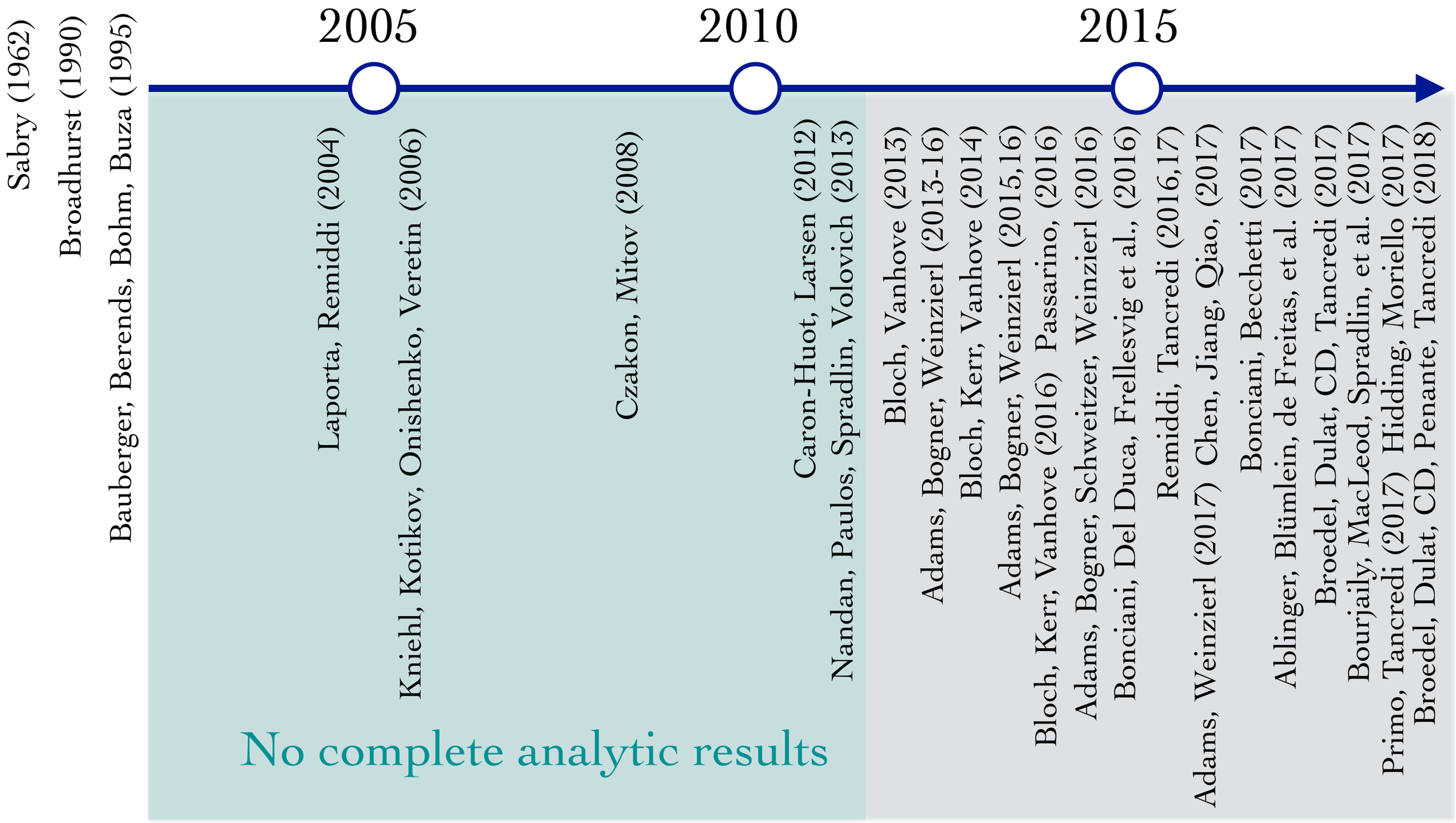
  Remiddi, Tancredi; Broedel, Duhr, Dulat, Penante, Tancredi;

- Related to iterated integrals of modular forms and Lambert–Eisenstein Series

  Adams, Bogner, Chaubey, Weinzierl; Ablinger, Blümlein, De Freitas, van Hoeij, Imamoglu, Raab, Radu, Schneider;

Incredible number of results/applications!

Elliptic Feynman integrals



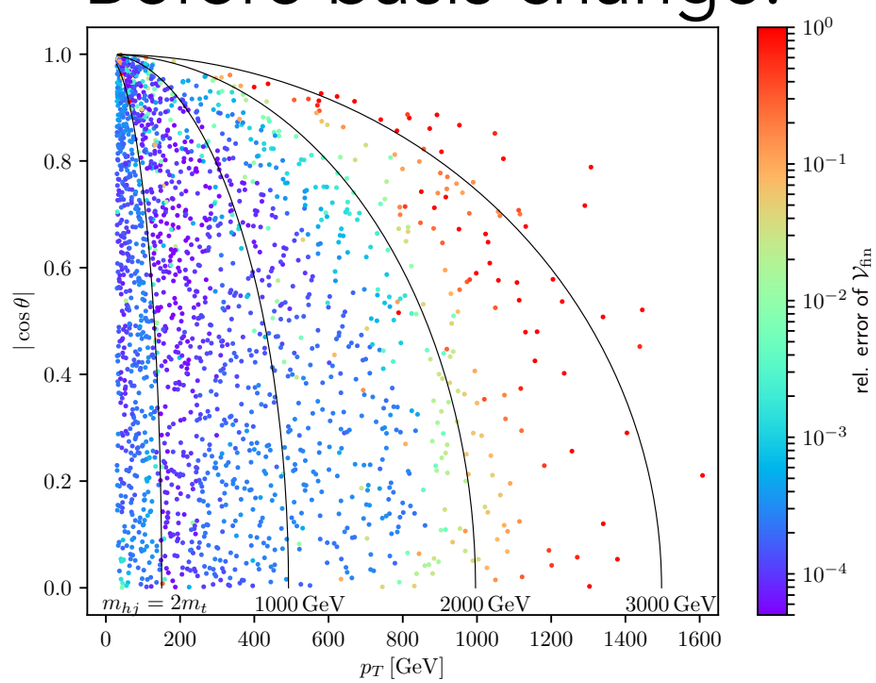
Slide: Claude Duhr (Loops & Legs, April 2018)

Higgs Boson + Jet Basis Change

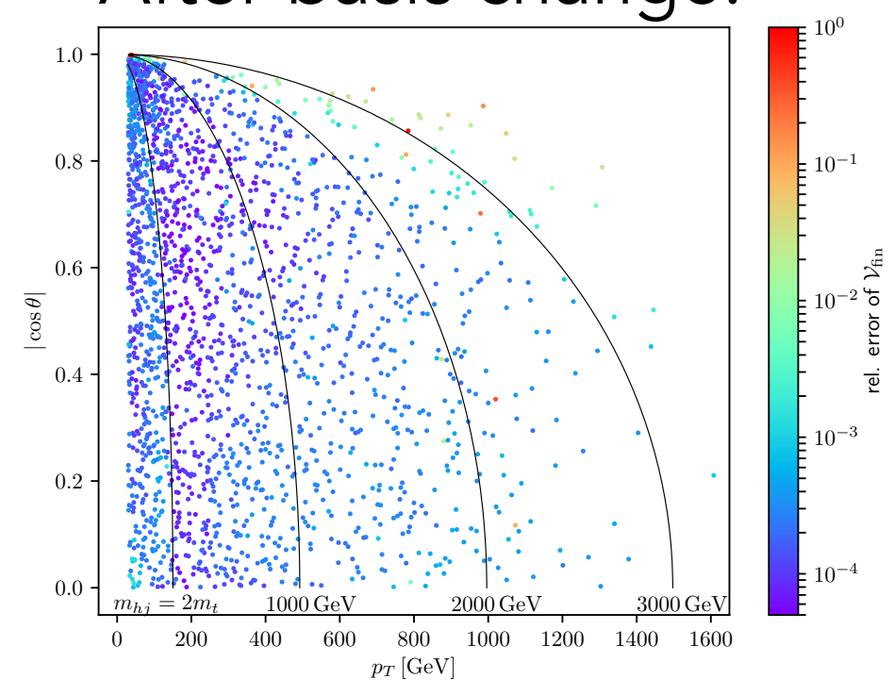
Since our paper we tried another integral basis (found by brute force):

- Quasi-finite
- Dimension factorises in denominator of coefficients
- “Simple” denominator factors and “relatively simple" numerators

Before basis change:



After basis change:



Phase-space points much more stable (esp. at large invariant mass)!

Coefficient code size: 340 MB → 100 MB

See: Matthias Kerner
Loops and Legs Proceedings 2018

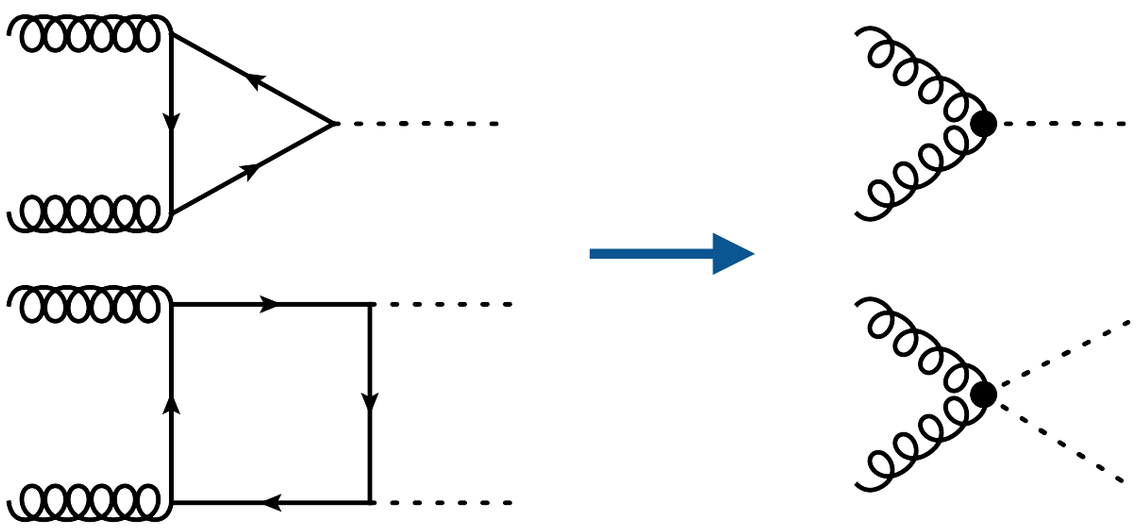
Can we automate finding this basis? Can we find an even better basis?

Heavy Top Limit

Heavy Top Limit (HTL): $m_T \to \infty$

Effective tree-level couplings between gluons and Higgs

Lowers number of loops by 1



HTL valid for

$$\sqrt{\hat{s}} \ll 2m_T$$

HH production for

$$2m_H < \sqrt{\hat{s}}$$

Small energy range in which HTL is technically justified

Born improved NLO HTL:

$$\mathrm{d}\sigma_{\mathrm{NLO}}(m_T) \approx \mathrm{d}\bar{\sigma}_{\mathrm{NLO}}(m_T) \equiv \underbrace{\frac{\mathrm{d}\sigma_{\mathrm{LO}}(m_T)}{\mathrm{d}\sigma_{\mathrm{LO}}(m_T \to \infty)}}_{\mathrm{N}} \mathrm{d}\sigma_{\mathrm{NLO}}(m_T \to \infty)$$

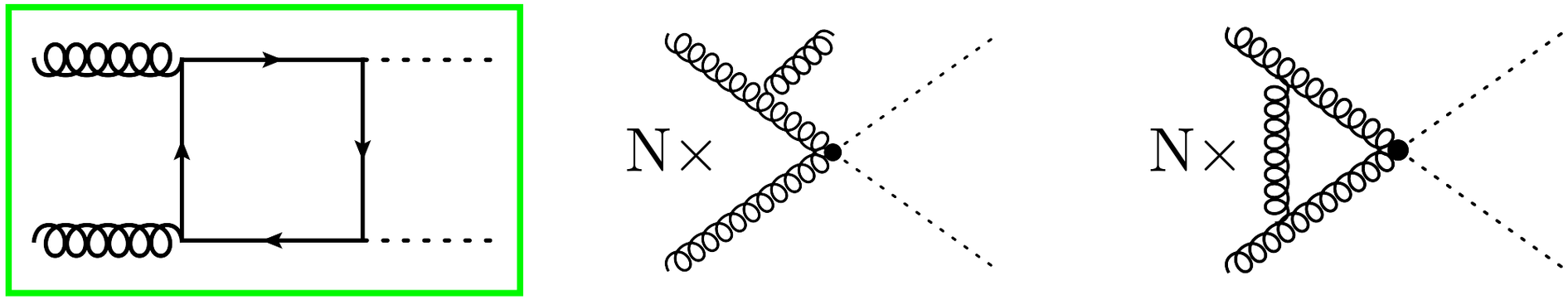
Spira et al. (HPAIR)

HH Approximations @ NLO (Schematically)

B R V

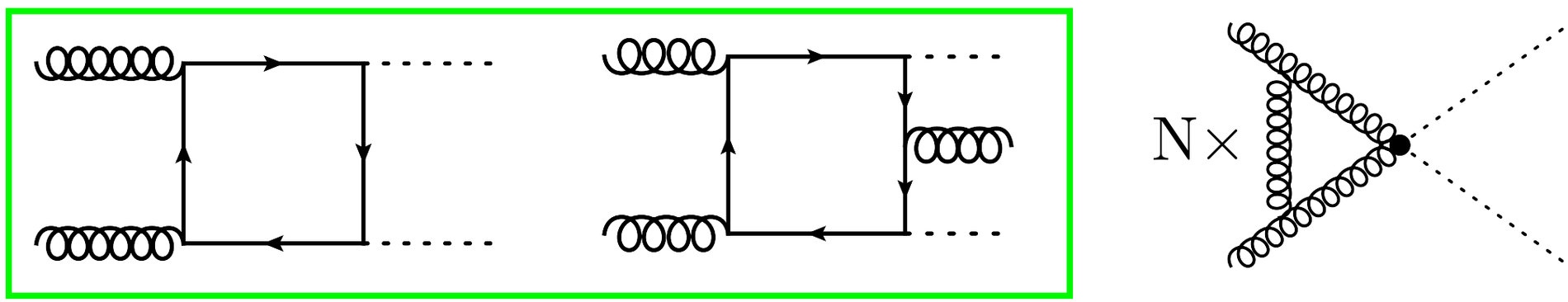
B-i HTL:

Dawson,
Dittmaier, Spira 98

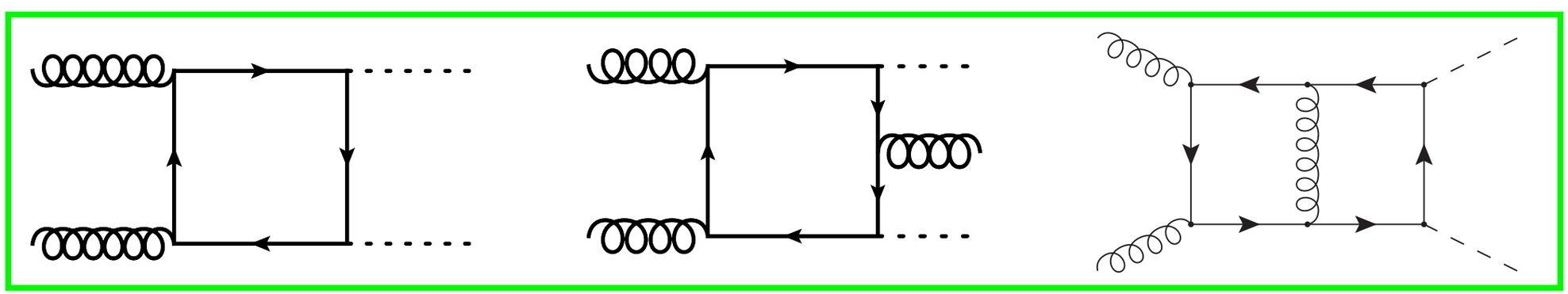


FTapprox:

Maltoni, Vryonidou,
Zaro 14



Full Theory:

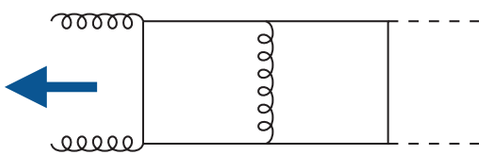


Borowka, Greiner, Heinrich, SPJ, Kerner, Schlenk, Schubert, Zirke 16;
Borowka, Greiner, Heinrich, SPJ, Kerner, Schlenk, Zirke 16;
Baglio, Campanario, Glaus, Mühlleitner, Spira, Streicher 18;

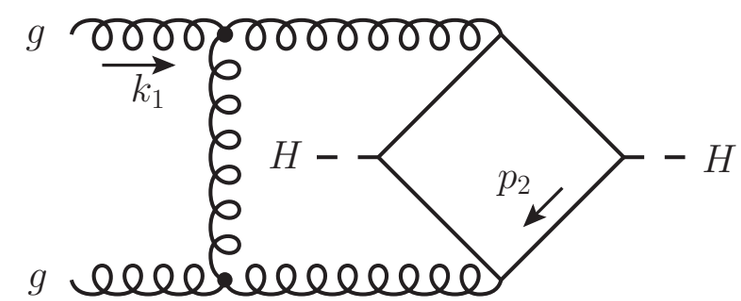
Amplitude Evaluation

Contributing integrals: $\sqrt{\hat{s}} = 327.25\,\mathrm{GeV}, \sqrt{-\hat{t}} = 170.05\,\mathrm{GeV}, M^2 = \hat{s}/4$

integral	value	error	time [s]
...			
F1_011111110_ord0	(0.484, 4.96e-05)	(4.40e-05, 4.23e-05)	11.8459
...			
N3_111111100_k1p2k2p2_ord0	(0.0929, -0.224)	(6.32e-05, 5.93e-05)	235.412
N3_111111100_1_ord0	(-0.0282, 0.179)	(8.01e-05, 9.18e-05)	265.896
N3_111111100_k1p2k1p2_ord0	(0.0245, 0.0888)	(5.06e-05, 5.31e-05)	282.794
N3_111111100_k1p2_ord0	(-0.00692, -0.108)	(3.05e-05, 3.05e-05)	433.342



$$I(s,t,m_t^2,m_h^2) = -\left(\frac{\mu^2}{M^2}\right)^{2\varepsilon}\Gamma(3+2\epsilon)M^{-4}\left(\frac{A_{-2}}{\epsilon^2}+\frac{A_{-1}}{\epsilon^1}+A_0+\mathcal{O}(\epsilon)\right)$$



Sector Decomposition

sector	integral value	error	time [s]	#points
5	(-1.34e-03, 2.00e-07)	(2.38e-07, 2.69e-07)	0.255	1310420
6	(-1.58e-03, -9.23e-05)	(7.44e-07, 5.34e-07)	0.266	1310420
...				
41	(0.179, -0.856)	(1.10e-05, 1.22e-05)	29.484	79952820
42	(0.359, -1.308)	(1.40e-06, 1.58e-06)	80.24	211436900
44	(0.0752, -1.185)	(5.44e-07, 6.76e-07)	99.301	282904860

Slide: Matthias Kerner

hhgrid

Amplitude is slow to evaluate:

Accuracy goal: 3% for F_1, 5-20% for F_2 (depending on F_2/F_1)

GPU Time/PS point: 80 min - 2 days (median 2 hours)

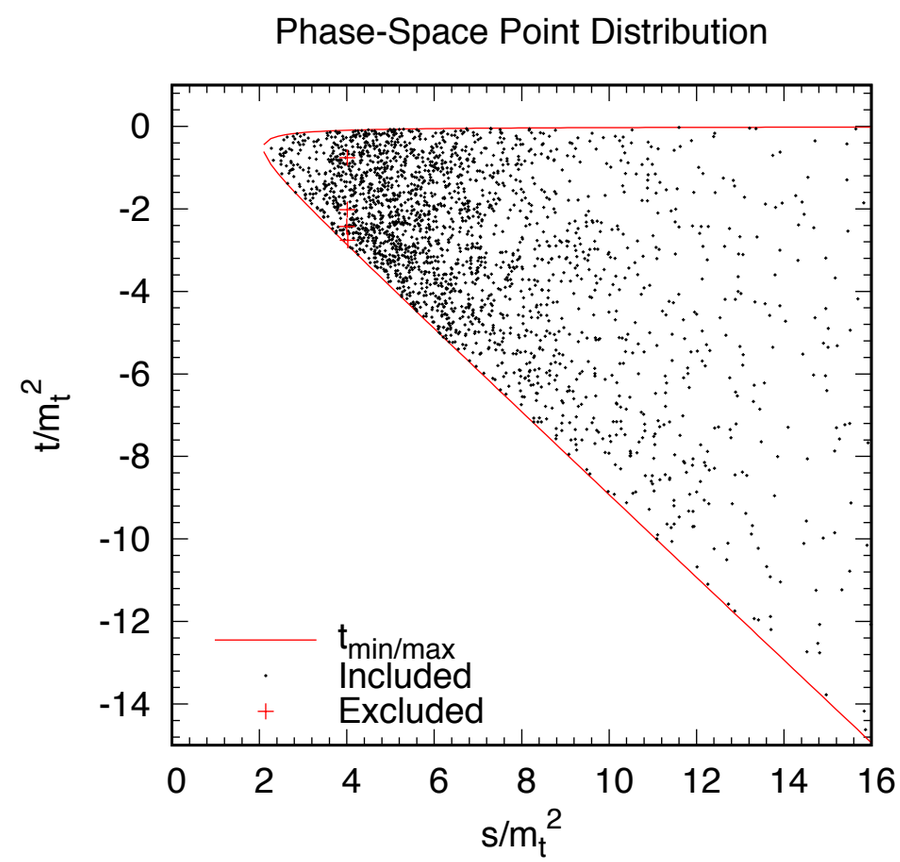
If a theoretical calculation is done, but it can not be used by any experimentalists, does it make a sound? - J. Huston

Constructed a grid in Mandelstam s, t

Based on 3746 phase-space points

Interfaced to POWHEG, MG5_amc@NLO

Sherpa



Heinrich, SPJ, Kerner, Luisoni, Vryonidou 17; SPJ, Kuttimalai 18

https://github.com/mppmu/hhgrid

Grid Details

Matrix element takes hours per phase-space point
Can not put directly into a Monte Carlo
But: Virtual matrix element depends only on $\hat{s}, \hat{t}$ (fixed m_T, m_H)
Can build 2D grid of our phase-space points and interpolate

Parametrisation

$$x = f(\beta(s)), \quad c_\theta = |\cos\theta| = \left|\frac{s + 2t - 2m_H^2}{s\beta(s)}\right|, \quad \beta = \left(1 - \frac{4m_H^2}{s}\right)^{\frac{1}{2}}$$

Choose $f(\beta)$ according to cumulative distribution function of phase space points used in the original calculation
Obtain nearly uniform distribution in (x, c_θ) unit square

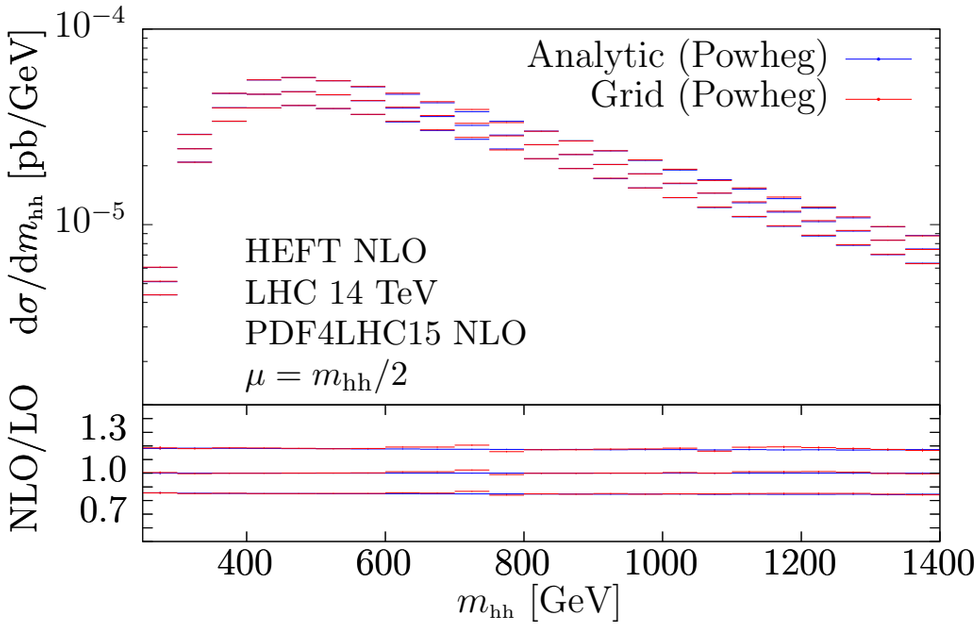
Interpolate with Clough-Tocher using the SciPy package

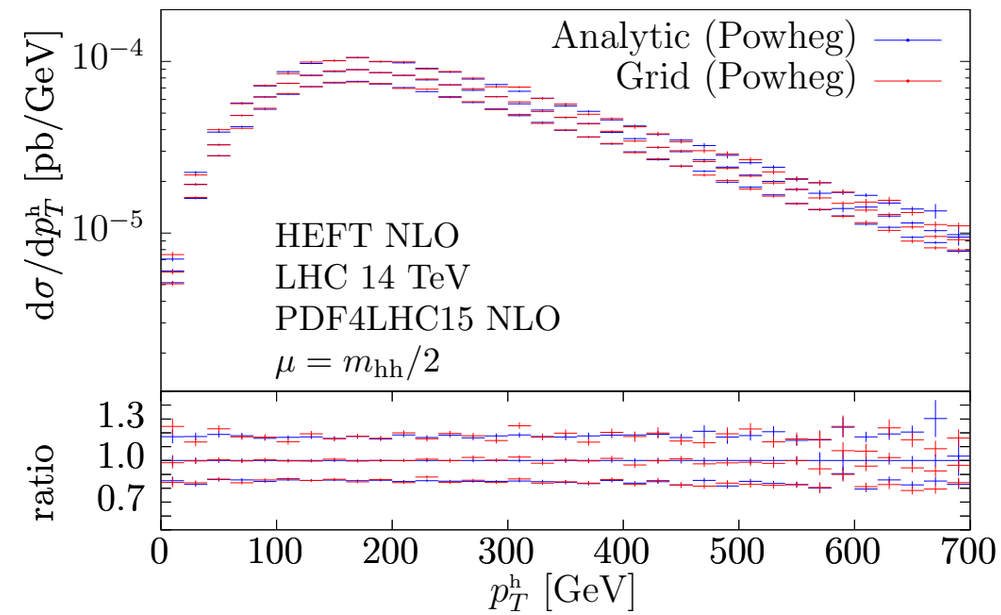
Clough, Tocher 65

2402+1344 events used for POWHEG/MG5_amc@NLO

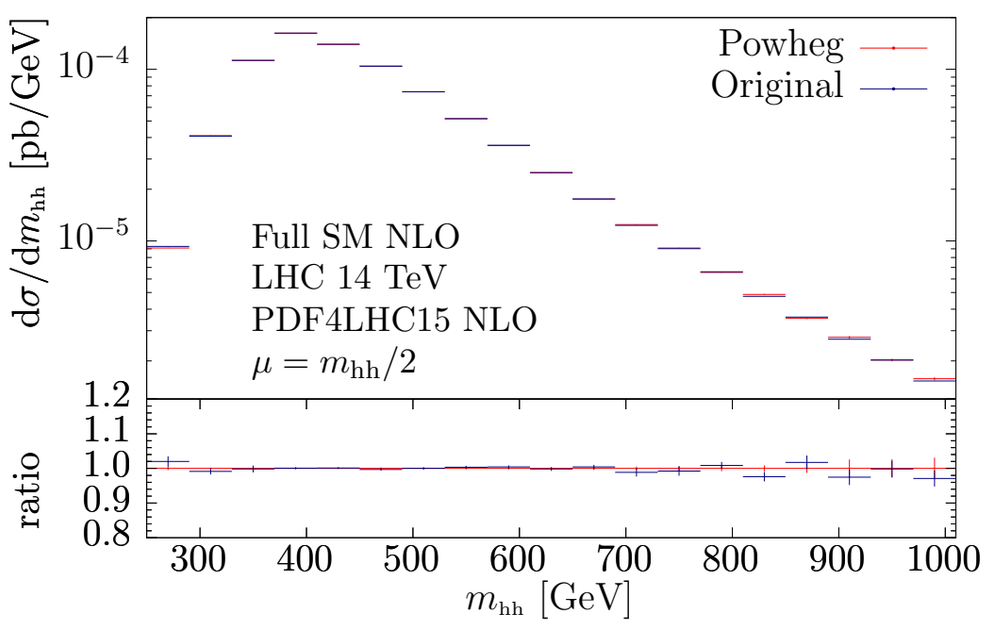
Grid Validation

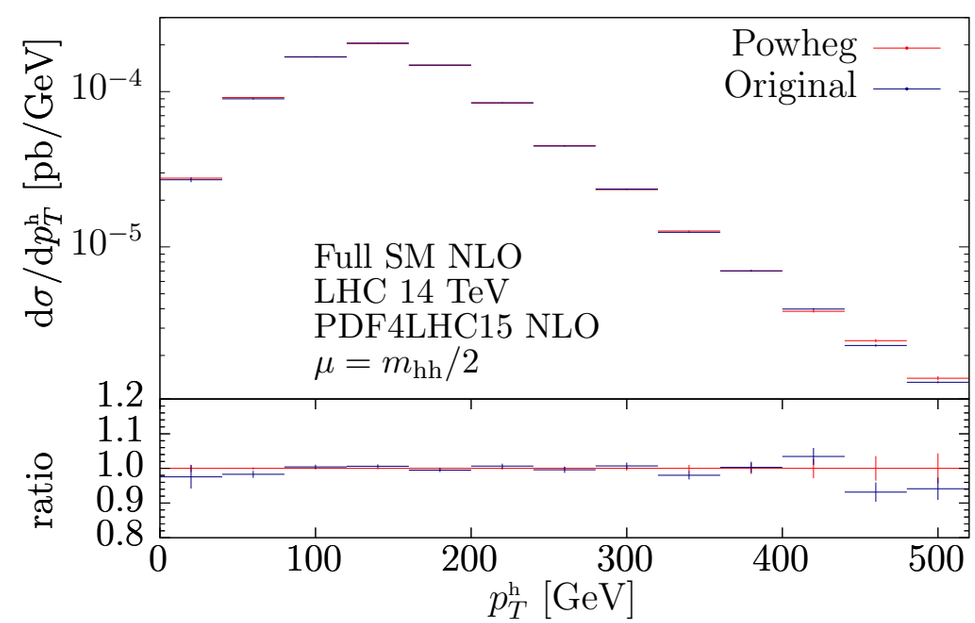
Use HEFT to study validity of grid





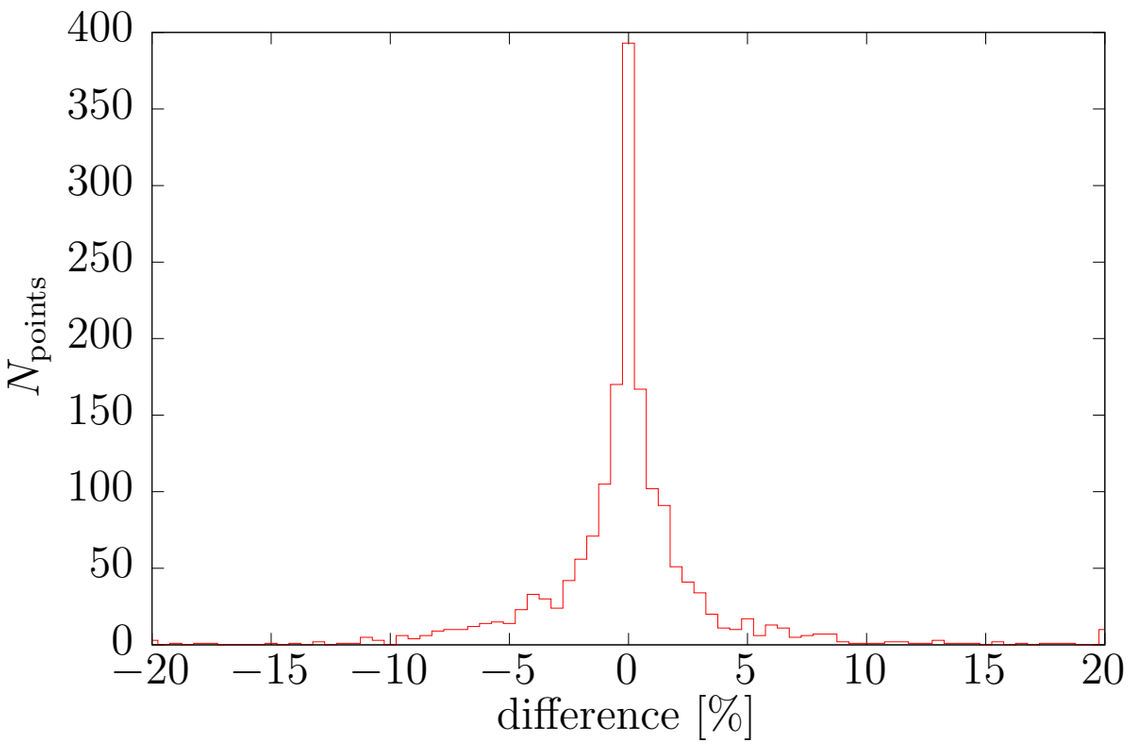
Full SM compare POWHEG (grid) with our original results



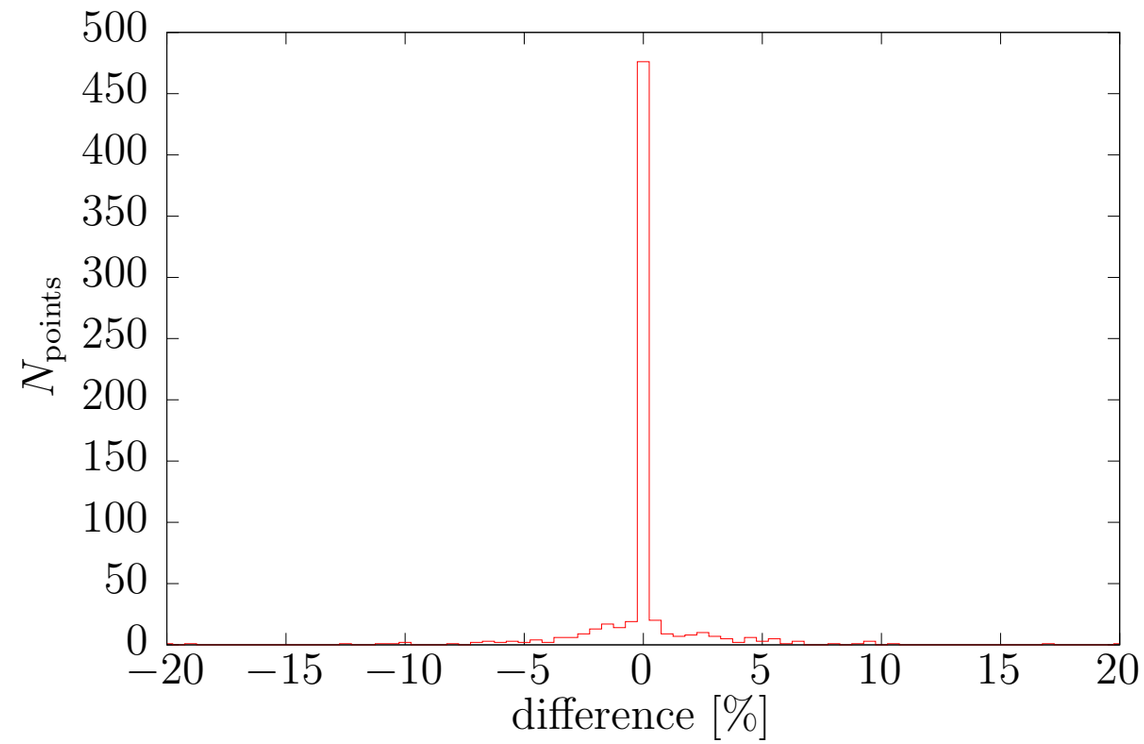


Heinrich, SPJ, Kerner, Luisoni, Vryonidou 17

Grid Stability



Grid vs grid with 50% of points



Grid vs grid with 80% of points

HH Checks

Real Emission / Subtraction Terms

- Independence of dipole-cut $\alpha_{\rm cut}$ parameter Nagy 03
- Agreement with literature Maltoni, Vryonidou, Zaro 14
- Agreement with FKS (`POWHEG/MG5_amc@NLO`)
  Frixione, Kunszt, Signer 96; Nason 04; Frixione et al 07; Alioli et al. 10; J. Alwall et al. 14

Virtual Corrections

- Two calculations of amplitude up to reduction
- Amplitude result invariant under $t \leftrightarrow u$
- Pole cancellation
- Mass renormalization using two methods:
  counter-term insertion vs. calculating $\mathrm{d}\mathcal{M}^{\rm LO}/\mathrm{d}m_T^2$ numerically
- Agreement of contributions $gg \to H \to HH$ with `SusHi` Harlander, Liebler, Mantler 13,16
- Convergence of $1/m_T$ expansion to full result
  where agreement is expected

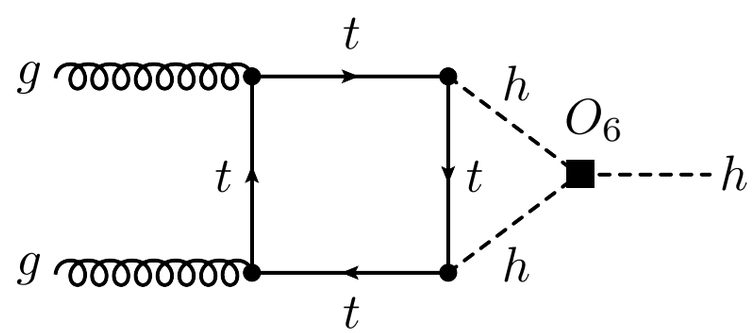
Data & Constraints

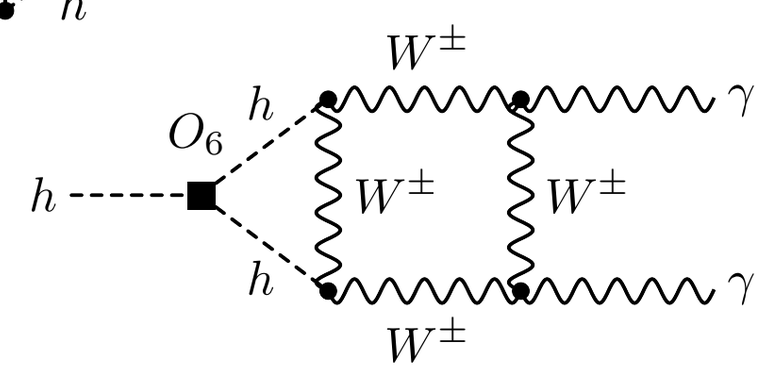
HH extremely challenging to measure, combining $b\bar{b}b\bar{b}, b\bar{b}\tau^+\tau^-, b\bar{b}\gamma\gamma$

$\leq 6.7\ \sigma_{\rm SM}$ ATLAS-CONF-2018-043

Several other promising ideas to obtain limits on λ_3:

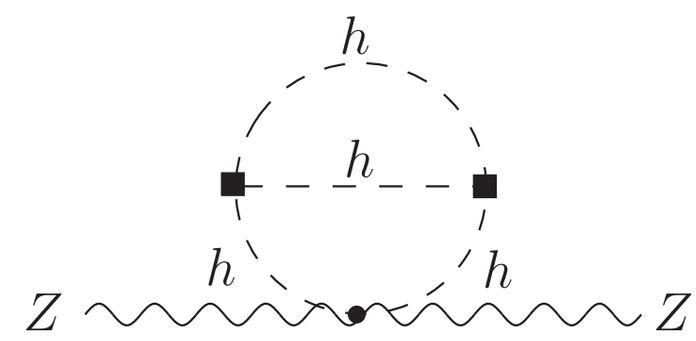
Electroweak corrections to single H production (also VBF, VH)





Gorbahn, Haisch 16; Bizoń, Gorbahn, Haisch, Zanderighi 16; Degrassi, Giardino, Maltoni, Pagani 16; Maltoni, Pagani, Shivaji, Zhao 17; Di Vita, Grojean, Panico, Riembau, Vantalon 17

Modification of precision EW observables (EW oblique corrections) S, T

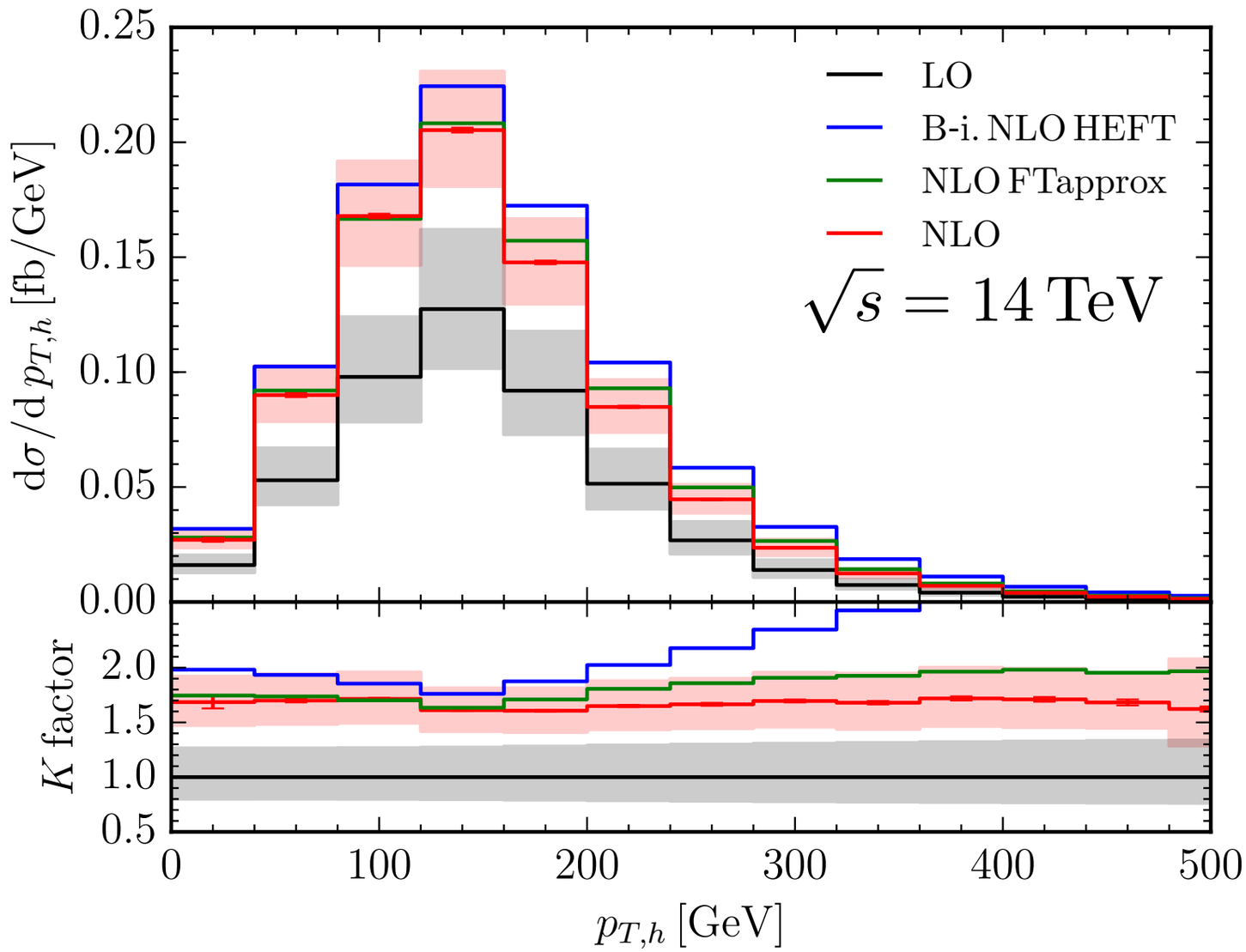


Degrassi, Fedele, Giardino 17; Kribs, Maier, Rzehak, Spannowsky, Waite 17;

Limits on λ_4: from (partial) EW corrections to HH

Bizon, Haisch, Rottoli 18; Borowka, Duhr, Maltoni, Pagani, Shivaji, Zhao 18

Higgs Boson Pair Results



HEFT: Can be poor approx. for larger $p_{T,h}$

Note: Ambiguous how to rescale HEFT real radiation by full LO born differentially

FTapp: Significantly better but still overestimating

Real radiation plays larger role for large $p_{T,h}$

Including m_T in real radiation does improve over HEFT in tails

Finite Basis

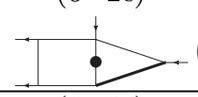
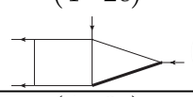
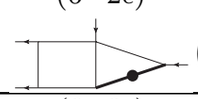
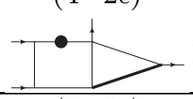
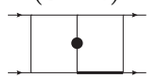
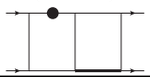
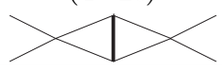
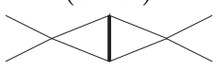
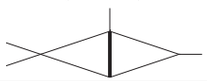
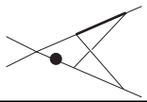
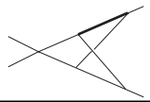
Always possible to pick finite basis of integrals, rewrite integrals using:

- Dimension Shifts Tarasov 96; Lee 10
- Dots

Panzer 14; von Manteuffel, Panzer, Schabinger 15

Two-loop EW-QCD Drell-Yan

von Manteuffel, Schabinger 17

Finite Basis...			Conventional...		
$(6-2\epsilon)$  (s,t)	201 s	2.34×10^{-4}	$(4-2\epsilon)$  (s,t)	384 s	8.12×10^{-4}
$(6-2\epsilon)$  (s,t)	150 s	4.83×10^{-4}	$(4-2\epsilon)$  (s,t)	56538 s	1.67×10^{-2}
$(6-2\epsilon)$  (s,t)	280 s	1.00×10^{-3}	$(4-2\epsilon)$  (s,t)	214135 s	8.29×10^{-3}
$(6-2\epsilon)$  (s,t)	294 s	1.21×10^{-3}	$(4-2\epsilon)$  (s,t)	3484378 s	30.9
$(4-2\epsilon)$  (s)	91 s	3.76×10^{-4}	$(4-2\epsilon)$  (s)	87 s	3.76×10^{-4}
$(6-2\epsilon)$  (s)	17 s	5.15×10^{-4}	$(4-2\epsilon)$  (s)	20 s	1.95×10^{-4}
$(6-2\epsilon)$  (s)	119 s	2.32×10^{-3}	$(4-2\epsilon)$  (s)	118 s	2.12×10^{-3}
Total/Max:	3995 s	5.84×10^{-3}	Total/Max:	5136862 s	30.9

← **Rel. Err.**

Huge decrease in time to numerically integrate and relative error

Divergences

From the master formula, 3 possibilities for poles in ϵ to arise:

1. Overall $\Gamma(N_\nu - LD/2)$ diverges (single UV pole)
2. $\mathcal{U}(\vec{x})$ vanishes for some $x = 0$ and has negative exponent (UV sub-divergences)
3. $\mathcal{F}(\vec{x}, s_{ij})$ vanishes on the boundary and has negative exponent (IR divergences)

Outside the Euclidean region ($\forall s_{ij} < 0$) there is a further possibility:

4. $\mathcal{F}(\vec{x}, s_{ij})$ vanishes inside the integration region (May give: Landau singularity which is either a normal or anomalous threshold)

Not discussed here (can be handled by pySecDec)

See: Soper 00; Borowka 14

If only condition 1 leads to a divergence the integral is **Quasi-finite**