Virtual HET Seminar, Brookhaven National Laboratory, USA 10 Feb 2022

Computing Feynman Integrals Numerically & ZH Production via Gluon Fusion

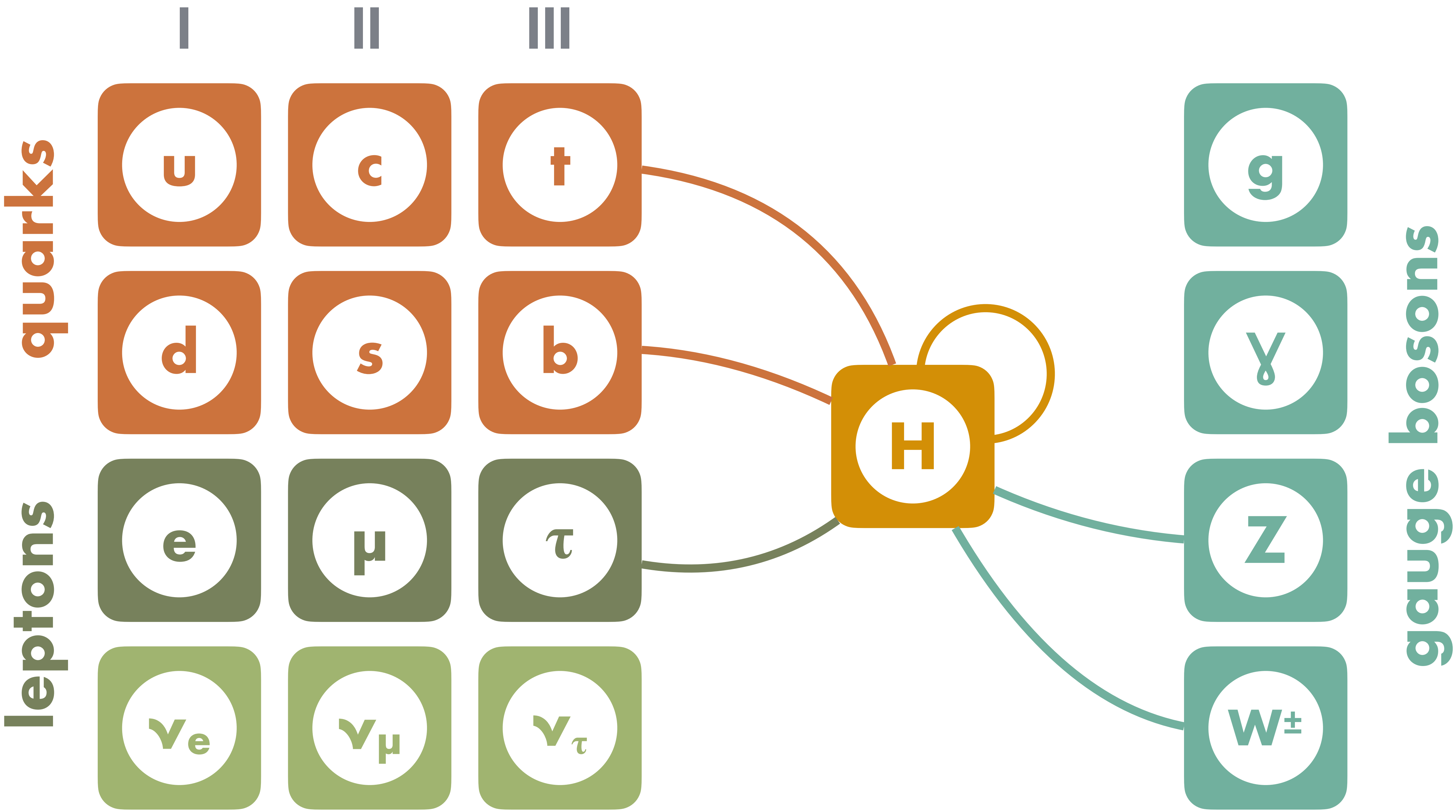
Stephen Jones

IPPP, Durham / Royal Society URF

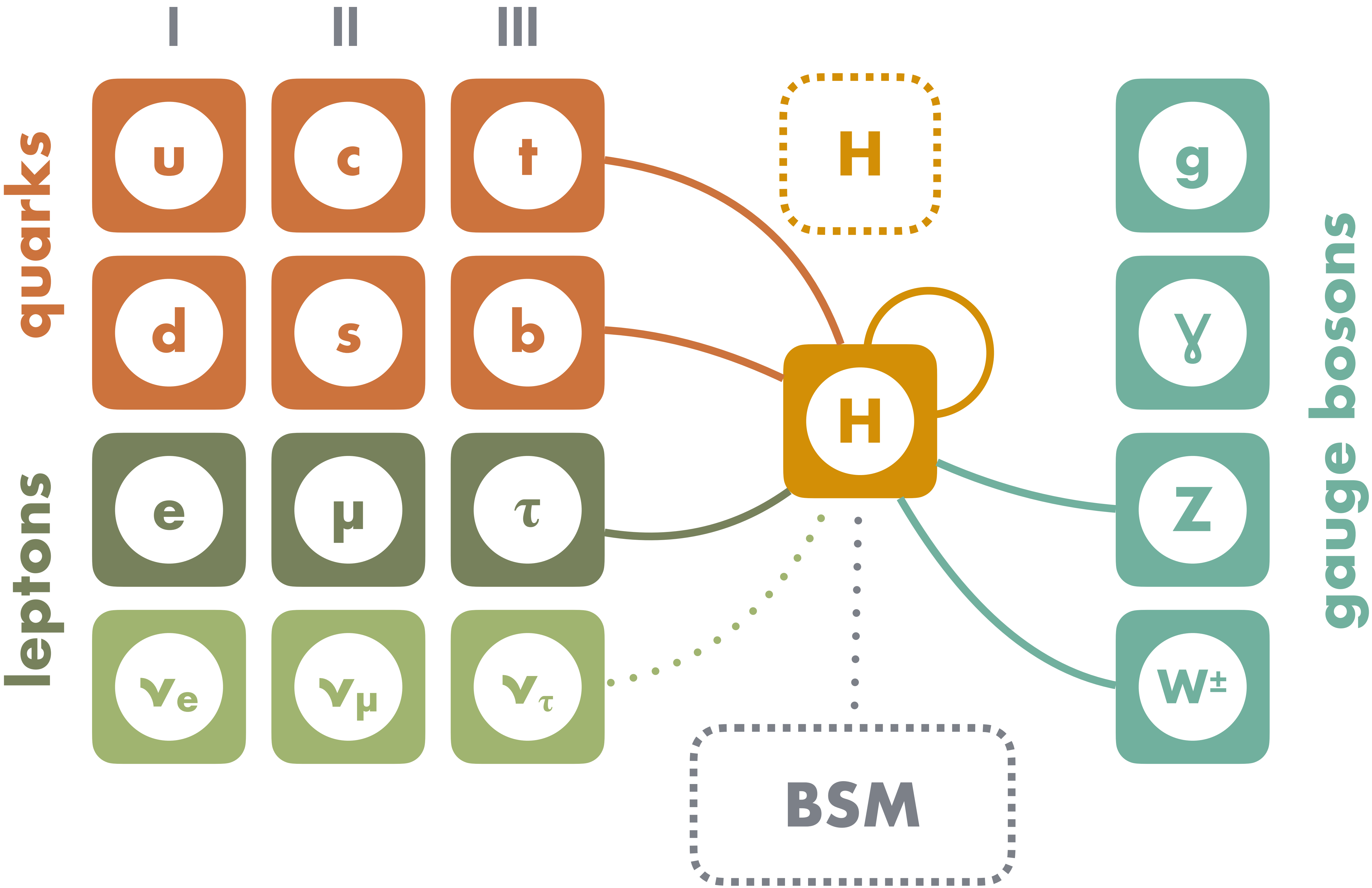
In collaboration with:
Chen, Heinrich, Kerner, Klappert, Schlenk
+ Jahn, Langer, Magerya, Põldaru, Villa
+ Davies, Mishima, Steinhauser

TBA [22XX.XXXX]
pySecDec Release [2108.10807]
JHEP 03 (2021) 125 [2011.12325]

The Standard Model

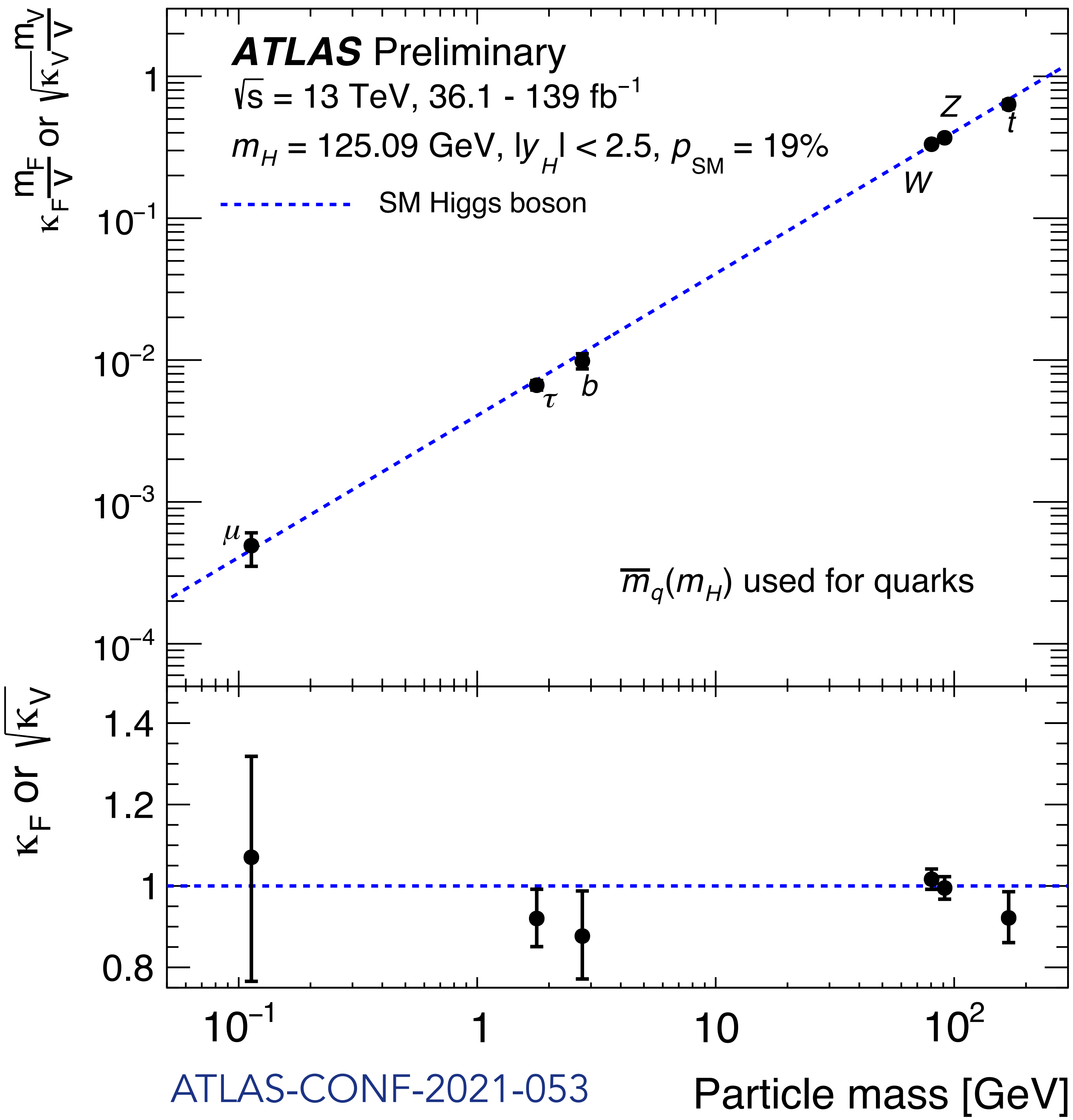


The Standard Model and Beyond

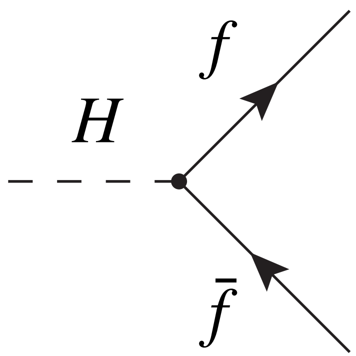


Higgs Couplings

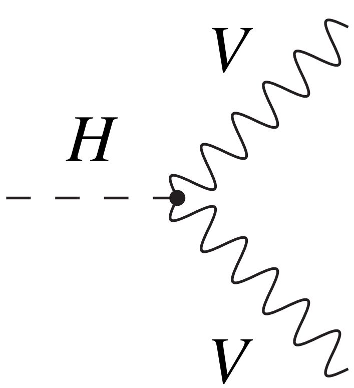
Incredible progress has been made in establishing properties of the Higgs Boson



ATLAS-CONF-2021-053



$$= -i\frac{m_f}{v}$$

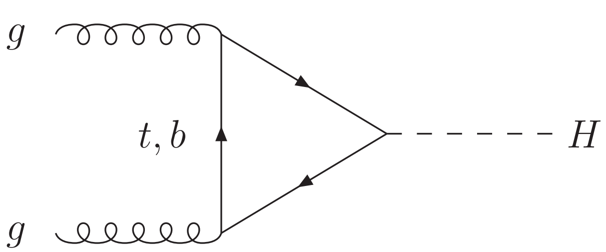


$$= 2i\frac{M_V^2}{v}g_{\mu\nu}$$

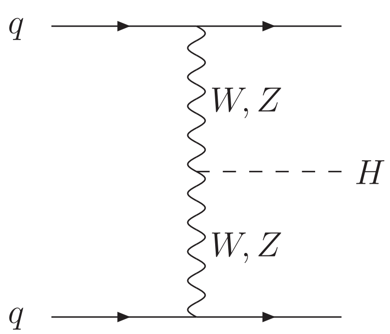
Higgs Production & Decay

σ [pb] @ 13 TeV

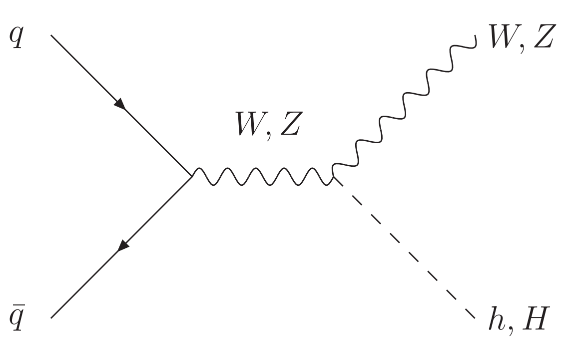
Higgs in 140 fb^{-1}



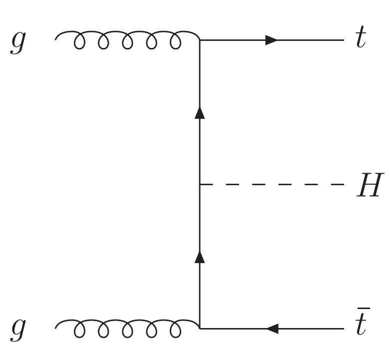
49 pb / 6.9M



3.8 pb / 520k

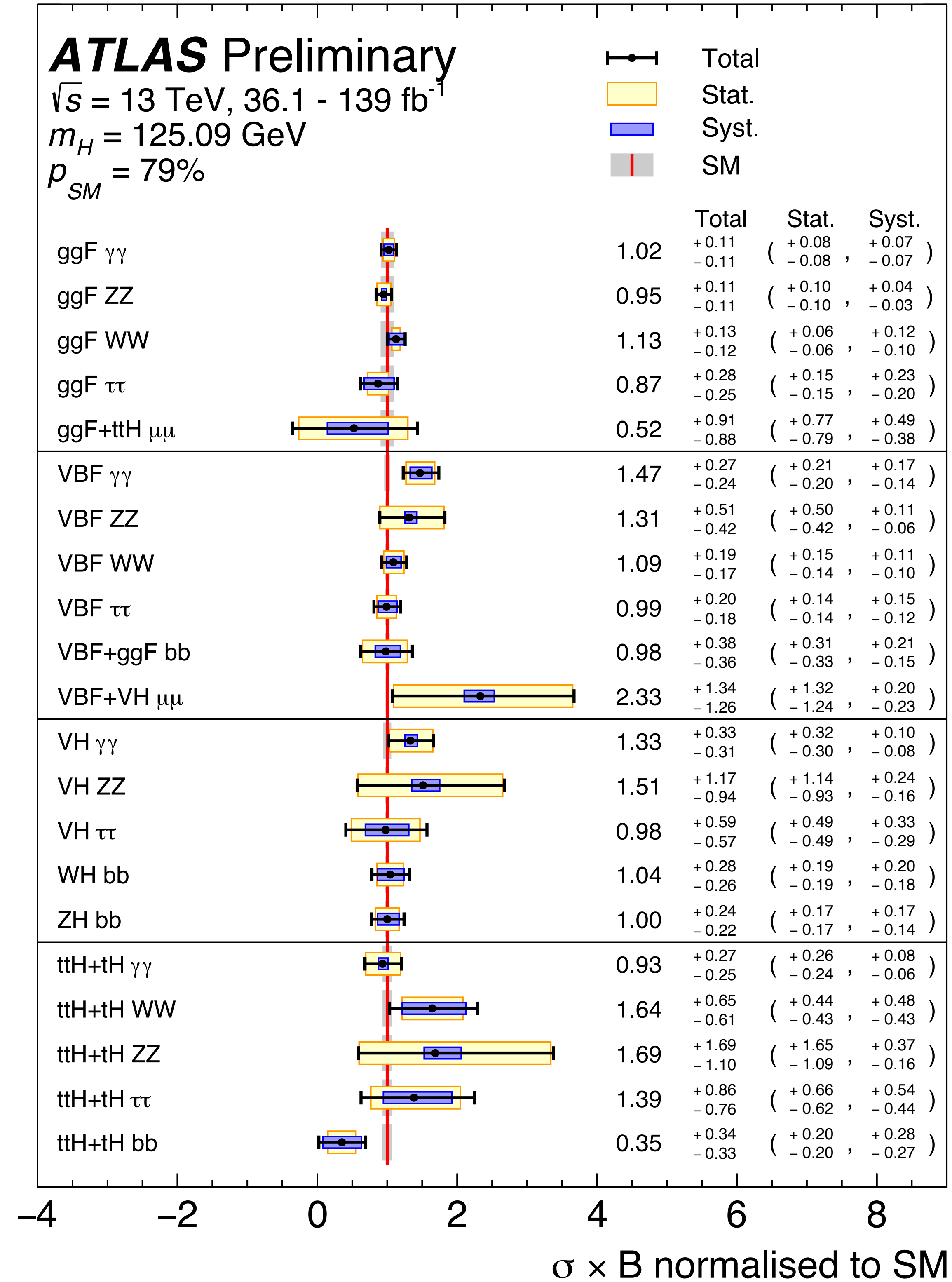


2.3 pb / 320k

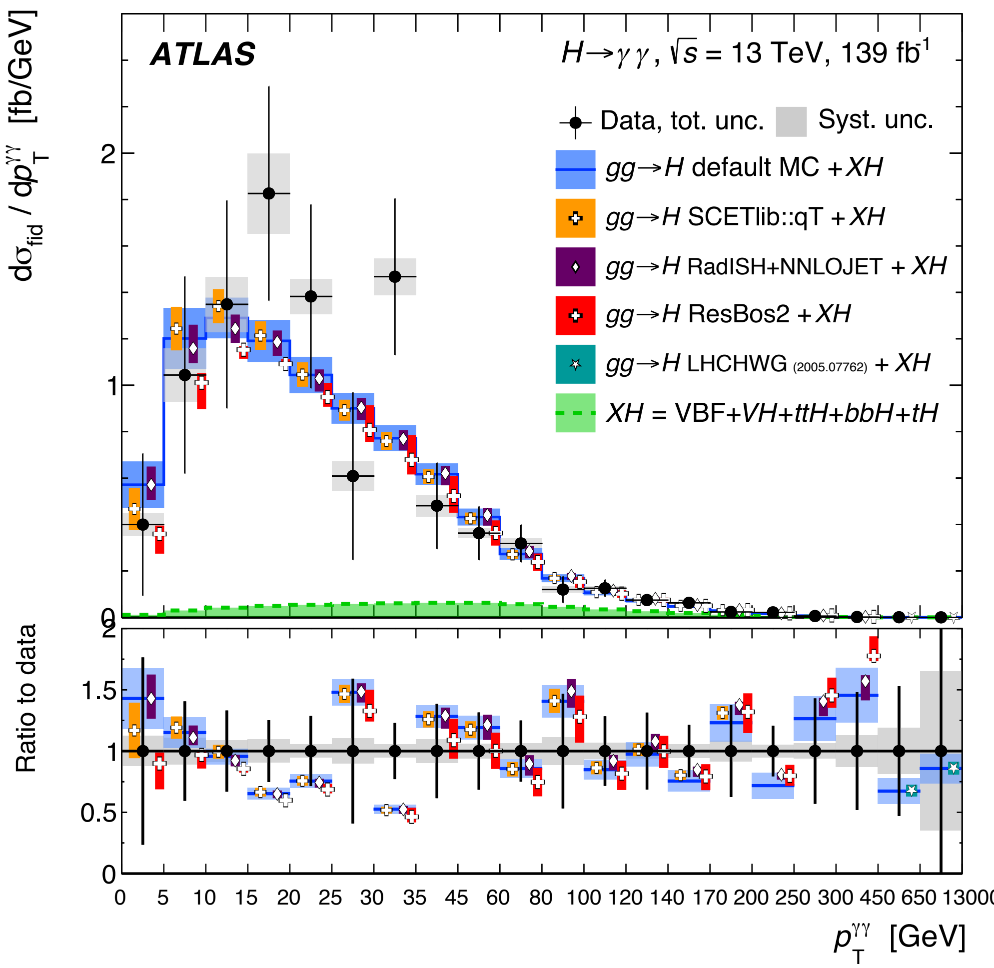


0.5 pb / 70k

ATLAS-CONF-2021-053



Experiments: Recent Progress



Measurement of $gg \to H(\to \gamma\gamma)$

$$\sigma_{\text{fid}}^{\text{exp}} = 67 \pm 7 \text{ fb}$$
$$\sigma_{\text{fid}}^{\text{th}} = 64 \pm 4 \text{ fb}$$

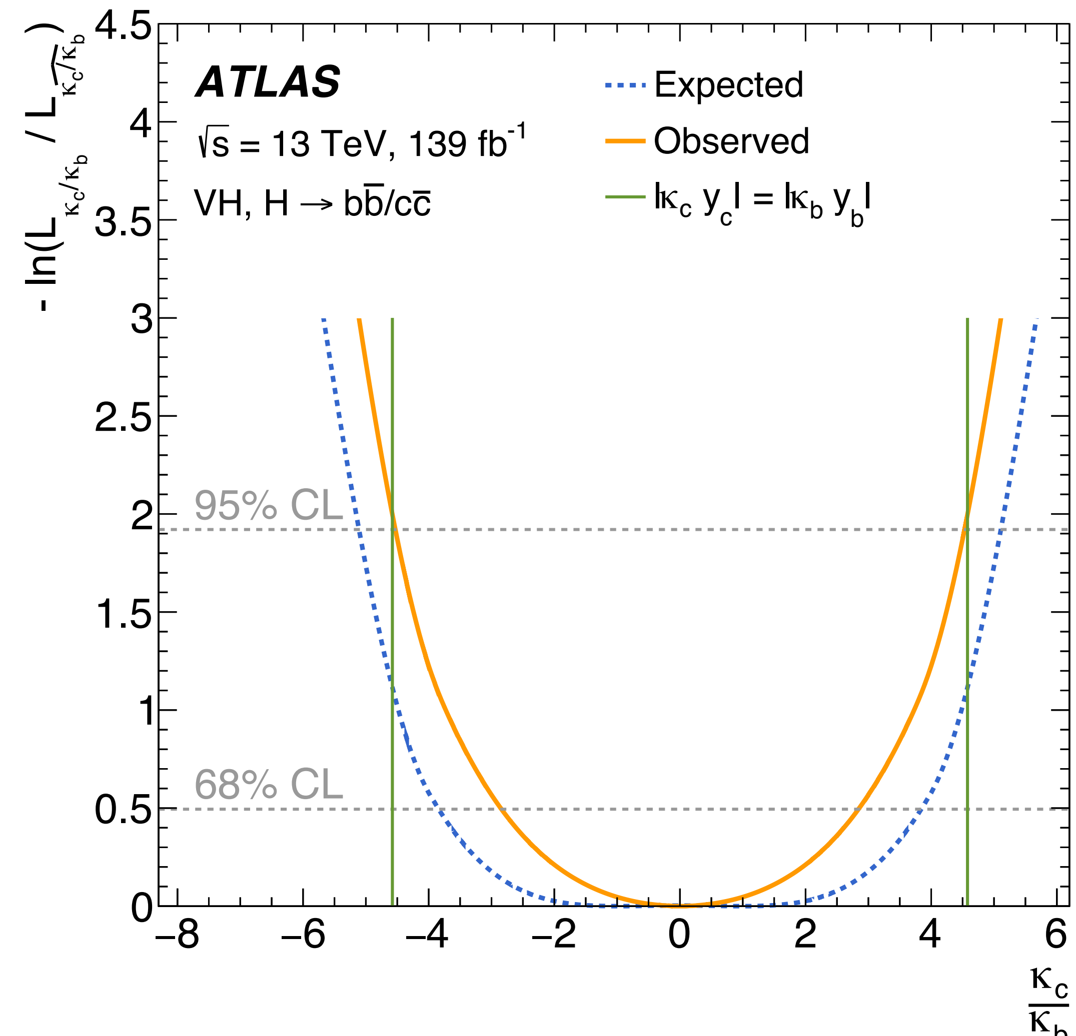
Also with differential fiducial cross sections (to high $p_T^{\gamma\gamma}$!) CERN-EP-2021-227

Direct constraint on y_b/y_c from $VH(\to c\bar{c})$

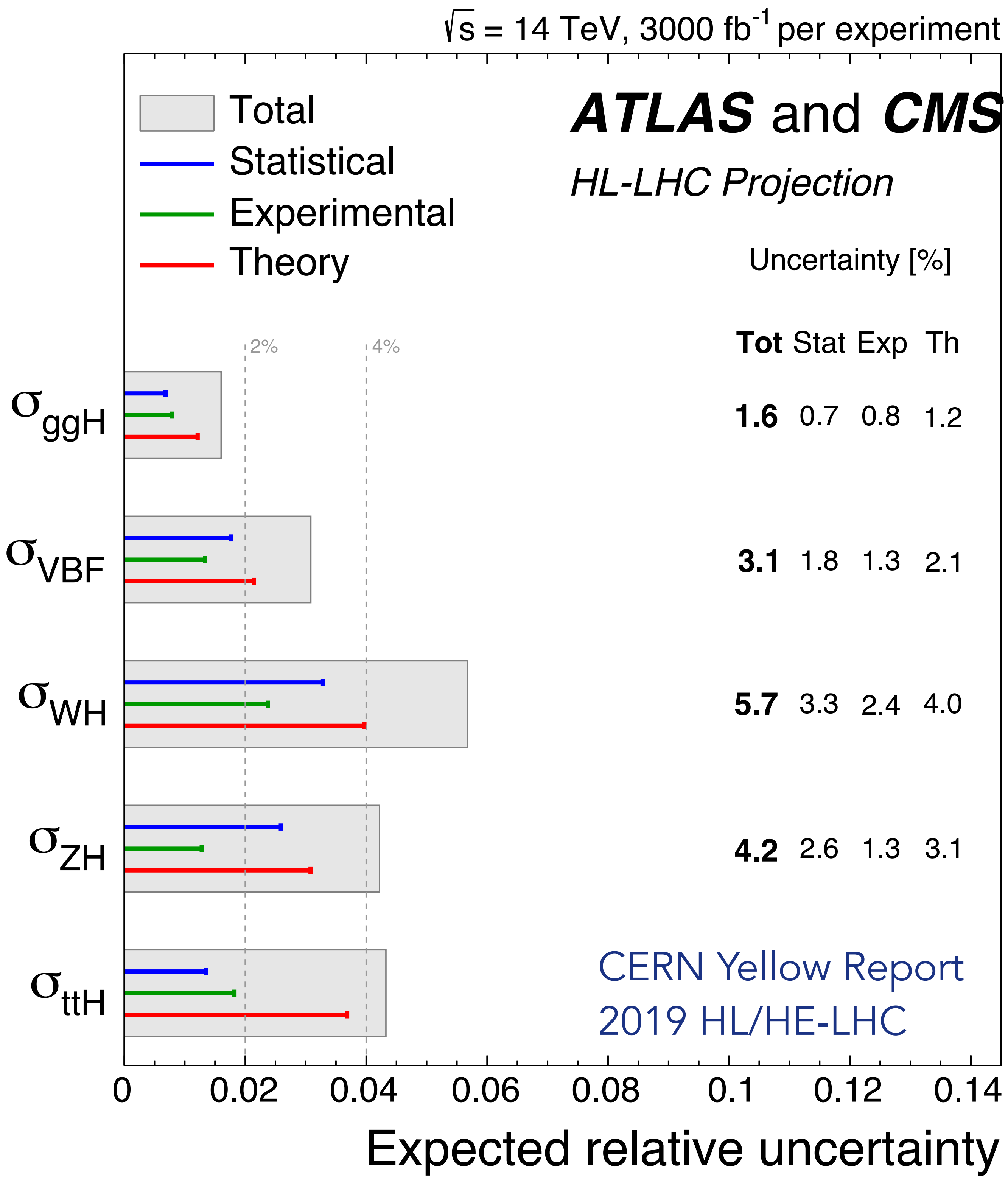
$|\kappa_b/\kappa_c| < 4.5 < m_b/m_c$ @ 95% CL

Result implies that the Higgs coupling to charm quarks is smaller than the coupling to bottom quarks (as predicted by SM)

CERN-EP-2021-251



Future Experiments: HL-LHC



HL-LHC construction underway

~10x integrated luminosity of LHC (LHC 0.3 ab^{-1}, HL-LHC: 3 ab^{-1})

Theory uncertainty is expected to dominate HL-LHC Higgs physics

Plot shown assumes reduction by factor 2 of today's uncertainties

Outline

Motivation & Background

Higgs precision, why corrections to $gg \to ZH$ are interesting

Numerical calculation of Feynman integrals

Numerically evaluating amplitudes with pySecDec

Expansion by regions

Virtual Results & Comparisons

Motivation: Higher Order Computations

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

Parton Distribution Functions (PDFs)

Hard Scattering Matrix Element

Non-perturbative effects ~ few %

With $\alpha_s \sim 0.1$, expect NLO ~ 10% correction, NNLO ~ 1% correction

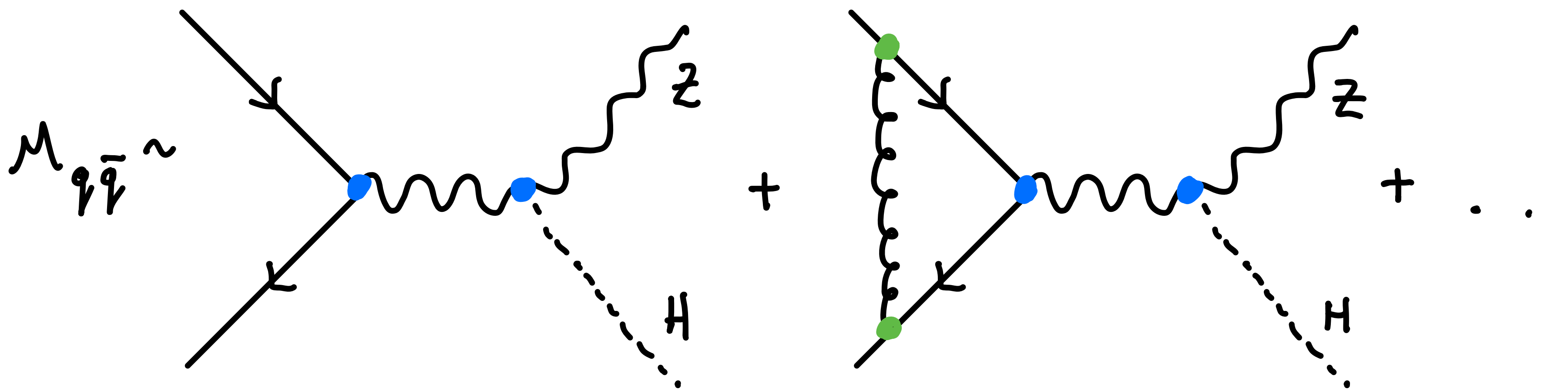
However, there are important exceptions:

- Higgs production (NLO ~100%, NNLO ~10%, N3LO ~ 2%)
- New partonic channels can open at higher-orders (e.g. $gg \to ZH$)
- Distributions can be modified substantially (even if σ_{tot} is stable)

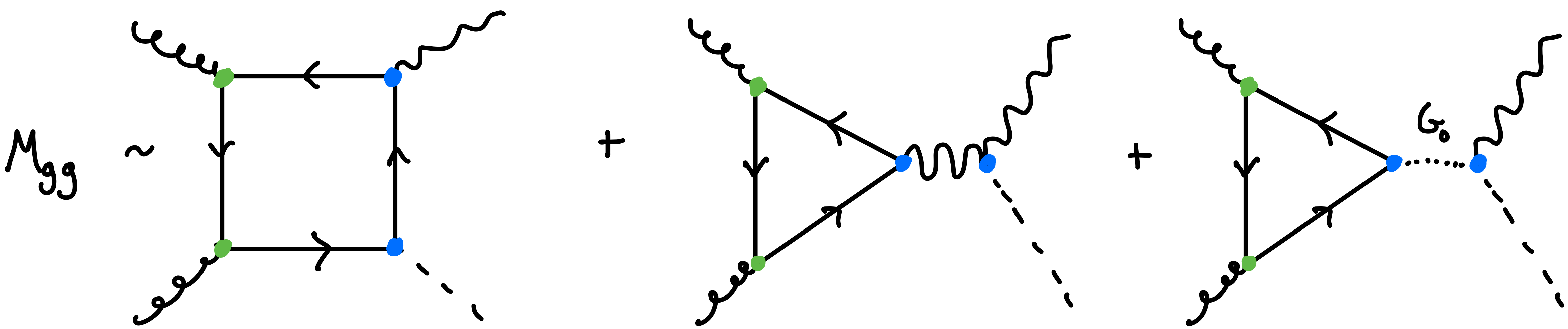
Overview of $pp \to ZH$ (I)

Consider the matrix element for $pp \to ZH$ with QCD corrections

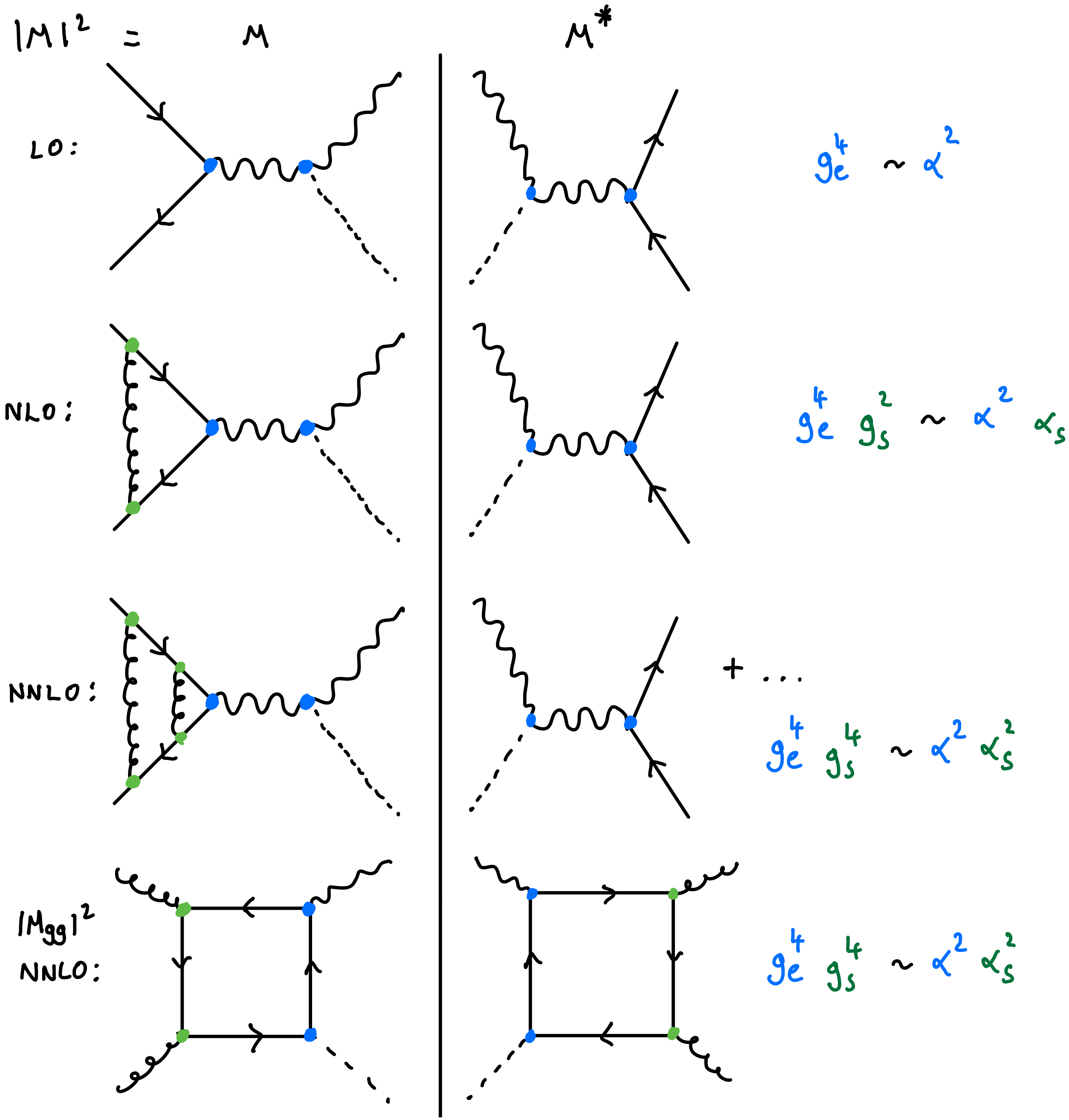
Various channels contribute: $q\bar{q} \to ZH$ and $gg \to ZH$



The $gg \to ZH$ channel is **loop-induced** (i.e. LO in this channel is 1-loop)



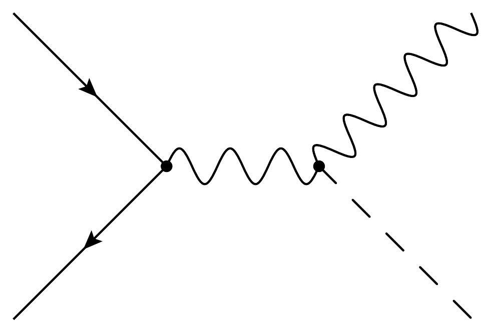
Overview of $pp \to ZH$ (II)

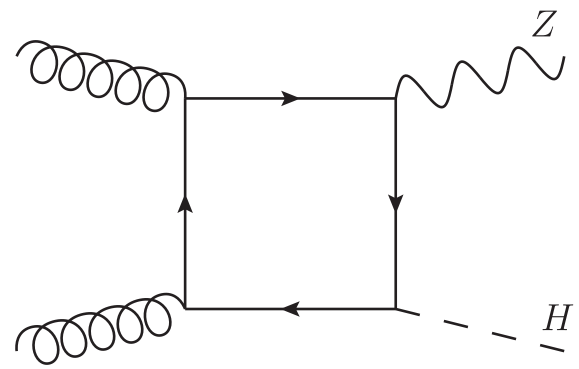


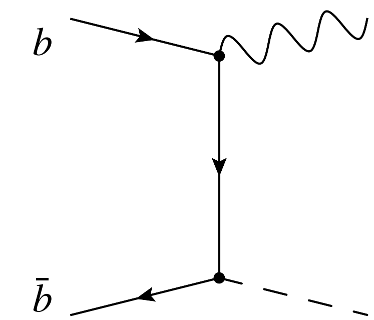
The $gg \to ZH$ channel contributes to $pp \to ZH$ starting at NNLO in QCD

However due the large gluon-gluon luminosity at the LHC it contributes significantly (~10%) to the total cross section

Overview of $pp \to ZH$ (III)







Drell-Yan piece (NNLO known)
Brein, Djouadi, Harlander 03;
Ferrera, Grazzini, Tramontano 14;
See also: Kumara, Mandal, Ravindran 14
+ $q\bar{q}$ piece with closed top loops (1-3%)

Gluon-fusion piece

$b\bar{b}$ piece (NNLO known)
Ahmed, Ajjath, Chen, Dhani, Mukherjee, Ravindran 19

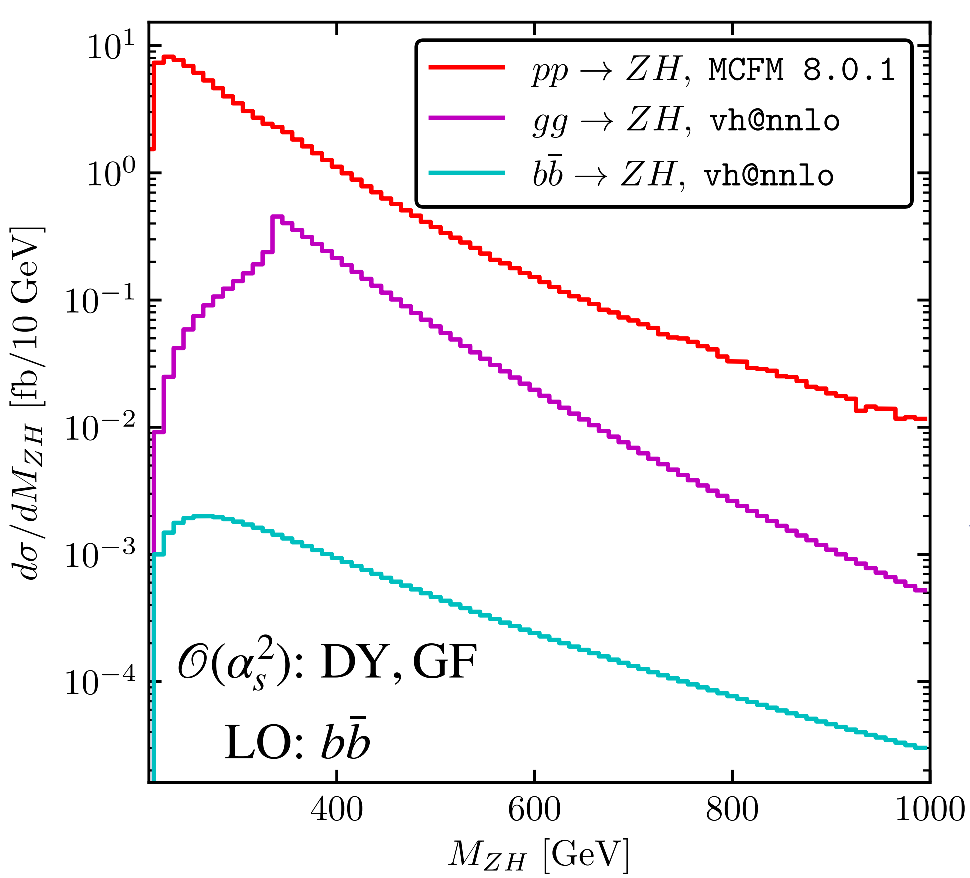
Available in various codes:

HAWK (NLO QCD + NLO EW)
Denner, Dittmaier, Kallweit, Mück 14

vh@nnlo (NNLO QCD + NLO EW)
Harlander, Klappert, Liebler, Simon 18;
Brein, Harlander, Zirke 12

MCFM (NNLO QCD)
Campbell, Ellis, Williams 16

GENEVA (NNLL′+NNLO with PS)
Alioli, Broggio, Kallweit, Lim, Rottoli 19



Harlander, Klappert, Liebler, Simon 18

Why Calculate $gg \to ZH$?

The practical reason …

~10% of total cross section

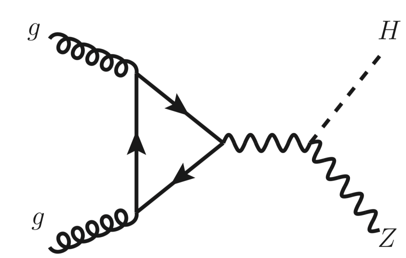
~100% scale uncertainty
(and underestimated?)

	Signal
Cross-section (scale)	0.7% (qq), $\boxed{25\%\ (gg)}$
$H \to b\bar{b}$ branching fraction	1.7%
Scale variations in STXS bins	3.0%–3.9% ($qq \to WH$), 6.7%–12% ($qq \to ZH$), $\boxed{37\%\text{–}100\%\ (gg \to ZH)}$
PS/UE variations in STXS bins	1%–5% for $qq \to VH$, 5%–20% for $gg \to ZH$
PDF+α_S variations in STXS bins	1.8%–2.2% ($qq \to WH$), 1.4%–1.7% ($qq \to ZH$), 2.9%–3.3% ($gg \to ZH$)
m_{bb} from scale variations	M+S ($qq \to VH$, $gg \to ZH$)
m_{bb} from PS/UE variations	M+S
m_{bb} from PDF+α_S variations	M+S
p_T^V from NLO EW correction	M+S

ATLAS 2007.02873

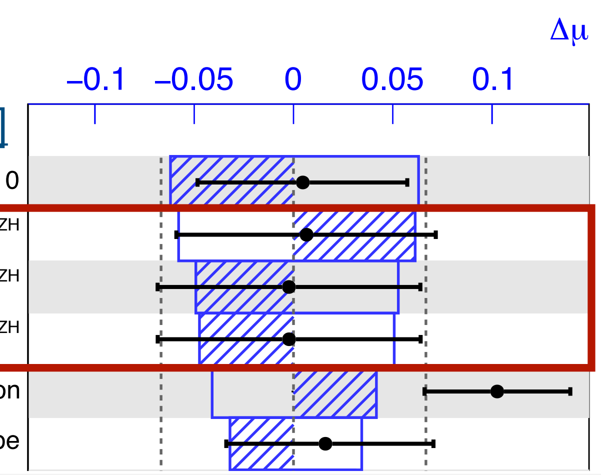
ATLAS and CMS use ggZH@LO(QCD) from POWHEG
[inclusive calculation: NLO + NLL (QCD)]

$\rightarrow$ *Large scale uncertainties*



ggZH accounts only for ~10% of inclusive ZH cross section
Uncertainties still very important!

ATLAS ZH
[2007.02873]



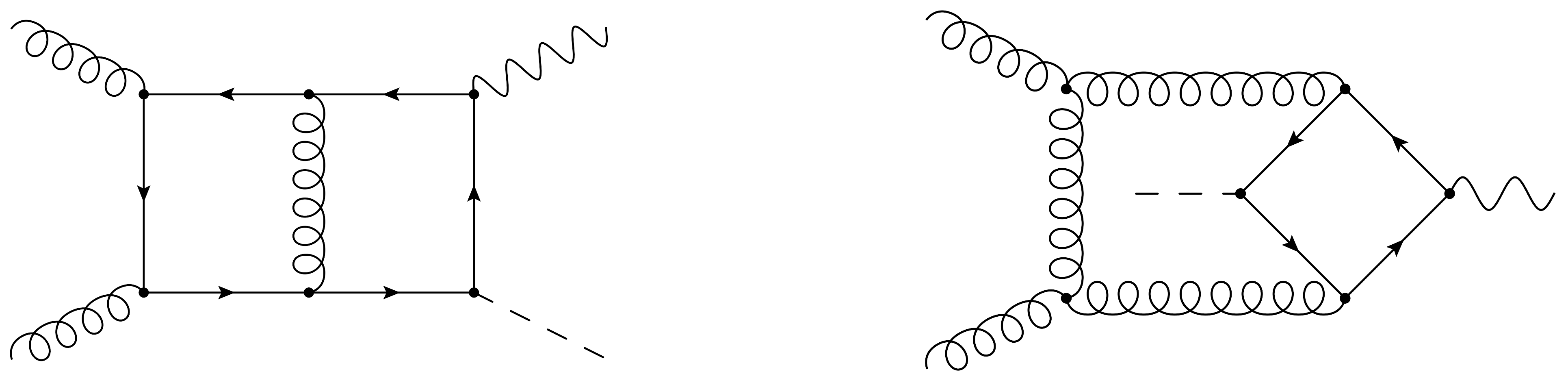
No full NLO calculation available for the foreseeable future (?)

How to improve?

Philipp Windischhofer / ATLAS (LHCXSWG Meeting 9/11/2020)

Why Calculate $gg \to ZH$? (II)

And another reason…



Provides new and interesting challenges:

Amplitude depends on large number of scales $s, t, m_Z^2, m_H^2, m_T^2$

Feynman Integrals appearing are non-trivial (internal masses, elliptic…)

Can test our techniques to breaking point then develop new approaches!

Can we find a basis of integrals with simple coefficients?

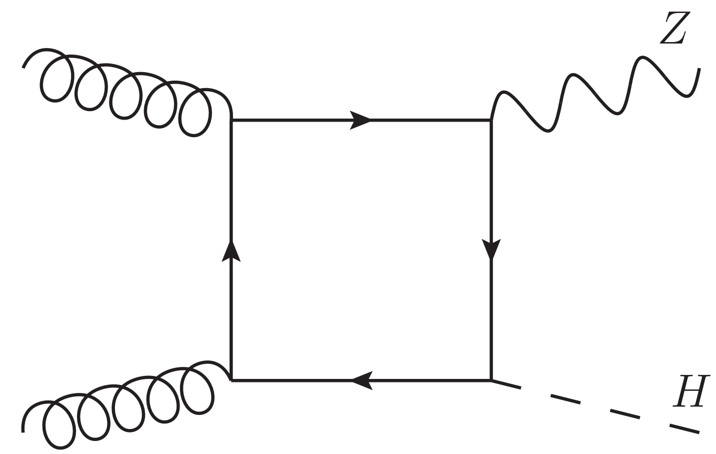
How can we obtain a reduction to a finite basis (many dots/numerators)?

Can we improve numerical performance near thresholds? …

ZH in Gluon Fusion

Full leading order (loop induced)
Dicus, Kao 88; Kniehl 90

NLO in the limit of $m_t \to \infty$ $(K \approx 2)$
(asymptotic expansion)
Altenkamp, Dittmaier, Harlander, H. Rzehak, Zirke 12

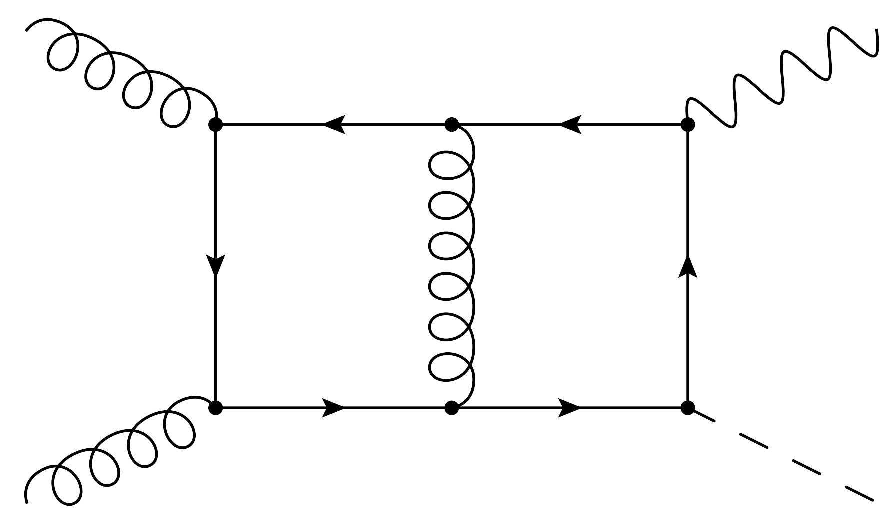


Virtual Corrections:

Expansion around large top quark mass
($1/m_t^8$) + Padé approx
Hasselhuhn, Luthe, Steinhauser 17

Expansion around small top quark mass
($1/m_t^{10}$ & m_t^{32}) + Padé approx
Davies, Mishima, Steinhauser 20

Expansion around small p_T up to p_T^4
Alasfar, Degrassi, Giardino, Gröber, Vitti 21



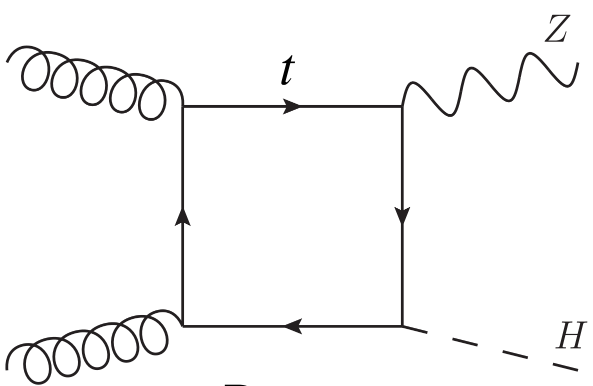
Full numerical result
Chen, Heinrich, SPJ, Kerner, Klappert, Schlenk 20

NLO result: Expansion around small m_z, m_h
Wang, Xu, Xu, Yang 21 (2107.08206)

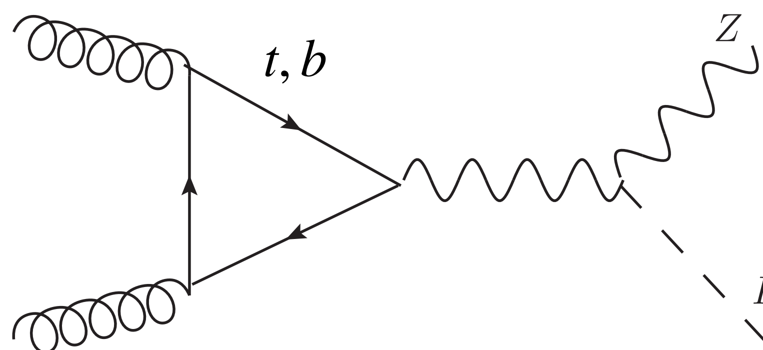
Setup & Amplitudes

Diagrams: $gg \to ZH$

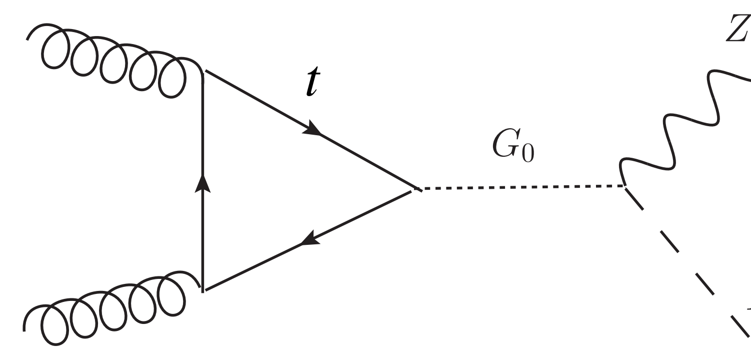
Leading Order (1-loop) Diagrams



Box

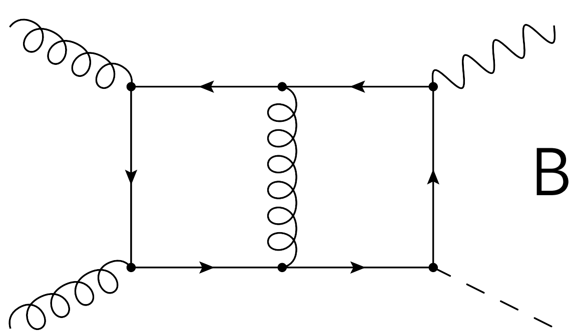


Triangle

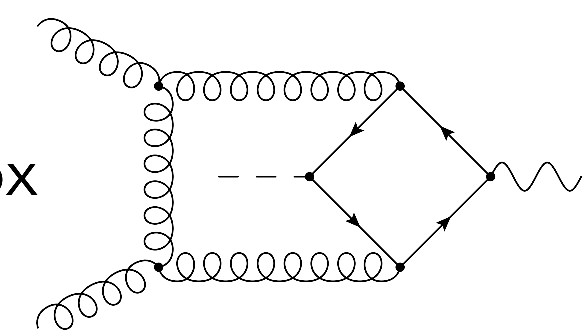


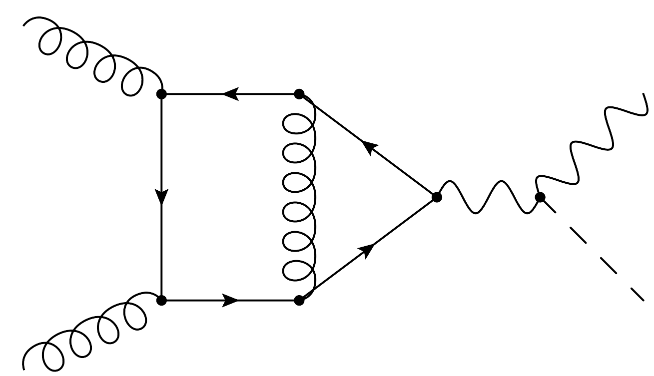
Goldstone

NLO (2-loop) Virtual Diagrams

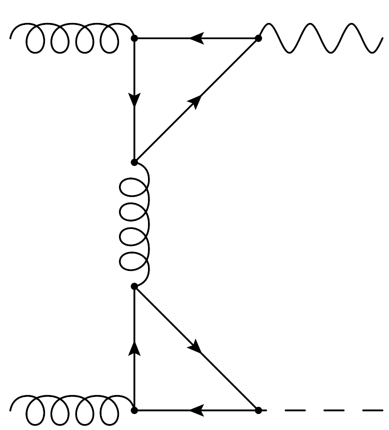


Box





Triangle &
Goldstone



1PR
(start at 2-loop)

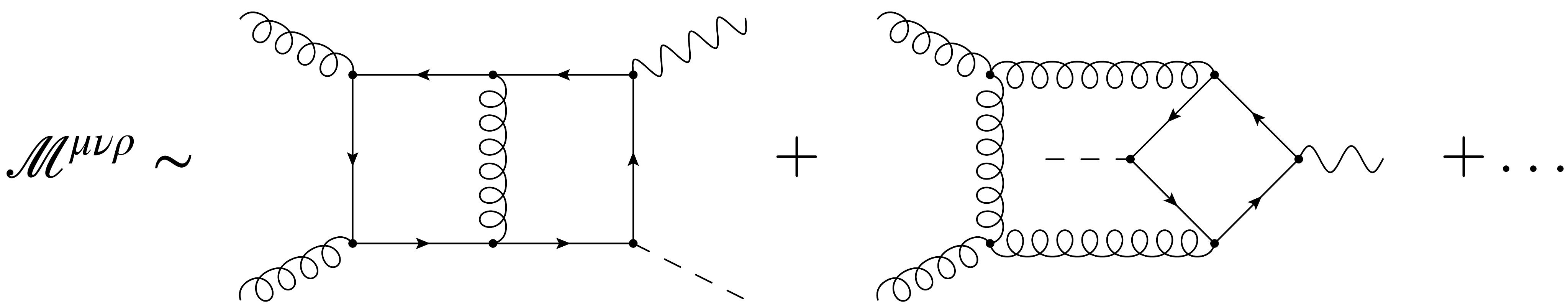
We compute in Feynman Gauge

Keep the dependence on all EW couplings symbolic (can be varied)

Set $m_b = 0$

Amplitudes

Schematically:



$$\mathscr{M}^{\mu\nu\rho} = \sum_i A_i T_i^{\mu\nu\rho}, \qquad A_i = \sum_k C_{i,k}\ I_k$$

Rational functions
Large num. terms/ high degree
Handled with specialist symbolic manipulation programs

Feynman integrals
Analytically: Involved special functions (Polylogs, Elliptic…)

In this work, we will compute them numerically

Dealing with the Integrals

Feynman Integrals

Feynman integrals have many faces, each make different properties manifest…
Switching the representation of our integrals allows us to understand/simplify/complete the calculation

Feynman Parametrisation

$$I \sim \int_0^\infty \prod_{j=1}^{N} \mathrm{d}x_j \;\; x_j^{\nu_1-1} \dots x_j^{\nu_N-1} \delta(1-\sum_{i=1}^{N} x_i) \frac{\mathcal{U}^{N_\nu-(L+1)d/2}}{\mathcal{F}^{N_\nu-Ld/2}}$$

$$I = \int \mathrm{d}^d k_1 \dots \mathrm{d}^d k_l \; \frac{1}{\prod_{i=1}^{N} (q_i^2 - m_i^2)_i^{\nu}}$$

…

⋮

…

Lee-Pomeransky Representation

$$I \sim \int_0^\infty \prod_{j=1}^{N} \mathrm{d}x_j \; x_j^{\nu_1-1} \dots x_j^{\nu_N-1} G^{-D/2}$$

$$G = \mathcal{U} + \mathcal{F}$$

Lee, Pomeransky 13

Baikov Representation

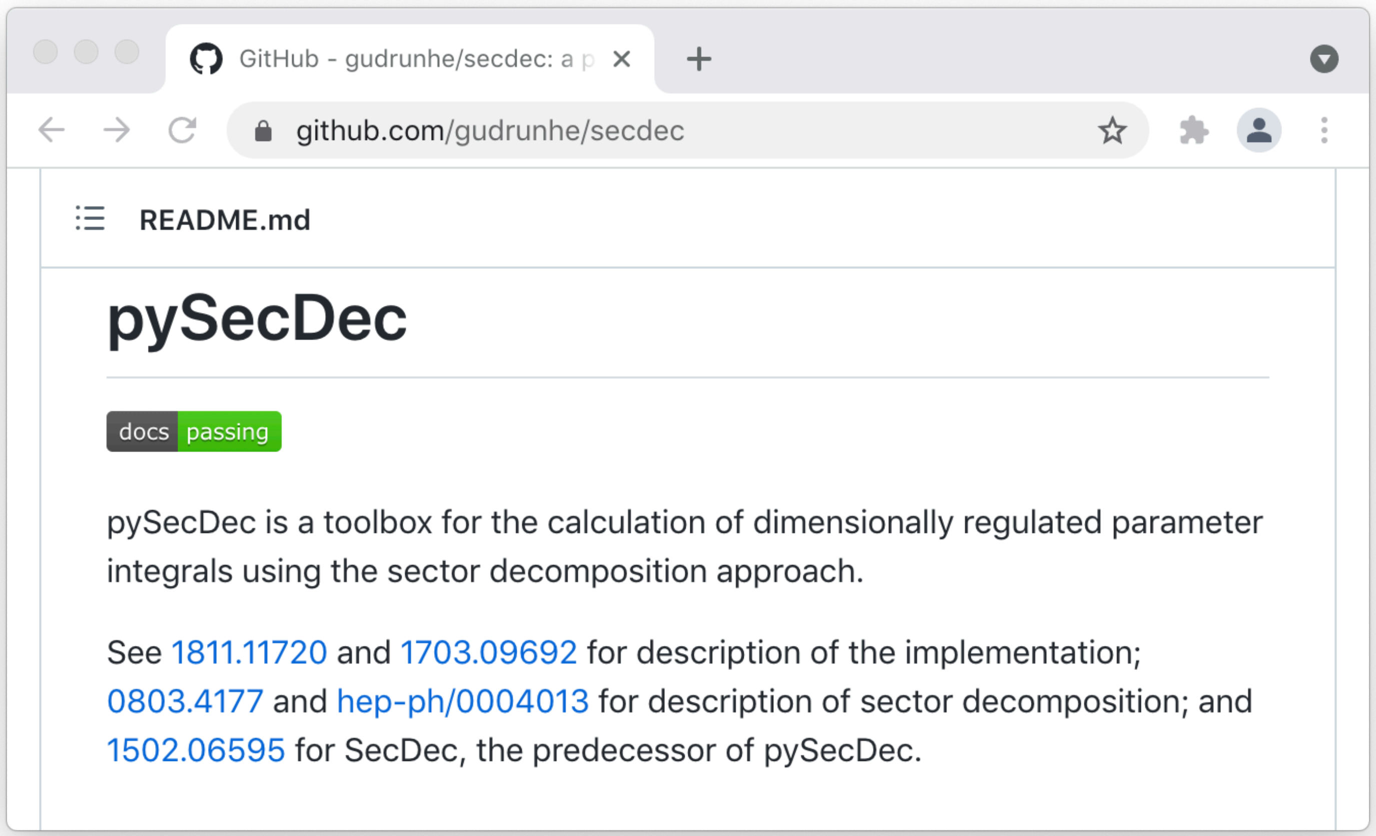
$$I \sim \int dz_1 \dots dz_N \; \frac{1}{\prod_{i=1}^{N} \; z_i^{\nu_i}} \; P^{\frac{d-L-E-1}{2}}$$

Baikov 96

pySecDec

pySecDec: a program for numerically evaluating dimensionally regulated parameter integrals on CPU or GPU (written in python, FORM, c++, CUDA)

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



Publicly available (Github)

Extensive tests (CI) and documentation

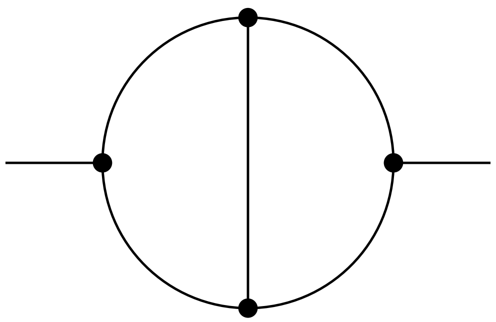
Install with:

```
python3 -m pip install --user --upgrade pySecDec
```

New: Expansion by Regions & Amplitude Evaluation

Heinrich, Jahn, SPJ, Kerner, Langer, Magerya, Põldaru, Schlenk, Villa 21

Computing Integrals Numerically

Input: $I = \int \mathrm{d}^d k_1 \cdots \mathrm{d}^d k_l \frac{1}{(k_1^2 - m^2)((k_1-k_2)^2 - m^2)\cdots}$ **or** $G =$ 

Feynman Parametrise

$$\mathcal{U} = x_1 x_3 + x_1 x_4 + x_1 x_5 + \ldots$$
$$\mathcal{F} = -p^2 x_1 x_2 x_3 - p^2 x_1 x_2 x_4 - \ldots$$

Resolve Singularities (Sector Decomposition)

C++ library ready for numerical integration

Output:

```
$ time python integrate_elliptic2L_euclidean.py
 + (2.470741991407315 60e-01 +/- 4.360755998051713 55e-16) + O(eps)
real        0m2.576s
```

Other Sector Decomposition Tools

Several other codes implement sector decomposition

Public:

- sector_decomposition + CSectors (uses GiNaC)
  https://particlephysics.uni-mainz.de/weinzierl/sector_decomposition/
  Bogner, Weinzierl 07; Gluza, Kajda, Riemann, Yundin 10

- FIESTA (uses Mathematica, C) ← **First to implement expansion by regions**
  https://bitbucket.org/feynmanIntegrals/
  Supports integration on GPUs
  A. Smirnov, V. Smirnov, Tentyukov 08, 09, 13, 15; Smirnov 16, 21

(Currently) Private:

- FORM & Python Implementations
  Fujimoto, Kaneko and Ueda 08, 10

- NIFT
  Zhao (in preparation)

New Feature 1: Amplitude Evaluation

New release: can evaluate entire amplitudes **(used in our ZH calculation)**
Let's see how this works in a simple example 1-loop 4-photon amp.

Step 1: Define Integrals

```
import pySecDec as psd

### Integral definitions ###

I = [

    # one loop bubble (u)
    psd.loop_integral.LoopIntegralFromGraph(
    internal_lines = [[0,[1,2]],[0,[2,1]]],
    external_lines = [['p1',1],['p2',2]],
    replacement_rules = [('p1*p1', 'u'),('p2*p2', 'u'),('p1*p2', 'u')]),

    # one loop bubble (t)
    psd.loop_integral.LoopIntegralFromGraph(
    internal_lines = [[0,[1,2]],[0,[2,1]]],
    external_lines = [['p1',1],['p2',2]],
    replacement_rules = [('p1*p1', 't'),('p2*p2', 't'),('p1*p2', 't')]),

    # one loop box (in 6 dimensions)
    psd.loop_integral.LoopIntegralFromGraph(
    internal_lines = [['0',[1,2]],[0,[2,3]],[0,[3,4]],[0,[4,1]]],
    external_lines = [['p1',1],['p2',2],['p3',3],['p4',4]],
    replacement_rules = [
                        ('p1*p1', 0), ('p2*p2', 0),
                        ('p3*p3', 0), ('p4*p4', 0),
                        ('p3*p2', 'u/2'), ('p1*p2', 't/2'),
                        ('p1*p4', 'u/2'), ('p1*p3', '-u/2-t/2'),
                        ('p2*p4', '-u/2-t/2'), ('p3*p4', 't/2')
                    ],
    dimensionality= '6-2*eps'
    ),

    # one loop box (in 8 dimensions)
    # ...

    )
]
```

Usual pySecDec Syntax

Step 2: Define Coefficients

```
from pySecDec import Coefficient

### Coefficients definitions ###

coeff = [
    # M++--
    [
        # bubble (u) coefficient
        Coefficient(['-8*(t-u)'],['-u-t'],['t','u']),

        # bubble (t) coefficient
        Coefficient(['-8*(u-t)'],['-u-t'],['t','u']),

        # box6 coefficient
        Coefficient(['-8*(t**2+u**2)'],['-u-t'],['t','u']),

        # box8 coefficient
        Coefficient(['-8*3*(2*eps)'],['1'],['t','u'])
    ]
]
```

Numerator **Denominator** **Variables**

Supports:
Multiple regulators & variables
Coeffs with poles in regulators
Multiple amplitudes at once

Amplitude Evaluation (II)

Step 3: Generate

```
from pySecDec.loop_integral import LoopPackage
from pySecDec.code_writer import sum_package

# import integrals and coefficients
import integrals
import coefficients

# 4-photon amplitude M++--

if __name__ == '__main__':

    # define input to loop_package
    package_generators = []
    names = ['bubble_u', 'bubble_t', 'box_6', 'box_8']
    for name, integral in zip(names, integrals.I):
        package_generators.append(
            LoopPackage(
                name = name,
                loop_integral = integral,
                requested_orders = [0]
                )
        )

    # generate code sum of (int * coeff)
    sum_package(
        'yyyy1L',
        package_generators,
        regulators = ['eps'],
        requested_orders = [0],
        real_parameters = ['t','u'],
        coefficients = coefficients.coeff,
        complex_parameters = []
    )
```

List of requested integrals

Pass integrals & coefficients

Step 4: Integrate

```
import math
from pySecDec.integral_interface import IntegralLibrary

if __name__ == "__main__":

    # load c++ library
    amp = IntegralLibrary('yyyy1L/yyyy1L_pylink.so')

    # choose Qmc integrator
    amp.use_Qmc()

    # integrate
    _, _, result = amp([-1.3,-0.8]) # t, u

    # print result
    print('Numerical Result:', result)
```

Uses all GPUs/ CPUs on machine

Usual pySecDec Syntax

Step 5: Generate, Compile, Run, ..., Profit!

```
$ python3 generate_yyyy1L.py
$ export CXX=nvcc
$ export SECDEC_WITH_CUDA_FLAGS="-gencode arch=compute_70,code=sm_70"
$ make -C yyyy1L -j 8
$ time python3 integrate_yyyy1L.py
> Numerical Result:
+ ((4.44089209850062616e-16,0.00000000000000000e+00)
+/- (2.30780196667706177e-16,0.00000000000000000e+00))*eps^-1

+ ((-2.84315958344628044e+01,-1.77910044838014154e-10)
+/- (4.25673251664533604e-09,2.41058630701162800e-09))

+ O(eps)

real    0m0.764s
user    0m2.884s
sys     0m0.188s
```

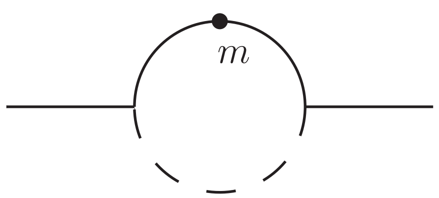
Timings on laptop CPU

New Feature 2: Expansion by Regions

One option for dealing with numerically unstable high-energy/threshold regions is expansion by regions **(not used in our ZH calculation)**

Beneke, Smirnov 98; Rakhmetov, Pak, Jantzen, Semenova, Becher, Neubert, Broggio, Ferroglia,…
(See e.g. Jantzen 11 for an introduction)

Idea: expand integrals around some small parameter, e.g. m^2/p^2



$$= \mu^{2\epsilon} \int dk \, \frac{1}{(k+p)^2(k^2-m^2)^2}$$

$$\text{(hard)}: \qquad |k^2| \gg m^2, \qquad \frac{1}{(k+p)^2(k^2-m^2)^2} \to \frac{1}{(k+p)^2\,(k^2)^2}\left(1 + 2\frac{m^2}{k^2} + \ldots\right)$$

$$\text{(soft)}: \qquad |k^2|, |k\cdot p| \ll p^2, \qquad \frac{1}{(k+p)^2(k^2-m^2)^2} \to \frac{1}{p^2\,(k^2-m^2)^2}\left(1 - \frac{k^2+2p\cdot k}{p^2} + \ldots\right)$$

Integrate expanded integrals over full integration range, sum over all regions

Concept can be systematically applied also in Feynman parameter space

Implemented in tools such as FIESTA/ ASY/ ASY2 and now in pySecDec

Smirnov 15; Smirnov, Smirnov, Tentyukov 09;

Feynman Integrals: Feynman Parametrisation

Introducing Feynman parameters and integrating over momenta, we obtain

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

L - # loops in Feynman integral, $N_\nu = \sum_{i=1}^{N} \nu_i$ - sum of propagator powers

$\mathcal{U}, \mathcal{F}$ are **homogenous polynomials** in the Feynman parameters x_i

$\mathcal{U}(\mathbf{x})$ is degree L

$\mathcal{F}(\mathbf{x})$ is degree $L+1$

$\mathcal{F}(\mathbf{x}, s_{ij}) = \mathcal{F}_0(\mathbf{x}, s_{ij}) + \mathcal{U}(\mathbf{x}) \sum_{i=1}^{N} x_i m_i^2$ ← **internal masses**

Both $\mathcal{U}, \mathcal{F}_0$ are linear in each Feynman parameter

It is straightforward to construct $\mathcal{U}, \mathcal{F}$ from a loop integral or graphically

Feynman Integrals: Lee-Pomeransky Representation

The Lee-Pomeransky representation is given by Lee, Pomeransky 13

$$G = \frac{\Gamma(D/2)}{\Gamma\left((L+1)\,D/2 - N_\nu\right)\prod_{j=1}^{N}\Gamma(\nu_j)} \int_0^\infty \prod_{j=1}^{N} \mathrm{d}x_j\; x_j^{\nu_j - 1} \left(\mathscr{G}(\mathbf{x}, s_{ij})\right)^{-D/2}$$

$$\mathscr{G}(\mathbf{x}, s_{ij}) = \mathscr{U}(\mathbf{x}) + \mathscr{F}(\mathbf{x}, s_{ij})$$

Inserting $1 = \int \mathrm{d}s\, \delta(s - \sum x)$ and scaling $x \to sx$ we recover Feynman's formula

In this representation, we need to consider integrals of the form

$$G = \int_0^\infty \frac{\mathrm{d}\mathbf{x}}{\mathbf{x}} \mathbf{x}^{\nu} \left[\sum_{i=1}^{m} c_i\, \mathbf{x}^{\mathbf{p}_i}\right]^{-\frac{D}{2}}, \quad \text{where} \quad \mathbf{x}^{\mathbf{a}} = \prod_{j=1}^{N} x_j^{a_j}$$

This is useful for two reasons:

1) $\mathscr{U} + \mathscr{F}$ is typically simpler than $\mathscr{U} \times \mathscr{F}$
2) Many of the arguments can apply to a more general $\mathscr{G}$

Semenova, A. Smirnov, V. Smirnov 18

Geometric Method: Set-up

In Feynman parameter space, there is a **geometric method** for finding regions

Pak, Smirnov 10

Each region will be defined by a **region vector** $\mathbf{v} = (v_1, \ldots, v_N, 1)$, in each region we will perform a change of variables $x_i \to t^{v_i} x_i$ and series expand about $t = 0$

Let us start by considering some polynomial (could be $\mathscr{U} + \mathscr{F}$ or something more general):

$$P(\mathbf{x}, t) = \sum_{i=1}^{m} c_i \, x_1^{p_{i,1}} \, \cdots \, x_N^{p_{i,N}} \, t^{p_{i,N+1}}$$

c_i - non-negative coefficients

x_i - integration variables

t - small parameter

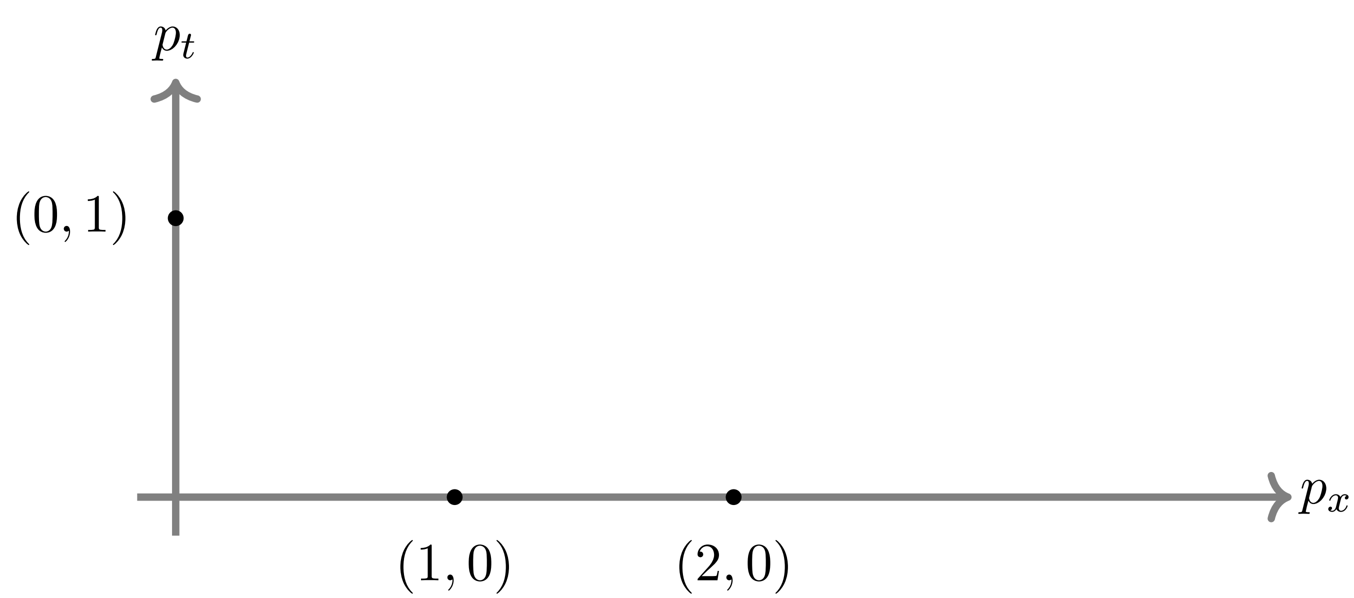
$\mathbf{p}_i' = (p_{i,1}, \ldots, p_{i,N+1}) \in \mathbb{N}^{N+1}$ - exponent vectors

Geometric Method: Determining the Regions

Ignoring, for now, the coefficients c_i we can introduce a simple but useful picture for such polynomials:

- For each variable x_i or t draw an orthogonal axis
- For each monomial, draw a dot at position $\mathbf{p}_i'$

Example: $P(x, t) = t + x + x^2$ has exponent vectors
$\mathbf{p}_1' = (0,1), \mathbf{p}_2' = (1,0), \mathbf{p}_3' = (2,0)$

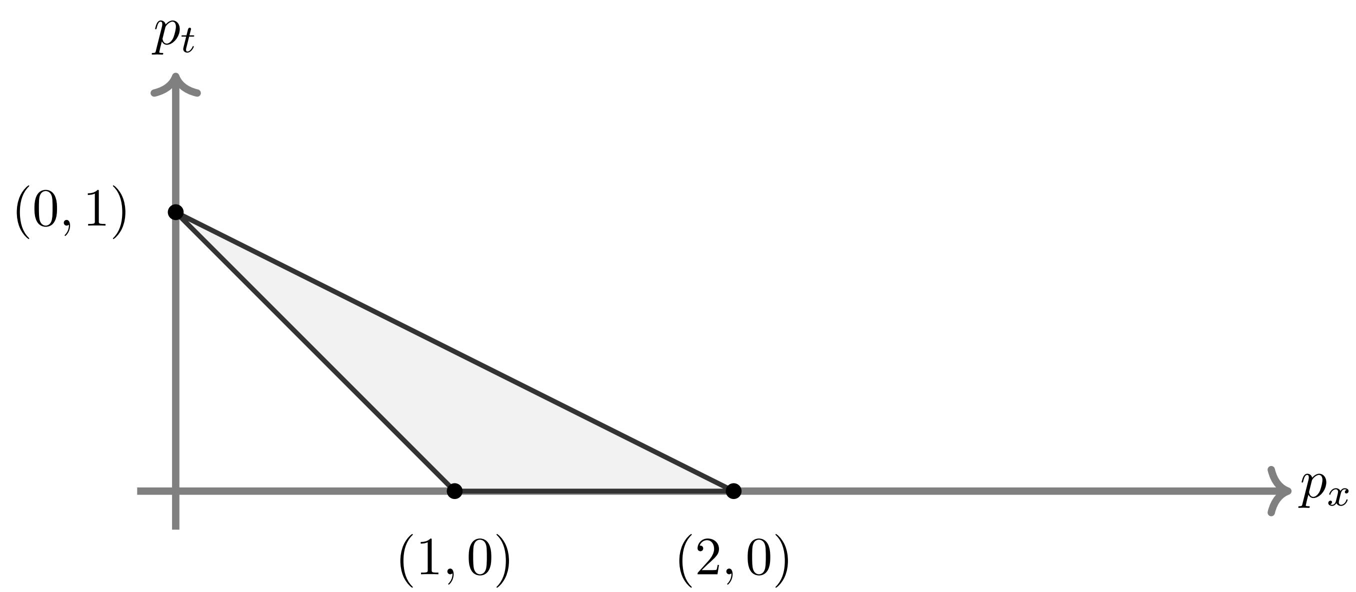


Geometric Method: Determining the Regions (II)

We may also define a **Newton polytope** of the polynomial, this is the convex hull of the exponent vectors:

$$\Delta = \mathrm{convHull}(\mathbf{p}'_1, \mathbf{p}'_2, \ldots) = \left\{ \sum_j \alpha_j \mathbf{p}'_j \,|\, \alpha_j \geq 0 \wedge \sum_j \alpha_j = 1 \right\}$$

Example: $P(x,t) = t + x + x^2$ has exponent vectors $\mathbf{p}'_1 = (0,1), \mathbf{p}'_2 = (1,0), \mathbf{p}'_3 = (2,0)$



Geometric Method: Determining the Regions (III)

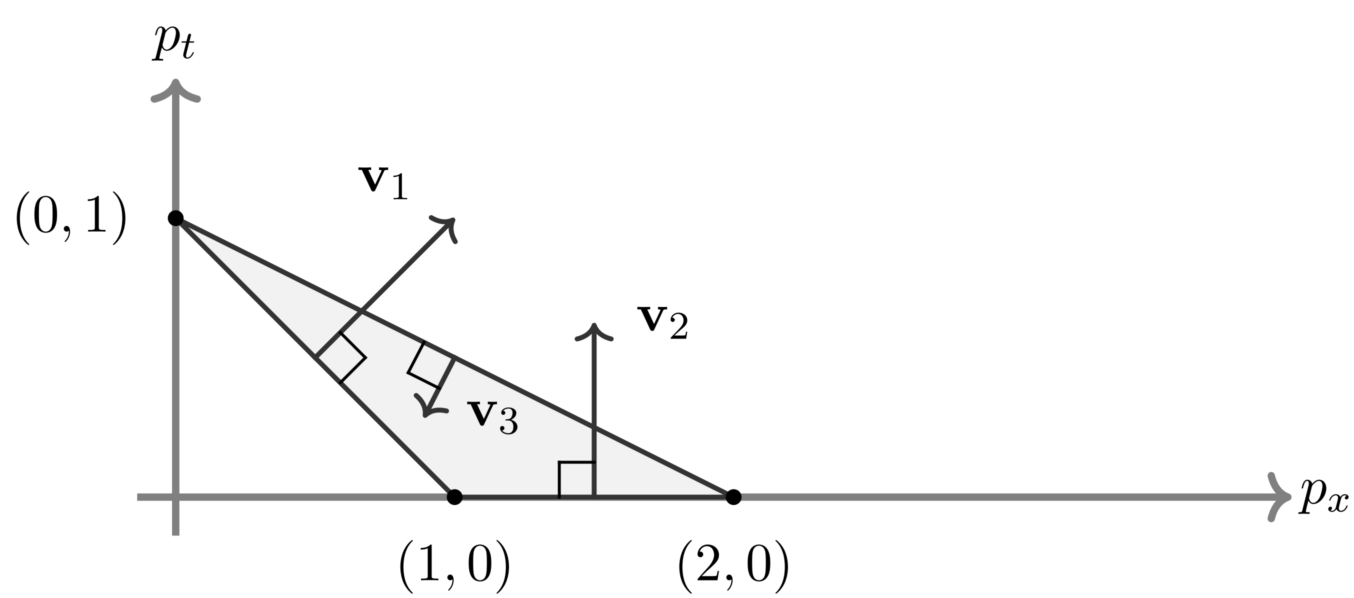
Alternatively, this polytope can also be described as the intersection of half spaces:

$$\Delta' = \bigcap_{f \in F} \left\{ \mathbf{m} \in \mathbb{R}^{N+1} \mid \langle \mathbf{m}, \mathbf{v}_f \rangle + a_f \geq 0 \right\}$$

F - set of polytope facets, $a_f \in \mathbb{Z}$

$\mathbf{v}_f$ - inward-pointing normal vectors for each facet

Several public tools exist for computing Newton polytopes/convex hulls and their representation in terms of facets exist, e.g. **Normaliz** and **Qhull**



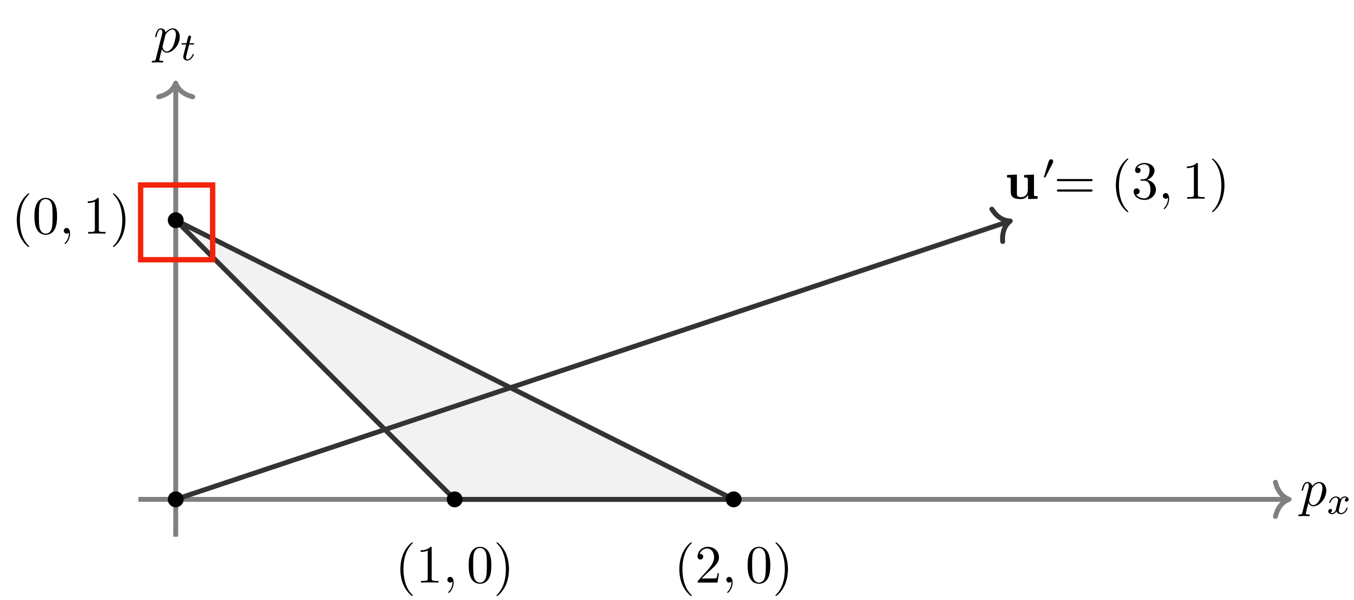
Geometric Method: Determining the Regions (IV)

Next, let us define a vector $\mathbf{u}$ such that $x_i = t^{u_i}$ and a vector $\mathbf{u}' = (\mathbf{u}, 1)$, for each point $\mathbf{x}$ in the integration domain, we can write:

$$P(t^{\mathbf{u}}, t) = \sum_{i=1}^{m} c_i \, t^{\langle \mathbf{p}_i', \mathbf{u}' \rangle}$$

Since $t \ll 1$, the largest term in the polynomial has the smallest $\langle \mathbf{p}_i', \mathbf{u}' \rangle$
Note that we can have several points with the same projection on $\mathbf{u}'$, i.e. we can have several largest terms

Example: $P(x,t) = t + x + x^2$ with $\mathbf{u}' = (3,1)$ gives $P(t^{\mathbf{u}}, t) = \underline{t} + t^3 + t^6$



Geometric Method: Determining the Regions (V)

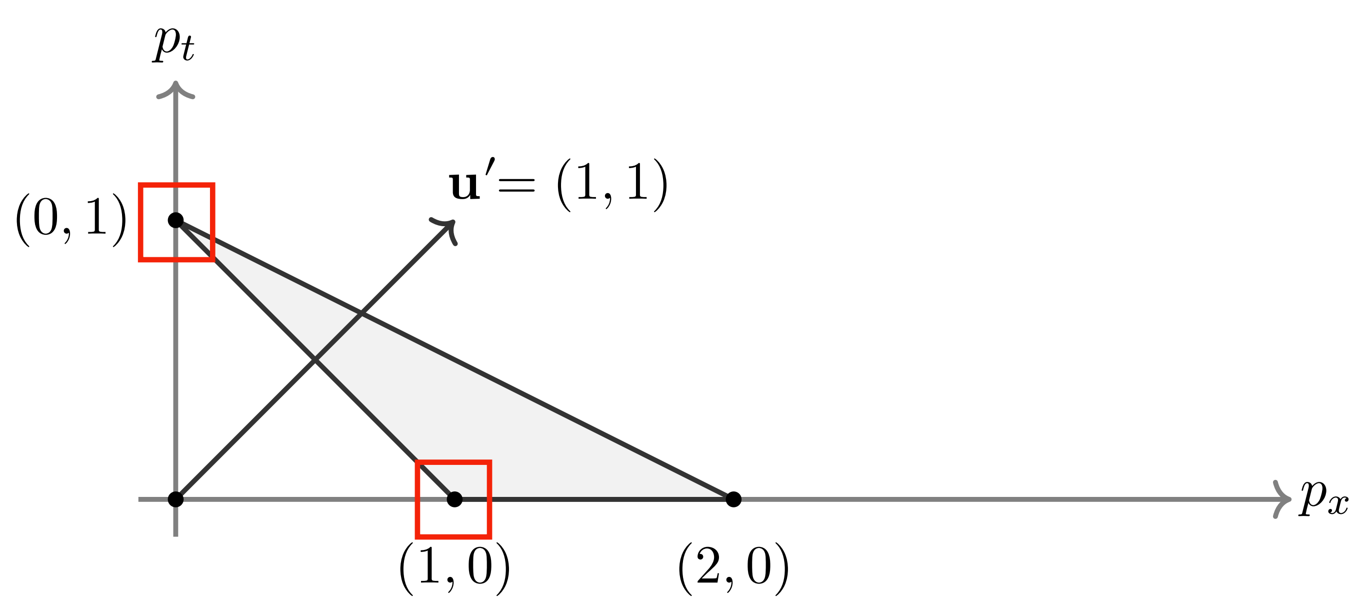
Next, let us define a vector $\mathbf{u}$ such that $x_i = t^{u_i}$ and a vector $\mathbf{u}' = (\mathbf{u}, 1)$, for each point $\mathbf{x}$ in the integration domain, we can write:

$$P(t^{\mathbf{u}}, t) = \sum_{i=1}^{m} c_i \, t^{\langle \mathbf{p}_i', \mathbf{u}' \rangle}$$

Since $t \ll 1$, the largest term in the polynomial has the smallest $\langle \mathbf{p}_i', \mathbf{u}' \rangle$

Note that we can have several points with the same projection on $\mathbf{u}'$, i.e. we can have several largest terms

Example: $P(x,t) = t + x + x^2$ with $\mathbf{u}' = (1,1)$ gives $P(t^{\mathbf{u}}, t) = \underline{t + t} + t^2$



Geometric Method: Determining the Regions (VI)

Rewrite our polynomial as: $P(\mathbf{x}) = Q(\mathbf{x}) + R(\mathbf{x})$

With $Q(\mathbf{x})$ defined such that it contains all of the lowest order terms in t

Then, binomial expansion of

$P(\mathbf{x})^m = Q(\mathbf{x})^m \left(1 + \dfrac{R(\mathbf{x})}{Q(\mathbf{x})}\right)^m$ converges for $\mathbf{x} = t^{\mathbf{u}}$ if $R(\mathbf{x})/Q(\mathbf{x}) < 1$

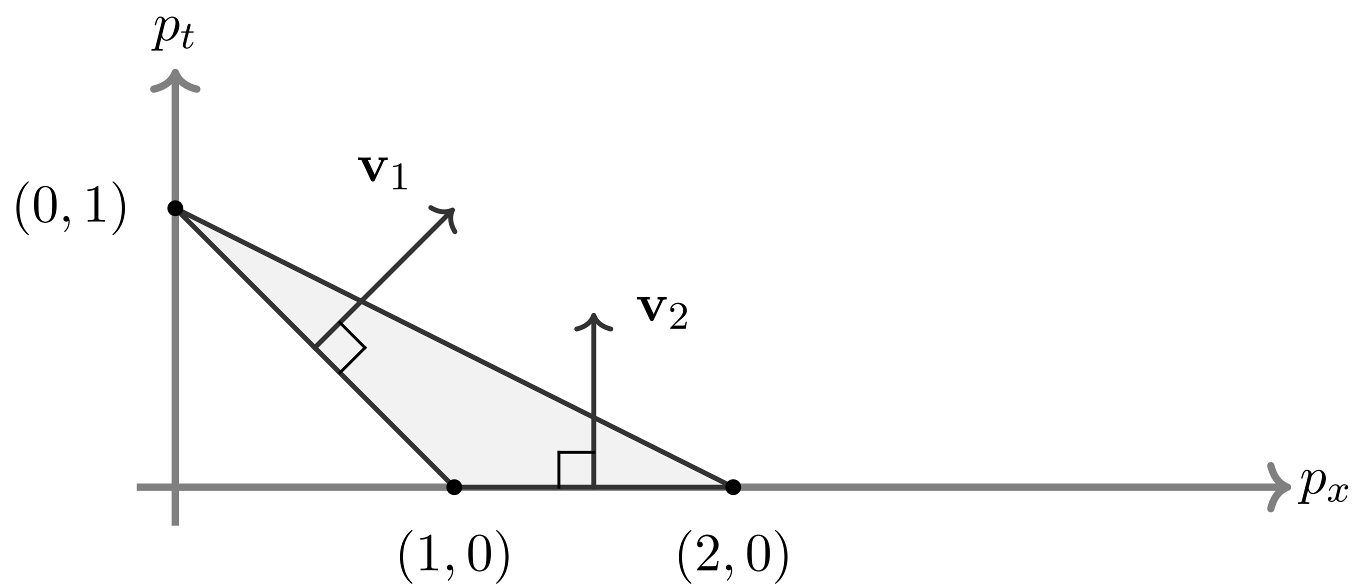
Some observations:

- An expansion with region vector $\mathbf{v}$ converges at a point $\mathbf{u}'$ if the lowest order terms along the direction $\mathbf{v}$ contain the lowest order terms along the direction $\mathbf{u}'$
- For any direction $\mathbf{u}'$ the vertices with the smallest $< \mathbf{p}_i', \mathbf{u}' >$ must be part of some facet F of the polytope
- Since $u_{N+1} > 0$, the lowest order terms for any $\mathbf{u}'$ must lie on a facet whose inwards pointing normal vector has a positive $(N+1)$-th component, let us call the set of such facets F^+

Geometric Method: Determining the Regions (VII)

How do we choose the regions?

The region vectors may be chosen as the facets whose inwards pointing normal vector has a positive $(N+1)$ component



Our original integral G may then be approximated as $G = \sum_{f \in F^+} G^{(f)} + \ldots$

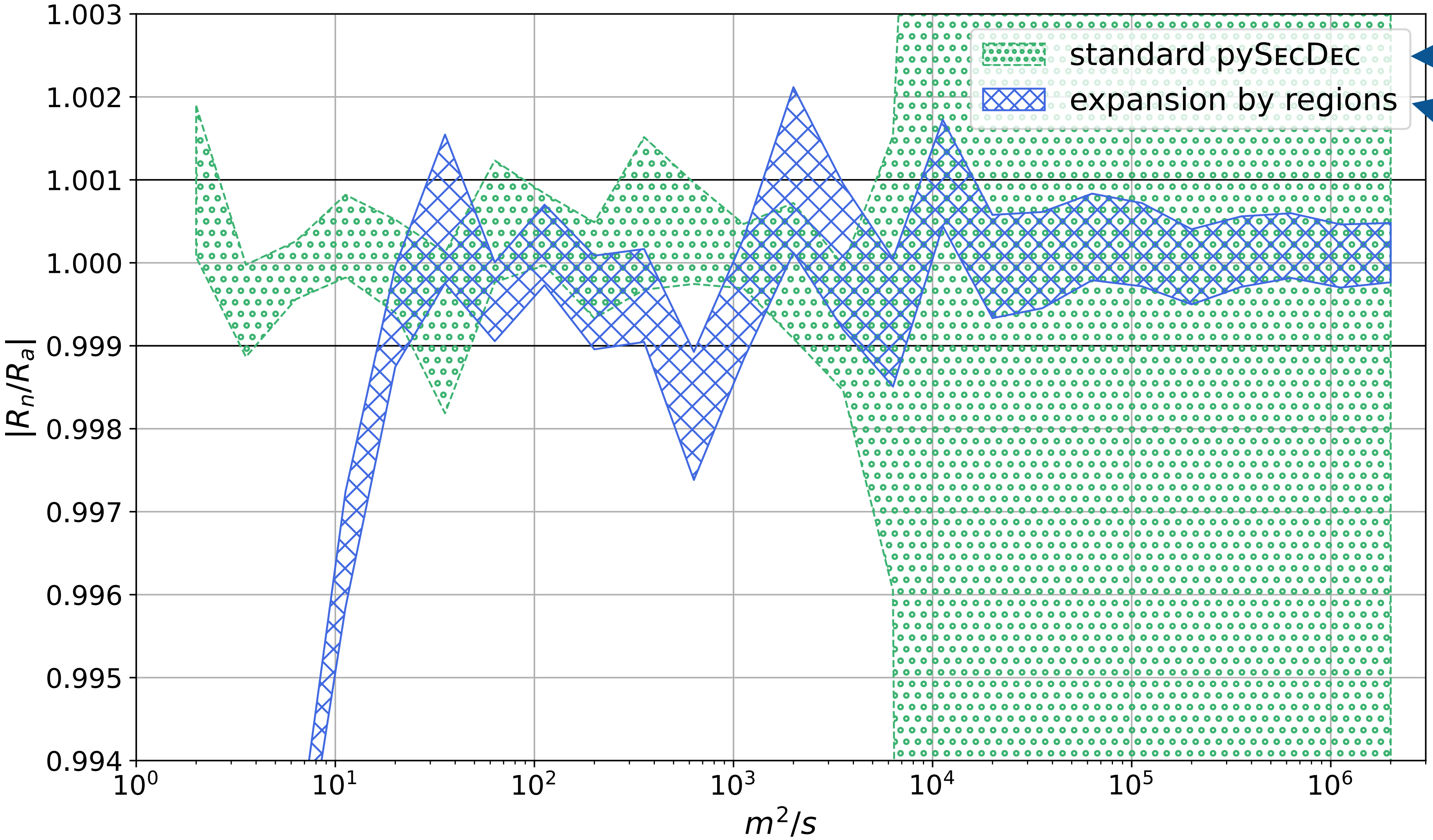
Where $G^{(f)}$ are the series expanded integrals integrated over the whole domain

The ``$+\ldots$" terms are overlap contributions/ multiple expansions, for sufficiently regulated Feynman integrals (dim reg + analytic regulators) these terms are usually scaleless (=0 in dim reg) and can be neglected

Expansion by Regions

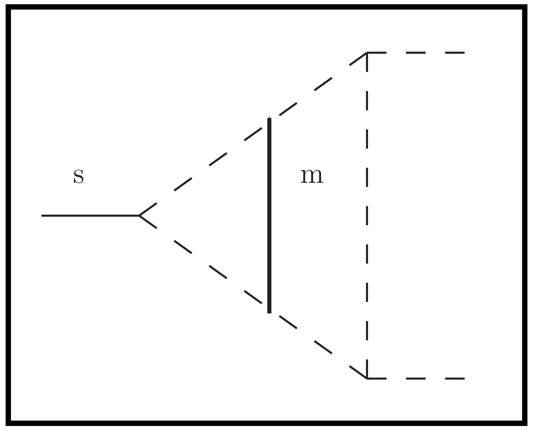
Consider a 2-loop form factor integral, plot the ratio of the finite $\mathcal{O}(\epsilon^0)$ piece of our numerical result R_n to the analytic result R_a

(at large m^2/s)



~1 day

~2 min



Where we have a large ratio of scales (m^2/s) the EBR result is much **faster** & **easier** to integrate

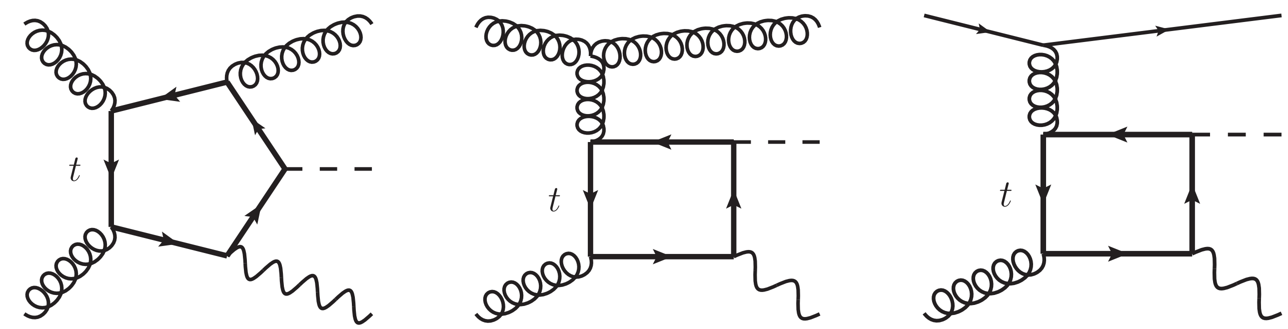
Results $gg \to ZH$

Finite Virtual Correction

Schematically,

$$\hat{\sigma} = \hat{\sigma}^{\mathrm{LO}} + \hat{\sigma}^{\mathrm{NLO}}$$
$$\hat{\sigma}^{\mathrm{LO}} = \int_n \mathrm{d}\sigma^{\mathrm{B}}$$
$$\hat{\sigma}^{\mathrm{NLO}} = \int_n \mathrm{d}\sigma^{\mathrm{V}} + \int_{n+1} \mathrm{d}\sigma^{\mathrm{R}} + \int_n \mathrm{d}\sigma^{\mathrm{C}}$$

Reals (not included in our work yet)



Virtual part ($\mathrm{d}\sigma^V$) and real part ($\mathrm{d}\sigma^R$) not separately finite for $\epsilon \to 0$

However, we can define a finite virtual contribution as follows:
1) UV renormalize: α_s in $\overline{\mathrm{MS}}$ & top quark mass in OS scheme
2) IR structure well known at NLO, subtract divergences

$$\mathcal{A}_i^{(0),\mathrm{fin}} = \mathcal{A}_i^{(0),\mathrm{UV}},$$
$$\mathcal{A}_i^{(1),\mathrm{fin}} = \mathcal{A}_i^{(1),\mathrm{UV}} - I_1 \mathcal{A}_i^{(0),\mathrm{UV}},$$

$$I_1 = I_1^{\mathrm{soft}} + I_1^{\mathrm{coll}},$$
$$I_1^{\mathrm{soft}} = -\frac{e^{\epsilon\gamma_E}}{\Gamma(1-\epsilon)} \left(\frac{\mu_R^2}{s}\right)^{\epsilon} \left(\frac{1}{\epsilon^2} + \frac{i\pi}{\epsilon}\right) 2C_A,$$
$$I_1^{\mathrm{coll}} = -\frac{\beta_0}{\epsilon} \left(\frac{\mu_R^2}{s}\right)^{\epsilon}.$$

A Few Conventions

We present results for the Born and Born-Virtual interference helicity amplitudes

Expand the helicity amplitudes in α_S

$$\mathscr{A}^{\text{fin}}_{\lambda_1\lambda_2\lambda_3} = \left(\frac{\alpha_s}{4\pi}\right) \mathscr{A}^{(0),\text{fin}}_{\lambda_1\lambda_2\lambda_3} + \left(\frac{\alpha_s}{4\pi}\right)^2 \mathscr{A}^{(1),\text{fin}}_{\lambda_1\lambda_2\lambda_3} + \ldots$$

Compute the square/interference

$$\mathscr{B} = \frac{1}{4} \sum_{\lambda_1\lambda_2\lambda_3} |\mathscr{A}^{(0),\text{fin}}_{\lambda_1\lambda_2\lambda_3}|^2,$$

$$\mathscr{V} = \frac{1}{4} \sum_{\lambda_1\lambda_2\lambda_3} 2\,\text{Re}\left(\mathscr{A}^{*(0),\text{fin}}_{\lambda_1\lambda_2\lambda_3} \mathscr{A}^{(1),\text{fin}}_{\lambda_1\lambda_2\lambda_3}\right)$$

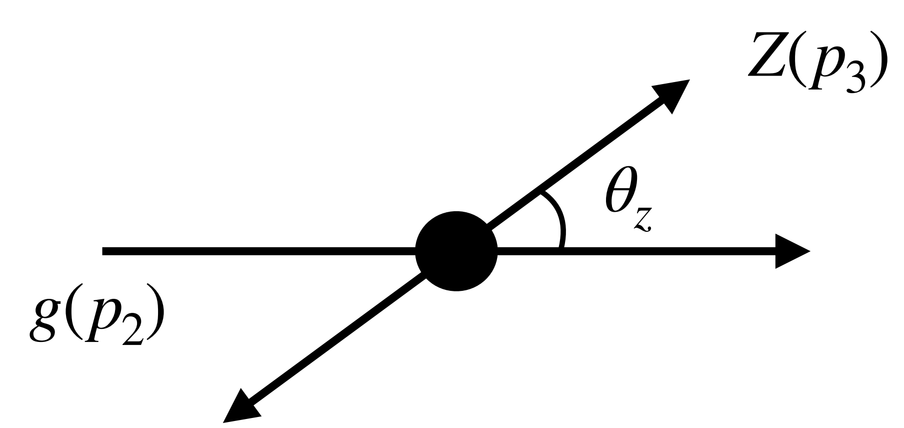
Renormalization scale set to $\mu_R^2 = s$
Electroweak coupling $e^2 = 4\pi\alpha = 1$
(can easily vary couplings/scales)

$2 \to 2$ amplitude depends on two kinematic variables (after fixing masses)

Choose:

$s = (p_1 + p_2)^2$

θ_z - angle in c.o.m frame between p_2-axis and p_3



Evaluation of the Amplitude

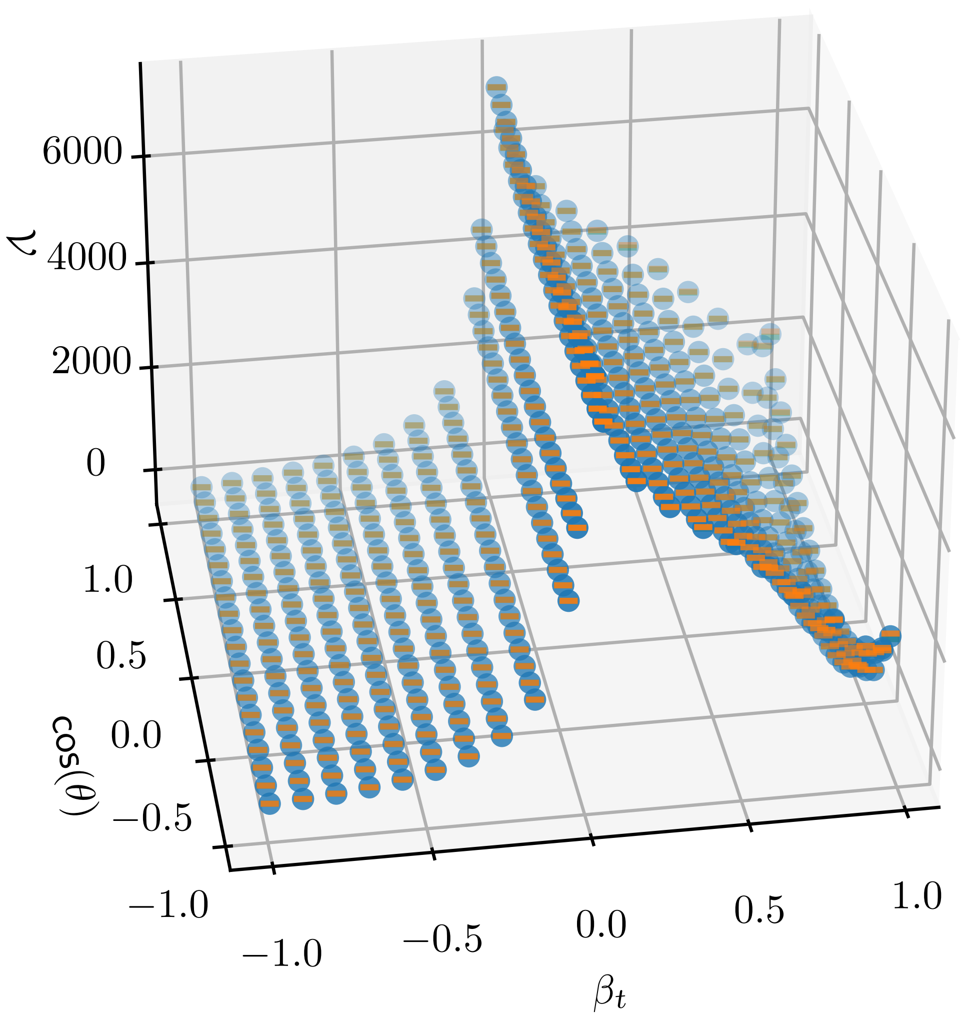
Again, $2 \to 2$ amplitude depends on two kinematic variables (after fixing masses)

$$\beta_t = \frac{s - 4m_t^2}{s + 4m_t^2 - (2m_z + m_h)^2},$$

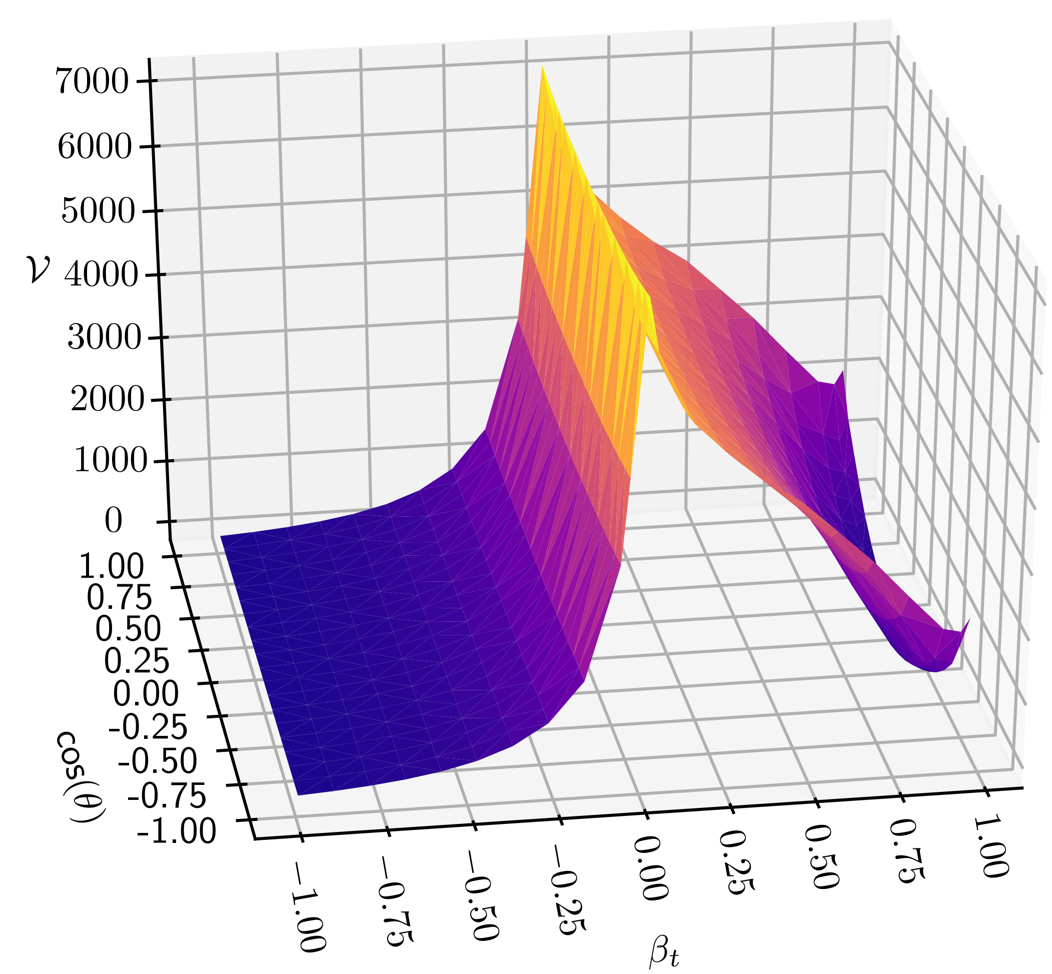
θ_z - angle in c.o.m frame between p_2-axis and Z-boson (p_3)

Sample grid of 20×20 points $+80$ extra top threshold/high-energy points in range:

$$-0.99 < \beta_t < 0.99 \text{ and } -0.99 < \cos(\theta_z) < 0.99$$

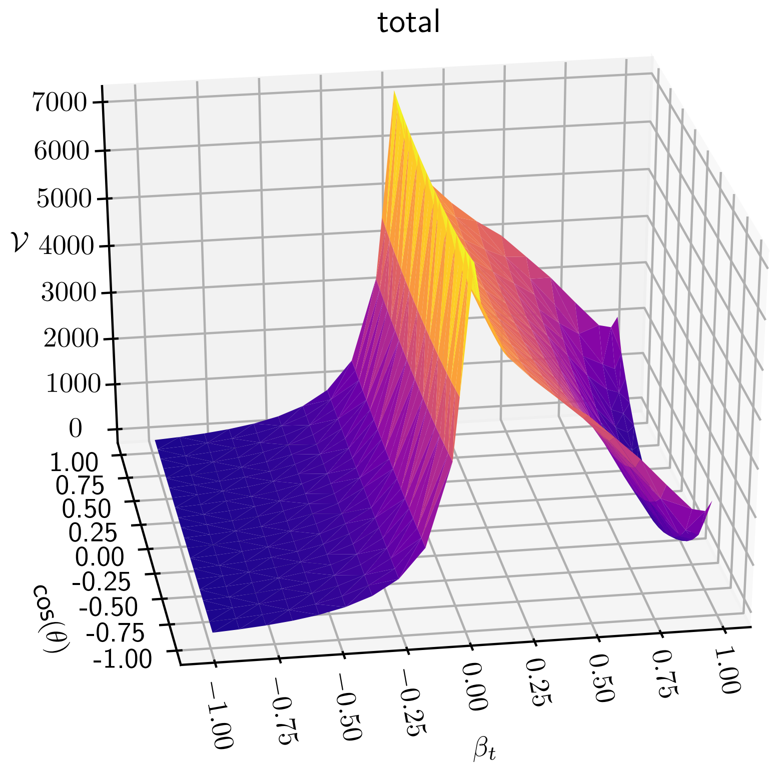


Interpolated surface (visualisation)

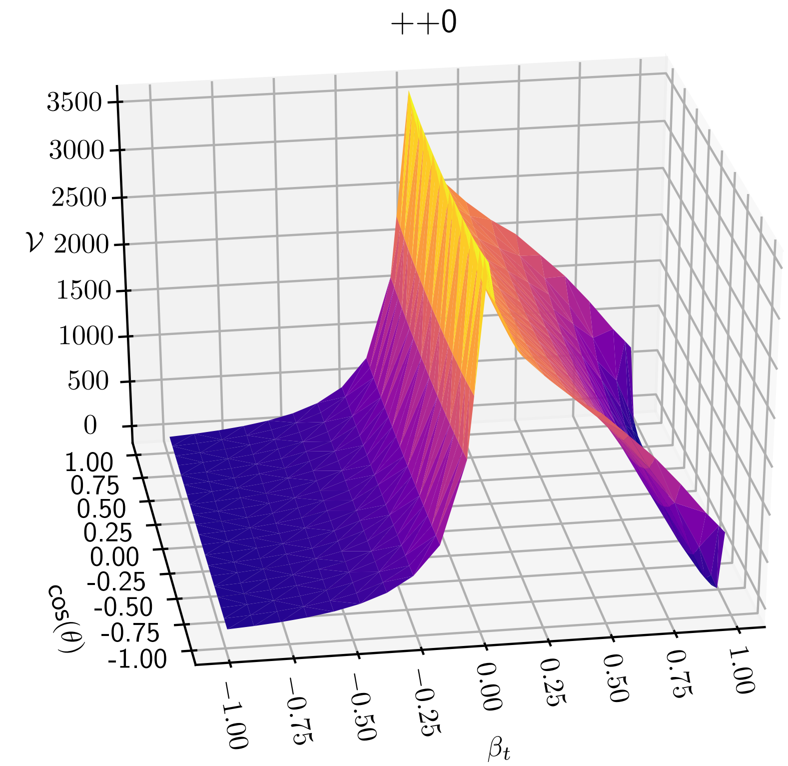


Amplitude Result

Observe that modes with longitudinally polarised Z boson dominate total

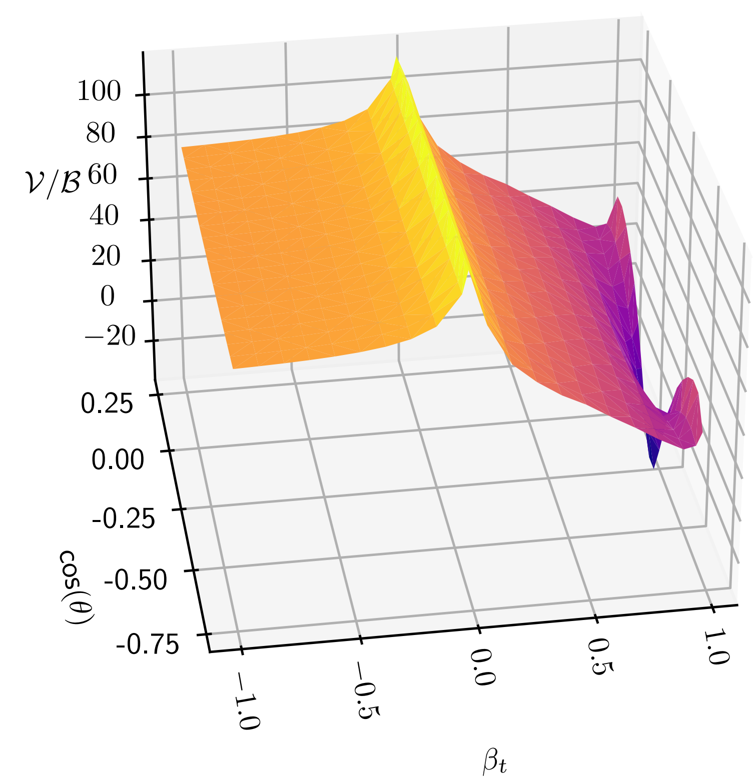


$\sim 2\times$



Ratio between Born squared amplitude and Born-Virtual interference not flat

Reminder: plot missing real contribution (so plot is not NLO/LO `K-factor')

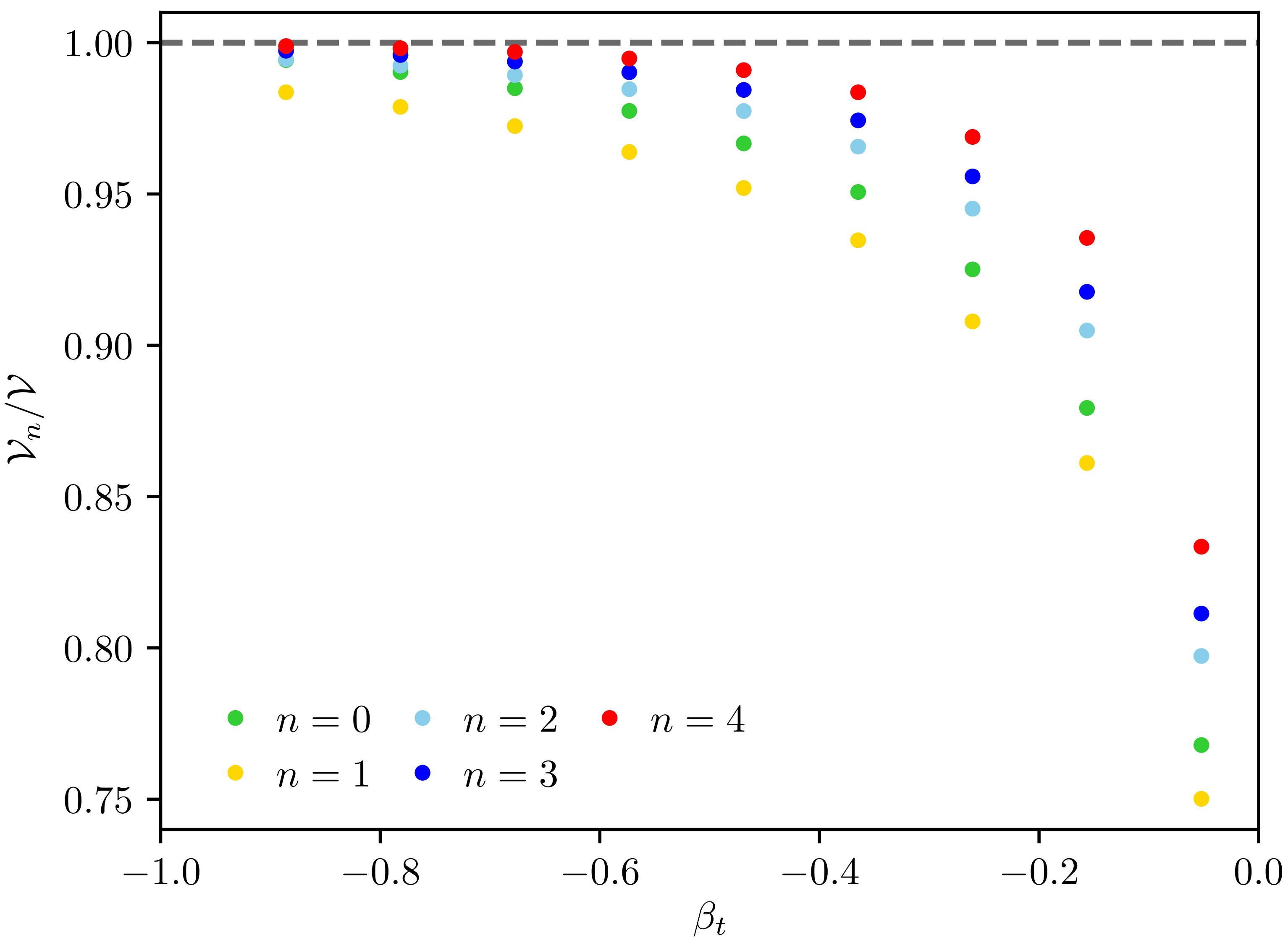


Comparison to Large m_t Expansion

The amplitude has been expanded around large-m_t and computed analytically
Hasselhuhn, Luthe, Steinhauser 17; Davies, Mishima, Steinhauser 20

Let us compare our result to the Born reweighted $1/m_t^{2n}$ expansion:

$$\mathcal{V}_n = \frac{\mathcal{B}}{\mathcal{B}_n}\widetilde{\mathcal{V}}_n + \mathcal{V}^{\mathrm{1PR}}$$



Per mille level agreement far below top quark threshold:

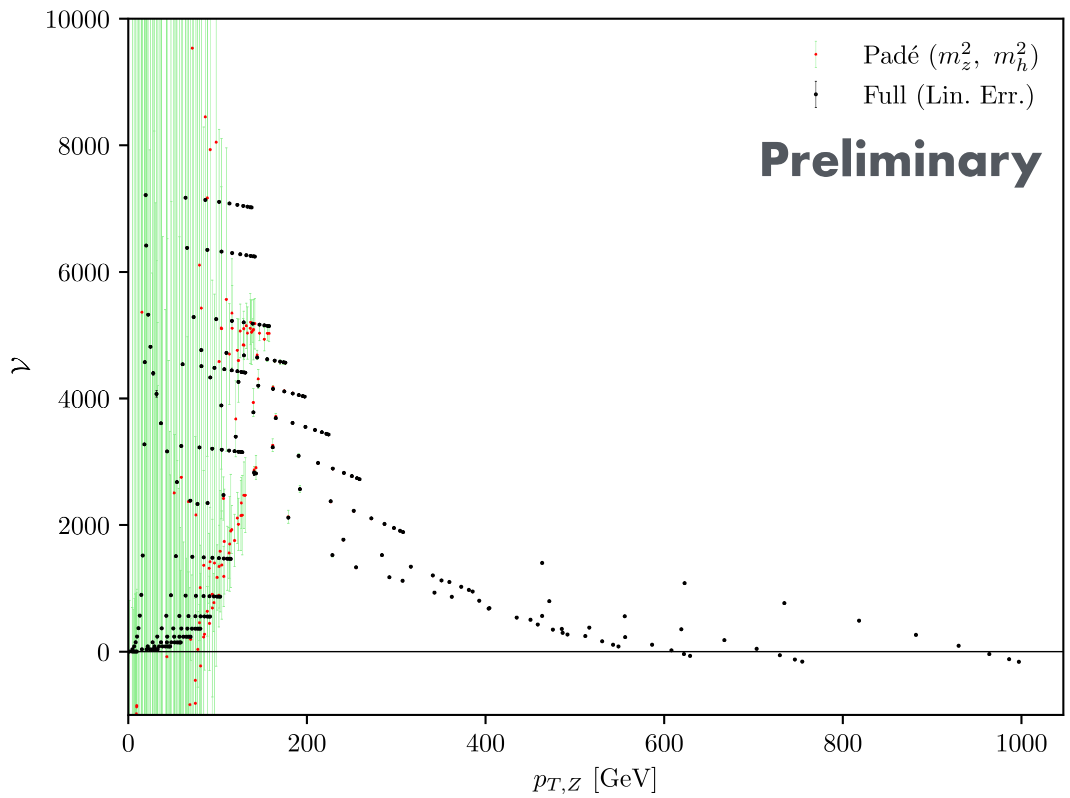
$$\mathscr{V}_4/\mathscr{V} = 0.9989$$

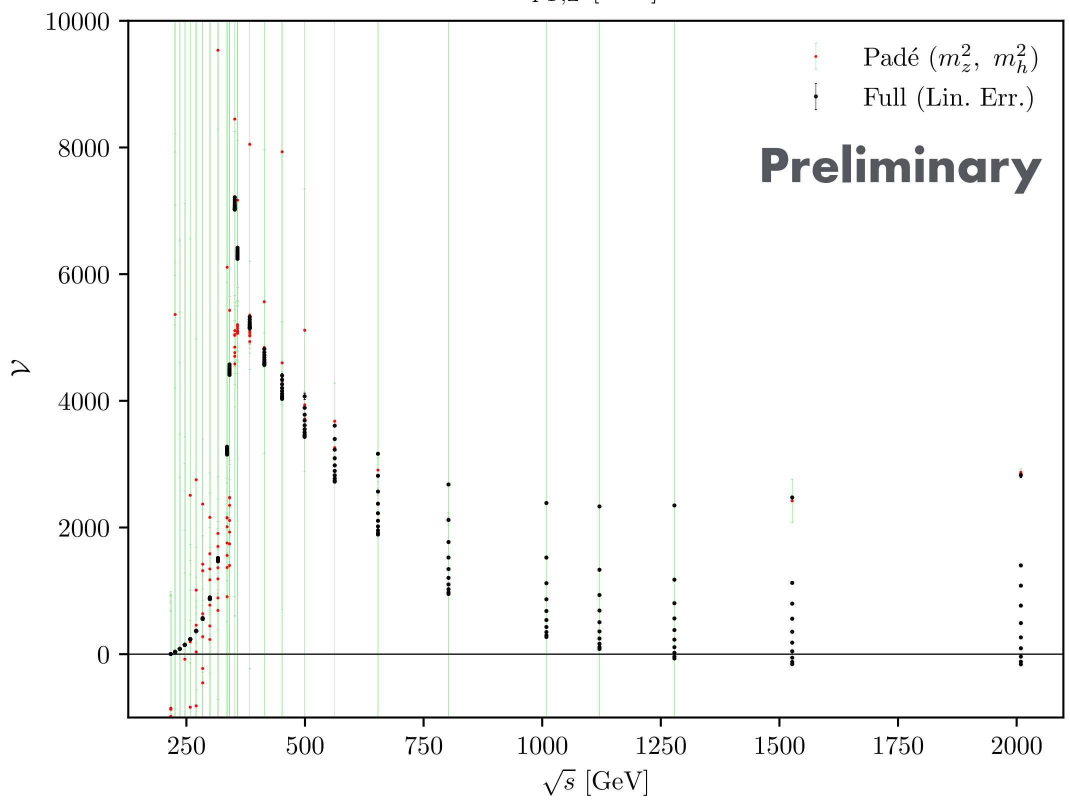
Expansion breaks down at threshold, observe that it differs from our result

Observation: $n = 1$ apparently worse than $n = 0$

Comparison to Small m_t Expansion

The amplitude has also been expanded around small m_t, m_h, m_z





Davies, Mishima, Steinhauser 20; Mishima 18

Agreement with Padé improved expanded result $\mathcal{O}(m_z^2,\ m_h^2,\ m_t^{32}\ )$
~2% level for $p_T \gtrsim 225$
~10% level for $150 < p_T < 225$

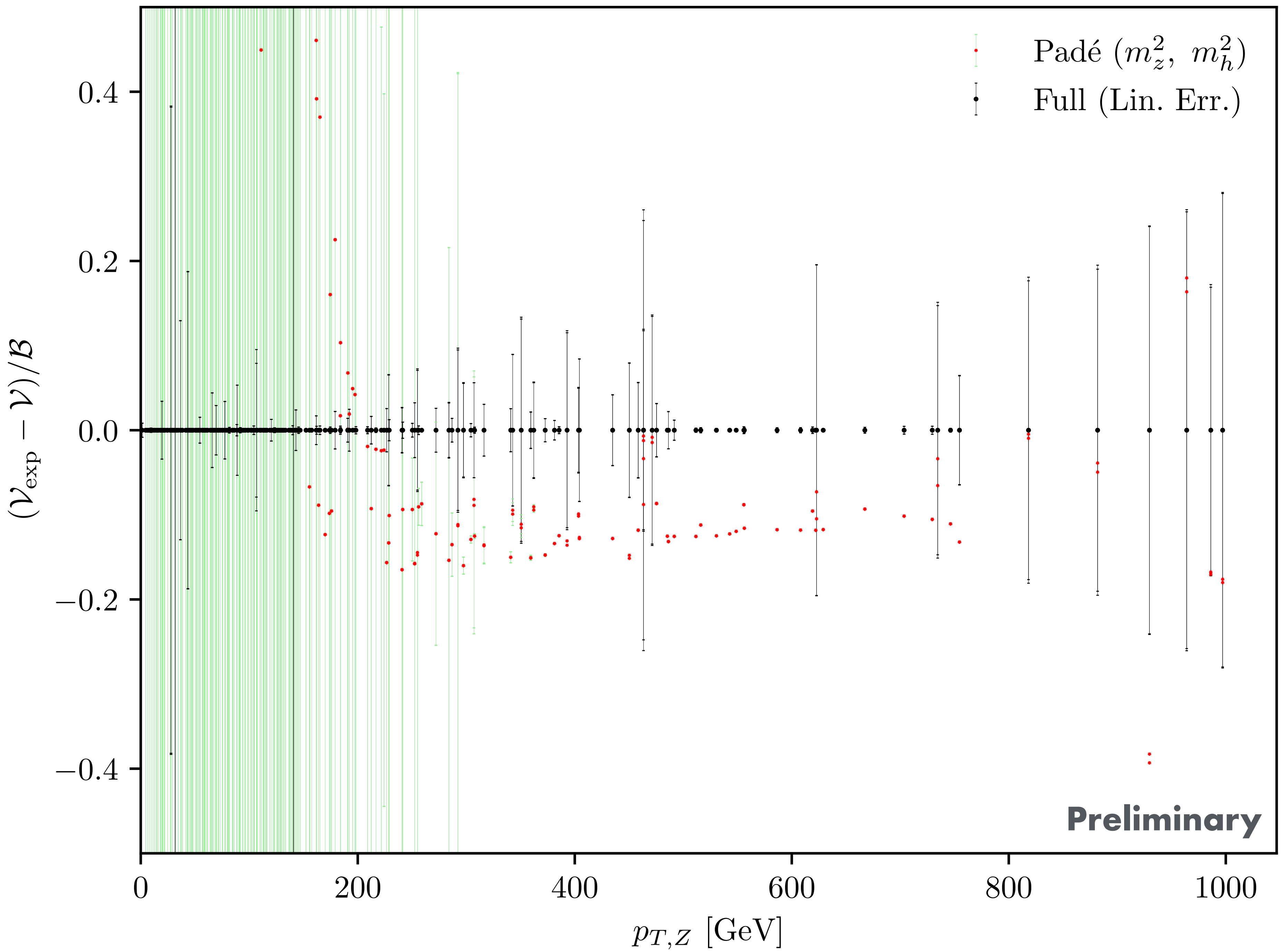
Consistent with Padé/full at LO level

Not all points agreeing well even for large $\sqrt{s}$

Convergence of the Padé result depends on $m_T \ll s, |t|, |u|$ can have small $|t|$ even for large s (if p_T is small)

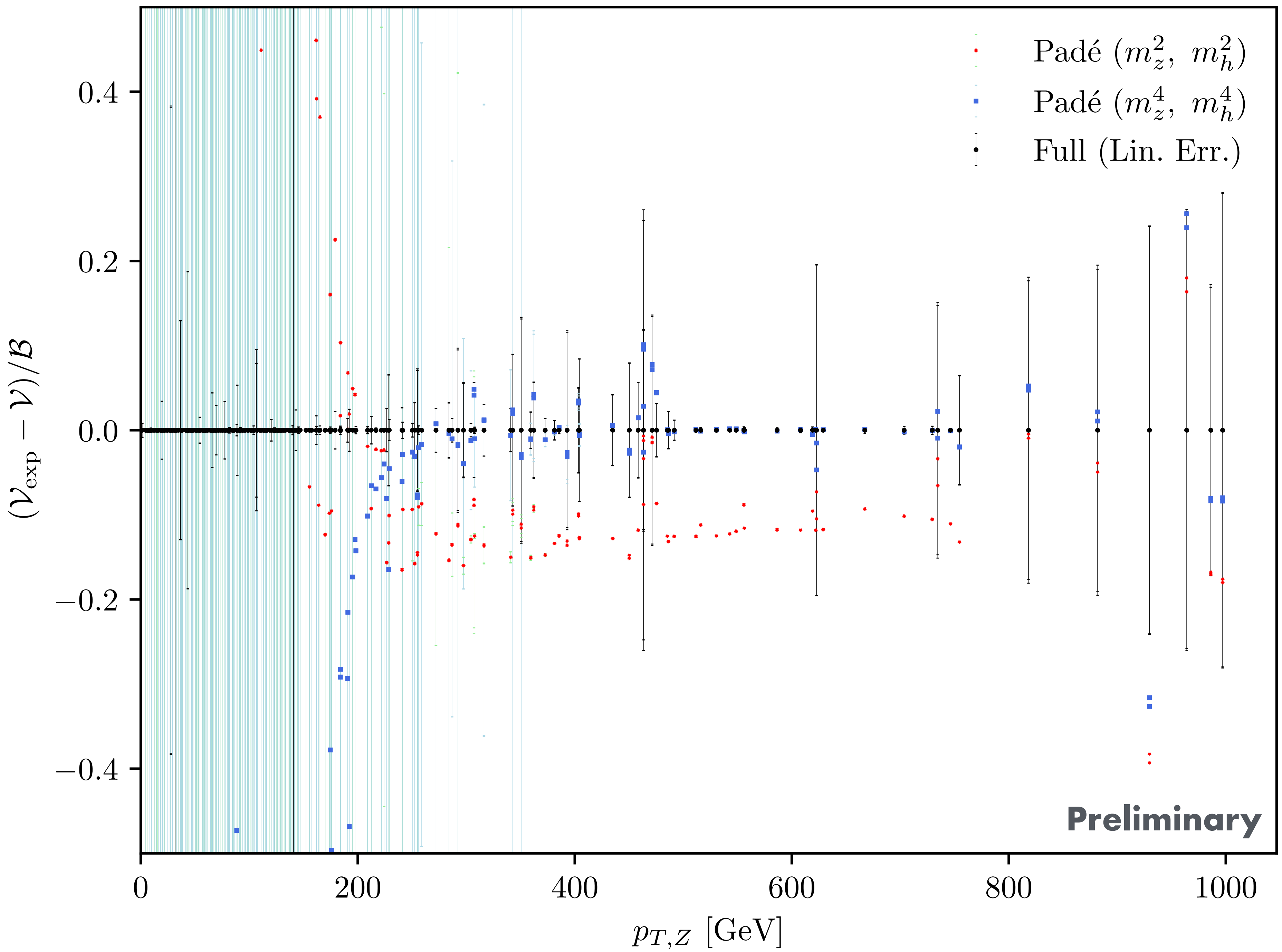
Comparison to Small m_t Expansion (II)

Can the Padé result be improved by including terms of order m_z^4, m_h^4?



Comparison to Small m_t Expansion (II)

Can the Padé result be improved by including terms of order m_z^4, m_h^4?



New Padé terms provided by: Davies, Mishima, Steinhauser

Now find **excellent** agreement for $p_T \geq 200$ GeV

Comparison to Expansion (Small m_h, m_z)

Can expand in only m_h, m_z and retain full m_t dependence Wang, Xu, Xu, Yang 21

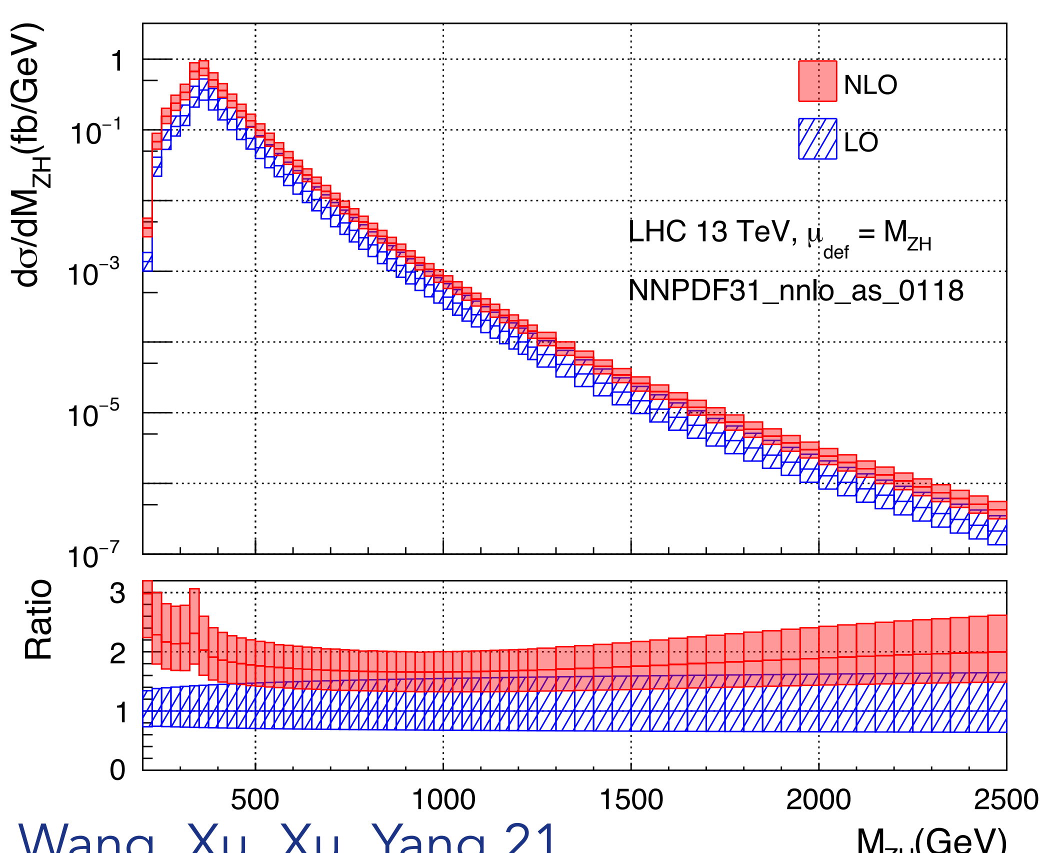
Integrals appearing in the expansion (scales s, t, m_t^2) are known

Caron-Huot, Henn 14; Becchetti, Bonciani 18; Xu, Yang 18; Wang, Wang, Xu, Xu, Yang 20;

Expansion shows good agreement with numerical result in most (all?) phase-space regions

No breakdown near top threshold

$\hat{s}/m_t^2$	$\hat{u}/m_t^2$	$\mathcal{V}'_{\text{fin}}$			
		pySecDec	$\mathcal{O}(m^0)$	$\mathcal{O}(m^2)$	$\mathcal{O}(m^4)$
1.707133657190554	−0.441203767016323	35.429092(6)	35.9823	35.5530	35.4478
3.876056604162662	−1.616287256345735	4339.045(1)	4319.37	4336.63	4338.73
4.130574250302561	−1.750372271104745	6912.361(3)	6870.47	6906.92	6911.64
4.130574250302561	−2.595461551488002	6981.09(2)	6979.28	6980.14	6980.85
134.5142052093564	−70.34125943305149	−153.9(4)	−154.543	−154.458	−154.460
134.5142052093564	−105.1770655376327	527(4)	524.585	525.958	525.965



Wang, Xu, Xu, Yang 21

Authors published NLO results for $gg \to ZH$

Virtuals: small m_h, m_z expansion

Reals: GoSam Cullen et al.

Gives stable invariant mass distribution, sizeable corrections ~LO (as expected)

Conclusion

We have entered the precision Higgs era

- Over coming years, expect greater demand for precise theory predictions (required to exploit experimental measurements)
- Detailed searches for deviations, e.g: differential measurements, constraining EFT couplings, off-shell measurements, …

I have presented a calculation which underscores the usefulness of

- Numerical methods for solving Feynman integrals **(new pySecDec out now!)**

Next steps…

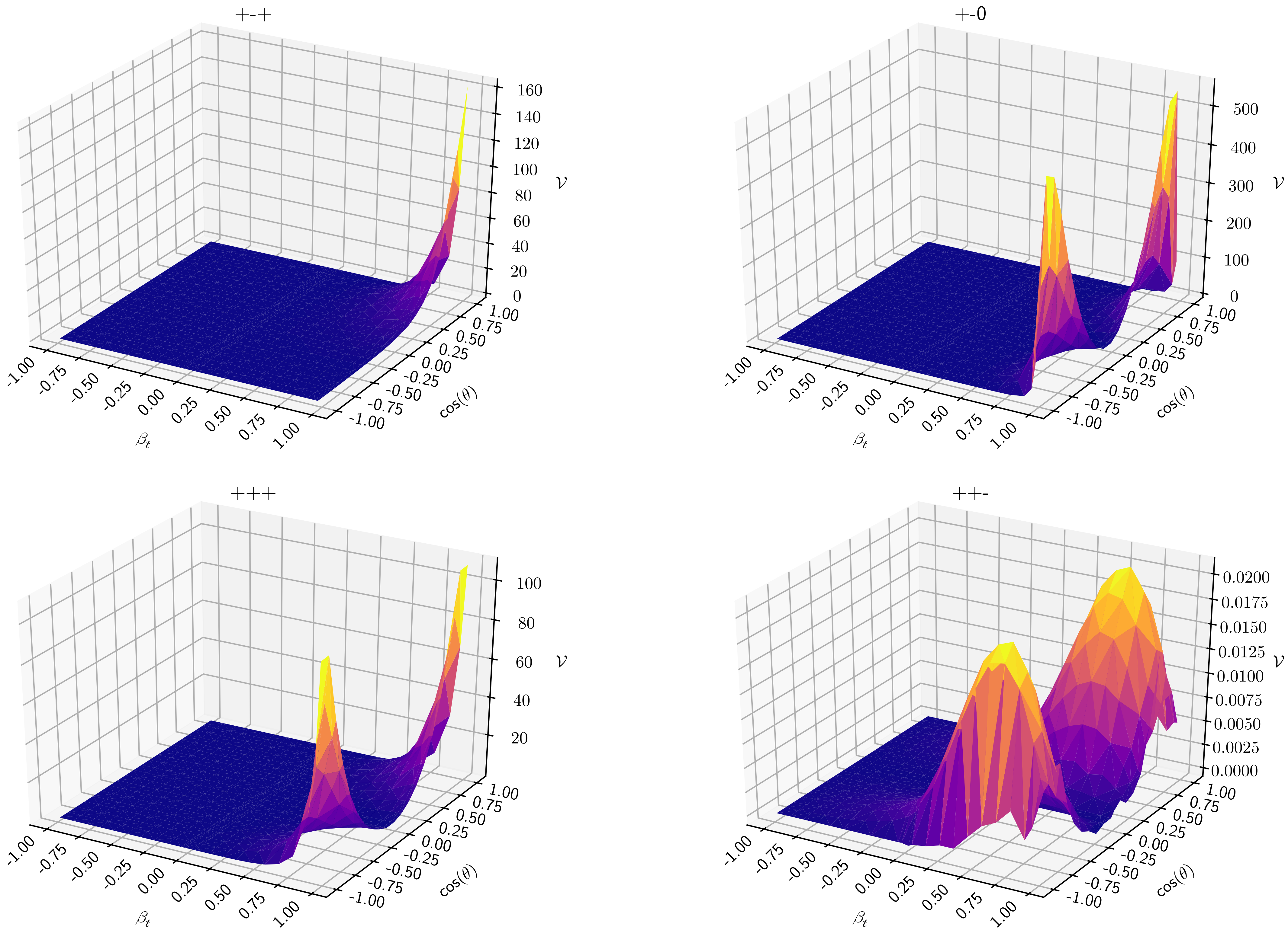
- Put the pieces together in order to obtain complete NLO results
- Incorporate into public tools for $pp \to ZH$ (?)

Thank you for listening!

Backup

Helicity Amplitudes

We can produce precise results for all helicity amplitudes also in kinematic limits!



Decomposition: $gg \to ZH$

Idea: construct projectors for linearly polarised amplitudes in c.o.m frame, directly compute polarised amplitudes Chen 19

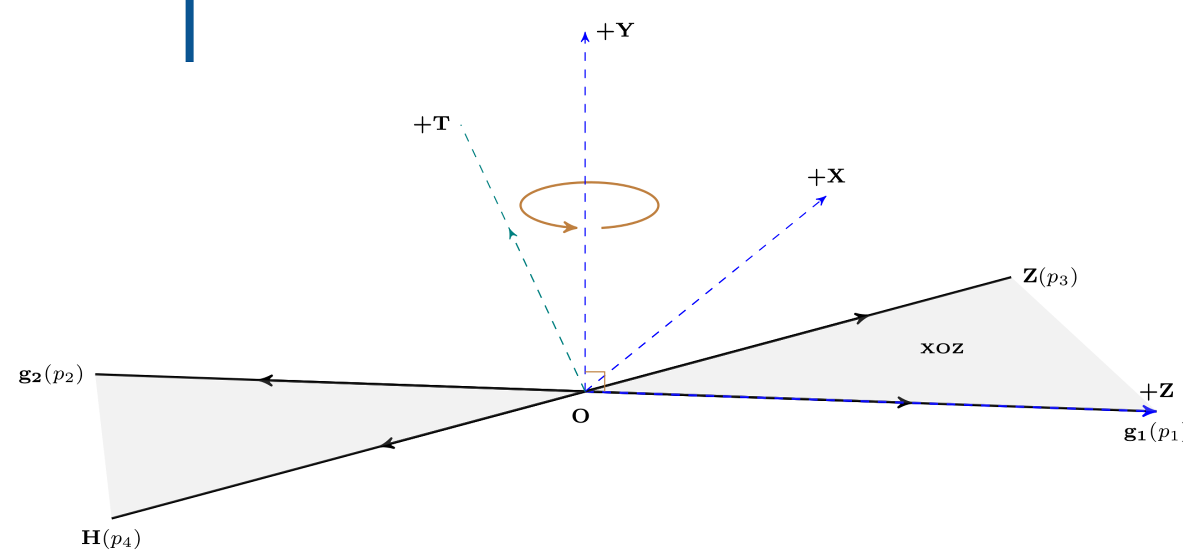
Polarisation vectors can be expressed (up to normalisation factors $\mathcal{N}_i$) in terms of external momenta:

$$
\begin{aligned}
\varepsilon_x^\mu =&\mathcal{N}_x \left(-s_{23}p_1^\mu - s_{13}p_2^\mu + s_{12}p_3^\mu\right)\\
\varepsilon_y^\mu =&\mathcal{N}_y \left(\epsilon^\mu_{\mu_1\,\mu_2\,\mu_3}\, p_1^{\mu_1}\, p_2^{\mu_2}\, p_3^{\mu_3}\right)\\
\varepsilon_T^\mu =&\mathcal{N}_T \left(\left(-s_{23}(s_{13}+s_{23}) + 2m_z^2 s_{12}\right) p_1^\mu + \right.\\
&\left.\left(s_{13}(s_{13}+s_{23}) - 2m_z^2 s_{12}\right) p_2^\mu + s_{12}(-s_{13}+s_{23})\, p_3^\mu\right)\\
\varepsilon_l^\mu =&\mathcal{N}_l \left(-2m_z^2\,(p_1^\mu + p_2^\mu) + (s_{13}+s_{23})\, p_3^\mu\right)
\end{aligned}
$$

$$
\begin{aligned}
\varepsilon_x \cdot \{p_1, p_2\} &= 0\\
\varepsilon_y \cdot \{p_1, p_2\} &= 0\\
\{\varepsilon_y, \varepsilon_T, \varepsilon_l\} \cdot p_3 &= 0\\
\varepsilon_i^2 &= -1
\end{aligned}
$$

Projectors are just products of pol. vecs.

$$
\begin{aligned}
\mathcal{P}_1^{\mu_1\mu_2\mu_3} &= \varepsilon_x^{\mu_1}\,\varepsilon_x^{\mu_2}\,\varepsilon_y^{\mu_3} & \mathcal{P}_2^{\mu_1\mu_2\mu_3} &= \varepsilon_x^{\mu_1}\,\varepsilon_y^{\mu_2}\,\varepsilon_T^{\mu_3},\\
\mathcal{P}_3^{\mu_1\mu_2\mu_3} &= \varepsilon_x^{\mu_1}\,\varepsilon_y^{\mu_2}\,\varepsilon_l^{\mu_3} & \mathcal{P}_4^{\mu_1\mu_2\mu_3} &= \varepsilon_y^{\mu_1}\,\varepsilon_x^{\mu_2}\,\varepsilon_T^{\mu_3},\\
\mathcal{P}_5^{\mu_1\mu_2\mu_3} &= \varepsilon_y^{\mu_1}\,\varepsilon_x^{\mu_2}\,\varepsilon_l^{\mu_3} & \mathcal{P}_6^{\mu_1\mu_2\mu_3} &= \varepsilon_y^{\mu_1}\,\varepsilon_y^{\mu_2}\,\varepsilon_y^{\mu_3}.
\end{aligned}
$$



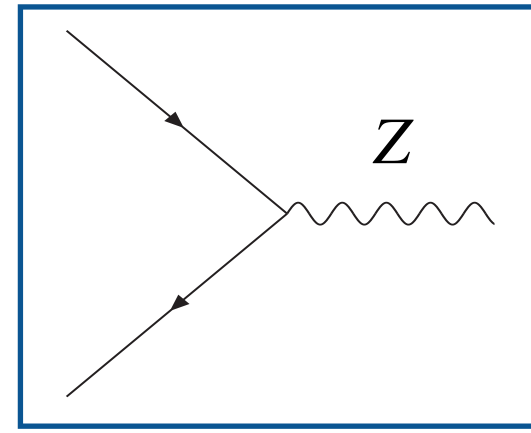
Cross checked with conventional form factor decomposition at LO and at NLO with expansions Davies, Mishima, Steinhauser 20

Dimensional Regularisation & γ_5

Z-Fermion Vertex

Contains vector $\sim v_t \gamma_\mu$ and axial-vector $\sim a_t \gamma_\mu \gamma_5$:

$$\mathcal{V}_\mu^{Vf\bar{f}} = i \frac{e}{2 \sin\theta_W \cos\theta_W} \gamma_\mu \left(v_t + a_t \gamma_5 \right)$$



We use dimensional regularisation ($d = 4 - 2\epsilon$) to regulate divergences appearing in loop integrals, however, one can't retain all properties of γ_5 in $d \neq 4$ dimensions

Larin Scheme (ZH, ZZ)

Sacrifice anti-commuting property of γ_5

$$J_\mu^5 = Z_{5,ns}\, J_{\mu,B}^5 = Z_{5,ns} \left[\frac{i}{3!} \epsilon_{\mu\nu\rho\sigma} \bar{\psi} \gamma^\nu \gamma^\rho \gamma^\sigma \bar{\psi} \right]$$

$$P^5 = Z_{5,p}\, P_B^5 = Z_{5,p} \left[\frac{i}{4!} \epsilon_{\mu\nu\rho\sigma} \bar{\psi} \gamma^\mu \gamma^\nu \gamma^\rho \gamma^\sigma \bar{\psi} \right]$$

Fix Ward identities/ABJ anomaly:

$$Z_{5,ns} = 1 + \alpha_s(-4C_F) + \ldots$$
$$Z_{5,p} = 1 + \alpha_s(-8C_F) + \ldots$$

Larin, Vermaseren 91; Larin 93

Alternative schemes exist e.g:

Kreimer Scheme (ZZ)

Retain $\{\gamma_5, \gamma^\mu\} = 0$, but, sacrifice cyclicity of traces involving γ_5

Define `reading point' and carefully manipulate all traces

Kreimer 90; Korner, Kreimer, Schilcher 92

Used in our calculation of $gg \to ZZ$

Agarwal, SPJ, von Manteuffel 20

Reduction

Integration by parts Identities: **loop/external momentum**

$$\int \mathrm{d}^d k_1 \cdots \mathrm{d}^d k_l \, \frac{\partial}{\partial k_i^\mu} \left[v^\mu \mathrm{I}(k_1, \ldots, k_l; p_1, \ldots, p_m) \right] = 0$$

Produce linear relations between integrals Tkachov 81; Chetyrkin 81

Can perform e.g. Gaussian elimination on system of equations

Relate integrals to a smaller set (**basis**) of **Master integrals**

The choice of basis impacts:

1) **Complexity of the coefficients in the amplitude**
2) **Difficulty of computing the integrals**

Always possible to pick a basis of finite integrals using:

Panzer 14; von Manteuffel, Panzer, Schabinger 15

- Dimension Shifts Tarasov 96; Lee 10
- Dots
- Numerator Insertions *(optional, not used for $gg \to ZH$)*

The finite basis greatly improves numerical performance

Reduction (II)

Need tools to actually solve these systems of equations…

ZZ Computation	**ZH Computation**
FinRed — von Mantueffel (Private) + Syzygy Solver — Agarwal, von Mantueffel	Kira 2 + FireFly — Maierhöfer, Usovitsch, Uwer 18; Klappert, Lange, P. Maierhöfer, Usovitsch 20; Klappert, Lange 20; Klappert, Klein, Lange 20
Master Integrals: 264	**Master Integrals:** 452

Both toolchains extensively rely on the use of finite fields
See e.g: von Mantueffel, Schabinger 14; Peraro 16

Even with these tools, still too difficult to obtain fully symbolic amplitudes

Fix mass ratios ZZ: $\frac{m_z^2}{m_t^2} = \frac{5}{18}$ and ZH: $\frac{m_z^2}{m_t^2} = \frac{23}{83}, \frac{m_H^2}{m_t^2} = \frac{12}{23}$

Master Integral Basis ($gg \to ZH$)

To select our master integrals, we took the following pragmatic approach:

1) Consider quasi-finite integrals (prefer finite integrals)

$$I = \frac{I_{-2}}{\epsilon^2} + \frac{I_{-1}}{\epsilon} + I_0 + \ldots \quad \rightarrow \quad I' = I'_0 + \ldots$$

2) Choose a basis in which the d-dependence of denominators factorises from the kinematic dependence (in practice we achieve this by brute force neglecting subsectors, public tools are available Smirnov, Smirnov 20; Usovitsch 20)

$$\frac{N(s,t,d)}{D(s,t,d)} I + \ldots \rightarrow \frac{N'(s,t,d)}{D'_1(d) D'_2(s,t)} I' + \ldots$$

3) Prefer simple denominator factors
4) Prefer computing fewer orders in epsilon for each master (found a basis in which all 7-propagator integrals start contributing only at ϵ^{-1})
5) Prefer simpler numerators (check number of terms/file size)

See also: Matthias Kerner, Loops and Legs Proceedings 2018

Steps 2-5 reduced the size of amplitude by factor of 5

Largest coefficient (double-tadpole) 150 MB → 5 MB

Amplitude Evaluation (IV)

A peak behind the curtain (enabled by setting verbose=True)

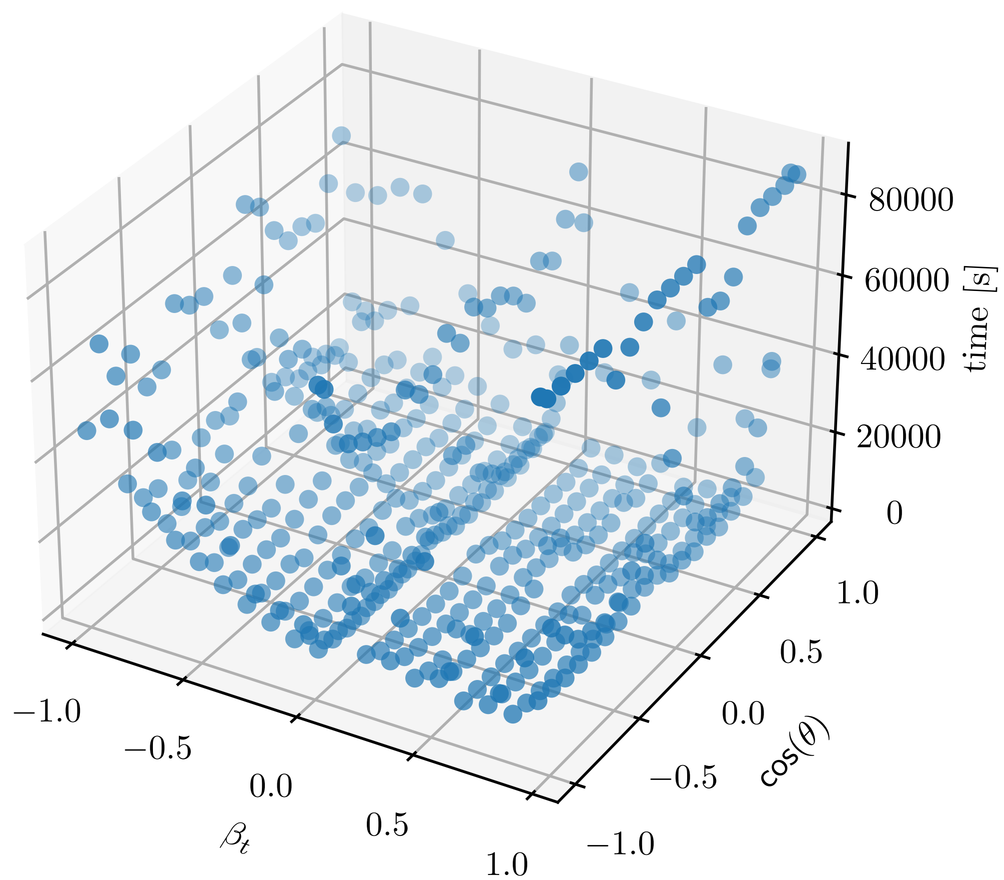
```
#
# Compute a first estimate of each integral (n: 50789 x 32 samples of integral)
# Note: each order of each sector is computed separately (and can be known to a different precision)
#
computing integrals to satisfy mineval 50000
integral 1/16: bubble_u_sector_1_order_0, time: 0.0 s  res: (0,0) +/- (0,0) -> (1,0) +/- (8.89904e-17,0), n: 0 -> 50789
integral 2/16: bubble_u_sector_1_order_1, time: 0.0 s  res: (0,0) +/- (0,0) -> (2.22314,0) +/- (1.9927e-15,0), n: 0 -> 50789
integral 3/16: bubble_t_sector_1_order_0, time: 0.0 s  res: (0,0) +/- (0,0) -> (1,0) +/- (9.00691e-17,0), n: 0 -> 50789
integral 4/16: bubble_t_sector_1_order_1, time: 0.0 s  res: (0,0) +/- (0,0) -> (1.73764,0) +/- (2.19938e-15,0), n: 0 -> 50789
...

#
# Estimate amplitude(s) using integral results
#
amplitude0 =  + ((0,0) +/- (2.41174e-16,0))*eps^-1 + ((-28.4316,-1.6741e-09) +/- (4.06609e-09,2.39249e-09)) + O(eps)

#
# Examine contribution of each integral to err. of amp. and estimate how many samples we need (n: known -> required)
# Note: not all sectors will be recomputed, different sectors will be computed to different precisions
#
sum: sum_eps^0, term: WINTEGRAL, integral 5: box_6_sector_1_order_0, current integral result: (0.478203,4.5158e-11) +/-
(2.31205e-10,4.72335e-11), contribution to sum error: 2.0946e-09, increase n:  50789 -> 968960972
sum: sum_eps^0, term: WINTEGRAL, integral 15: box_8_sector_5_order_0, current integral result: (0.0173611,-5.39441e-13) +/-
(3.16251e-12,1.29608e-12), contribution to sum error: 1.64054e-10, increase n: 50789 -> 193374172
…

#
# Iterate until we obtain the desired precision
# Note: reason for iteration is printed (long running/run away jobs can be debugged straightforwardly)
#
run further refinements: true
computing integrals to satisfy error goals on sums: epsrel 1e-14, epsabs 1e-14
...
```

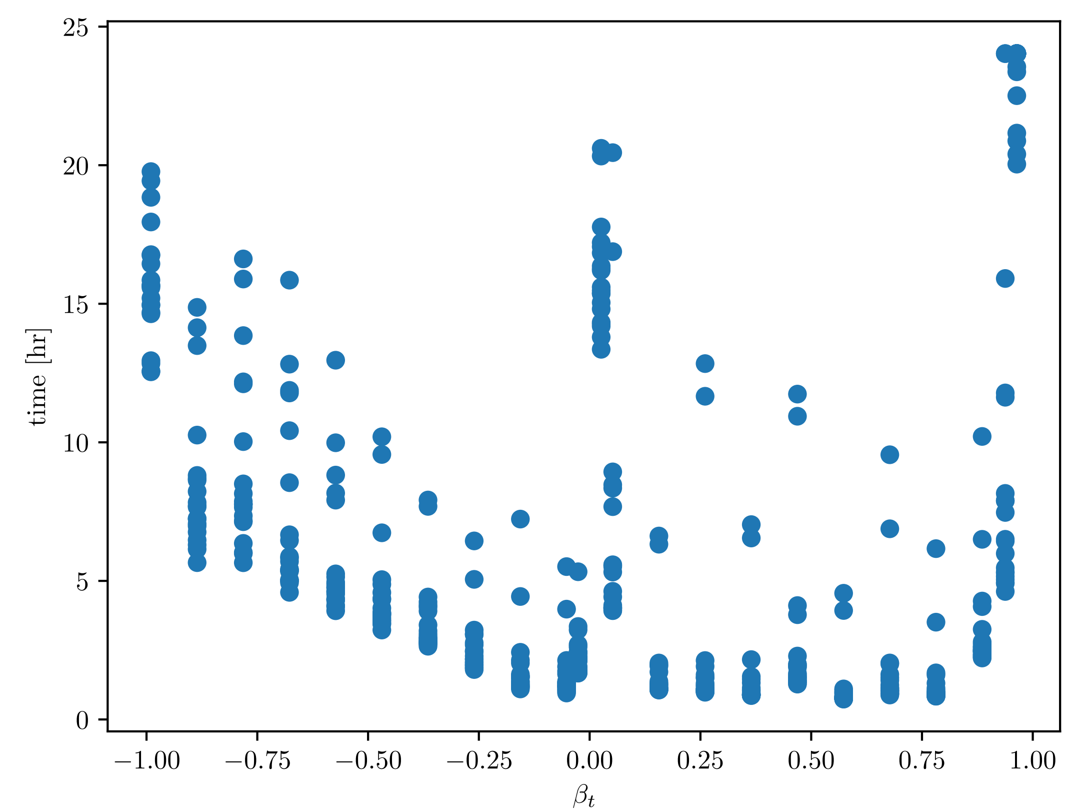
Evaluation of the Amplitude (Timing)

Each phase-space point evaluated with 2 x Nvidia Tesla V100 GPUs
Precision goal set to 0.3% for each (linearly polarised) amplitude

Timing/ point:

Min: 45 mins, **Max:** 24 hr (wall-clock), ~65 hr (high-energy), **Median:** 3.5 hr





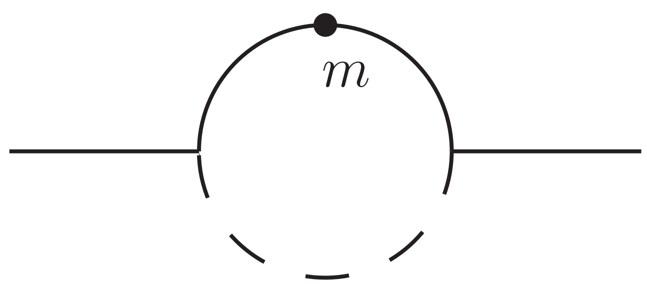
Worst performance near to ZH, $t\bar{t}$ thresholds, high-energy and forward scattering

Expansion by Regions: Momentum Space

Expansion by Regions: Bubble Example

Let us consider the large momentum limit $|p|^2 \gg m^2$ of some integral, we therefore want to expand in the small dimensionless ratio m^2/p^2

Example: (finite) 1-loop bubble integral Jantzen 2011



$D = 4 - 2\epsilon$

$$G = \mu^{2\epsilon} \int \frac{\mathrm{d}^D k}{i\pi^{D/2}} \frac{1}{(k+p)^2(k^2-m^2)^2}$$

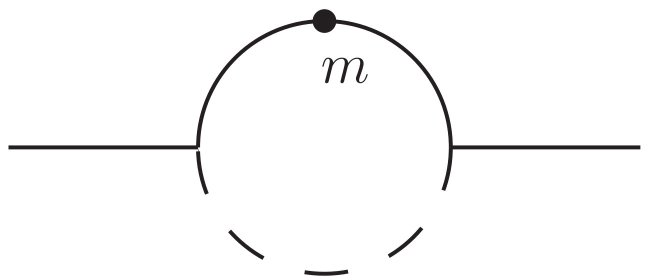
If we can compute the integral, the expansion after integration is straightforward

$$G = \frac{1}{p^2}\left[\ln\left(\frac{-p^2-i0}{m^2}\right) + \ln\left(1-\frac{m^2}{p^2}\right)\right] + \mathcal{O}(\epsilon)$$

$$= \frac{1}{p^2}\left[\ln\left(\frac{-p^2-i0}{m^2}\right) - \sum_{j=1}^{\infty}\frac{1}{j}\left(\frac{m^2}{p^2}\right)^j\right] + \mathcal{O}(\epsilon)$$

Now let's try to expand before integration

Expansion by Regions: Bubble Example (II)



$$G = \int \mathrm{D}k\, I = \mu^{2\epsilon} \int \frac{\mathrm{d}^D k}{i\pi^{D/2}} \frac{1}{(k+p)^2(k^2-m^2)^2}$$

Since $|p^2| \gg m^2$ we may start by expanding the 2nd propagator around small m^2:

$$I^{(h)} = \sum_i T_i^{(h)} I = \frac{1}{(k+p)^2}\left(\frac{1}{(k^2)^2} + 2\frac{m^2}{(k^2)^3} + \ldots\right)$$

Integrating the expansion over the **whole domain** $k \in \mathbb{R}^d$ we get:

$$G^{(h)} = \sum_i T_i^{(h)} G = \frac{1}{p^2}\left[-\frac{1}{\epsilon} + \ln\left(\frac{-p^2 - i0}{\mu^2}\right) - \sum_{j=1}^{\infty} \frac{2}{j}\left(\frac{m^2}{p^2}\right)^j\right] + \mathcal{O}(\epsilon)$$

Here, we implicitly assumed $k \gg m^2$ and neglected the region where $k \sim m$, let us compute this region too…

Expansion by Regions: Bubble Example (III)

Expanding the 1st propagator around large p^2:

$$I^{(s)} = \sum_i T_i^{(s)} I = \frac{1}{(k^2 - m^2)^2} \left(\frac{1}{p^2} - \frac{k^2 + 2p \cdot k}{(p^2)^2} + \dots \right)$$

Integrating the expansion over the **whole domain** $k \in \mathbb{R}^d$ we get:

$$G^{(s)} = \sum_i T_i^{(s)} G = \frac{1}{p^2} \left[\frac{1}{\epsilon} + \ln\left(\frac{\mu^2}{m^2}\right) + \sum_{j=1}^{\infty} \frac{1}{j} \left(\frac{m^2}{p^2}\right)^j \right] + \mathcal{O}(\epsilon)$$

Summing the **hard (h)** and **soft (s)** regions we get:

$$G = G^{(h)} + G^{(s)} = \frac{1}{p^2} \left[\ln\left(\frac{-p^2 - i0}{m^2}\right) - \sum_{j=1}^{\infty} \frac{1}{j} \left(\frac{m^2}{p^2}\right)^j \right] + \mathcal{O}(\epsilon)$$

This reproduces the expanded result, but why does this work?

1) Did we not **double-count** when replacing $\int Dk \rightarrow \int \mathrm{D}k I^{(h)} + \int \mathrm{D}k I^{(s)}$?

2) How do we **choose the regions**?

Expansion by Regions: Bubble Example (IV)

1) Did we not double-count when replacing $\int Dk \to \int \mathrm{D}k I^{(h)} + \int \mathrm{D}k I^{(s)}$ **?**

This example (and several others) was examined in great detail by Jantzen, he noted:

Jantzen 2011

The expansions $\sum_i T_i^{(h)}, \sum_i T_i^{(s)}$ converge absolutely in their respective domains

$$D_h = \{k \in \mathbb{R}^d : |k^2| \geq \Lambda^2\}$$
$$D_s = \{k \in \mathbb{R}^d : |k^2| < \Lambda^2\}$$
with $m^2 \ll \Lambda^2 \ll |p^2|$

The expansions commute $T_i^{(h)} T_j^{(s)} I = T_j^{(s)} T_i^{(h)} I = T_{i,j}^{(h,s)} I$

Thus $G = \sum_i \int_{k \in D_h} Dk\, T_i^{(h)} I + \sum_j \int_{k \in D_s} Dk\, T_j^{(s)} I = G^{(h)} + G^{(s)} - G^{(h,s)}$

Finally, the overlap/multiple expansion contribution $G^{(h,s)}$ turns out to be **scaleless (=0),** in dimensional regularisation such integrals vanish, **we did not double-count**!

Expansion by Regions in pySecDec

Geometric Method: Scaleless Integrals

Momentum space

In dimensional regularisation, **scaleless integrals are 0**

$$G(\{k_i\}_a, \{ck_i\}_b) = c^{N_G} G(\{k_i\}) \implies G(k_i) = 0, \quad \{k_i\} = \{k_i\}_a \cup \{k_i\}_b$$

Where $G(\{k_i\})$ is a Feynman integral depending on loop momenta $\{k_i\}$, $c \neq 0$ and N_G is some scaling dimension

Feynman parameter space

$$(\mathscr{U} \times \mathscr{F})\left(c^{\mathbf{v}}\mathbf{x}\right) = c^N(\mathscr{U} \times \mathscr{F})(\mathbf{x}), \quad \mathbf{v} \neq n\mathbf{1}, \quad n \in \mathbb{R}$$

Geometrical viewpoint

For Δ built from $\mathscr{U} + \mathscr{F}$

$$\dim(\Delta) = \dim(\mathbf{x}) \iff G \,\text{scaleful}$$
$$\dim(\Delta) < \dim(\mathbf{x}) \iff G \,\text{scaleless}$$

This was also used in earlier works on EBR Pak, Smirnov 10

Geometric Method: Overlap Contributions

Jantzen showed that under some quite general conditions Jantzen 2011

$$G = \sum_{f \in F^+} G^{(f)} - \sum_{\{f_1,f_2\} \subset F^+}^{\langle F_c^+ + 1 \rangle} G^{(f_1,f_2)} + \ldots - (-1)^n \sum_{\{f_1,\ldots,f_n\} \subset F^+}^{\langle F_c^+ + 1 \rangle} G^{(f_1,\ldots,f_n)} + \ldots + (-1)^{N_c} \sum_{f' \in F_{\mathrm{nc}}^+} G^{(f',f_1,\ldots,f_{N_c})}$$

overlap contributions / multiple expansions

Where F_{c}^+ yield commuting expansions, F_{nc}^+ non-commuting expansions and the sums $\{f_1, \ldots\}$ run over subsets containing at most one region from F_{nc}^+

However, for **regulated integrals** (dim reg + analytic regulators) **these multiple expansions usually vanish**

We always neglect them in our code, but one can easily check this case-by-case

Reason: Consider Newton polytope with $\dim(\Delta) = \dim(\mathbf{x}) + 1$

- 1st expansion: picks terms forming a facet f_1, $\dim(f_1) = \dim(\mathbf{x}) \implies$ scaleful
- 2nd expansion: picks out terms that are an ``intersection'' between two facets $f_{1,2}$, $\dim(f_{1,2}) = \dim(\mathbf{x}) - 1 \implies$ scaleless (unless facets are parallel)

Geometric Method: Additional Regulators

Issue 1: EBR can introduce spurious singularities that are not regulated by dim reg.

Let ν be a vector of propagator powers, $D = 4 - 2\epsilon$ space-time dimensions
For each facet $f \in F_{\Delta}^{N-1}$ a singularity is present in G if

$$\langle \mathbf{v}_f, \nu \rangle + a_f \frac{D}{2} \leq 0$$

e.g: Schlenk 16

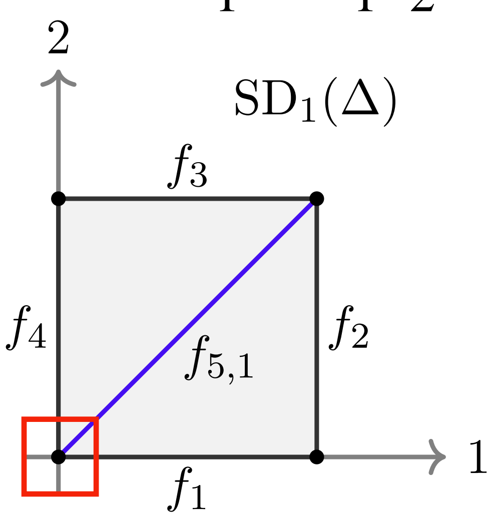
EBR will effectively subdivide the Newton polytope (by selecting only certain vertices for each expansion), this will introduce new **internal facets**

If these facets define a hyperplane that goes through $\mathbf{0}$ (i.e. $a_f = 0$) we can encounter spurious singularities, can introduce analytic regulators $\nu \to \nu + \delta \nu_\delta$ to regulate them

Examples:

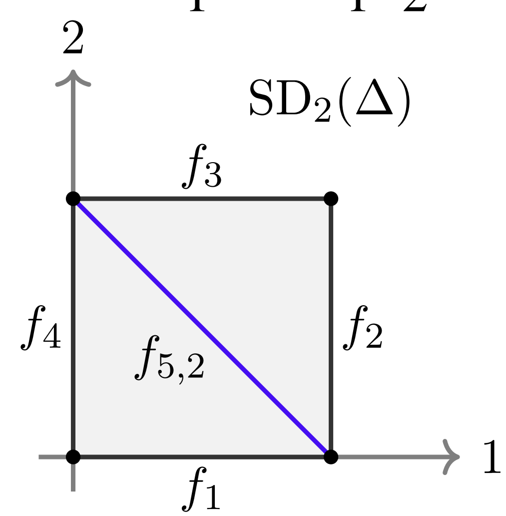
$$P_1(x,t) = 1 + tx_1 + x_1x_2 + tx_2$$

may need analytic regulator



$$P_2(x,t) = t + x_1 + tx_1x_2 + x_2$$

ok!



Geometric Method: Negative Coefficients

Issue 2: What happens if we have negative coefficients $c_i < 0$? ← **not handled by pySecDec (yet!)**

Consider a 1-loop massive bubble at threshold $y = m^2 - q^2/4 \to 0$

$$\mathscr{F} = q^2/4(x_1 - x_2)^2 + y(x_1 + x_2)^2$$

Can split integral into two subdomains $x_1 \le x_2$ and $x_2 \le x_1$ then remap

$$\begin{array}{l} x_1 = x_1'/2 \\ x_2 = x_2' + x_1'/2 \end{array} : \quad \mathscr{F} \to \frac{q^2}{4} x'^2_2 + y(x'_1 + x'_2)^2 \quad \text{(for first domain)}$$

Various tools attempt to find such re-mappings:

FIESTA Jantzen, A. Smirnov, V. Smirnov 12

Check all pairs of variables (x_1, x_2) which are part of monomials of opposite sign

For each pair, try to build linear combination x_1' s.t negative monomial vanishes

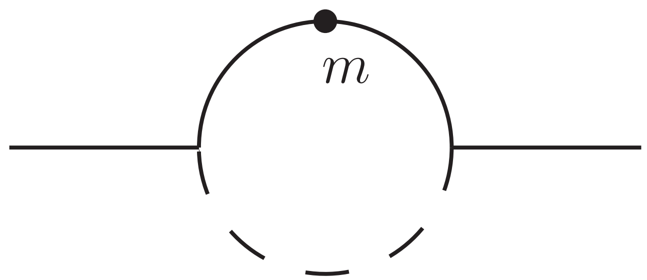
Repeat until all negative monomials vanish **or** warn user

ASPIRE Ananthanarayan, Pal, Ramanan, Sarkar 18; B. Ananthanarayan, Das, Sarkar 20

Consider Gröbner basis of $\{\mathscr{F}, \partial\mathscr{F}/x_1, \partial\mathscr{F}/x_2, \ldots\}$ (i.e. $\mathscr{F}$ and Landau equations)

Eliminate negative monomials with linear transformations $x_1 \to a x_1', x_2 \to x_2' + a x_1'$

pySecDec: EBR Bubble Example



$$G = \mu^{2\epsilon} \int \frac{\mathrm{d}^d k}{i\pi^{d/2}} \frac{1}{(k+p)^2(k^2-m^2)^2}$$

Step 1: Define Integral & Expand

```
from pySecDec import LoopIntegralFromPropagators, loop_regions, sum_package

# guard the code to allow multiprocessing inside pySecDec
if __name__ == "__main__":

    # define the loop integral for the case where we expand in msq
    li_m = LoopIntegralFromPropagators(
        propagators = ("(k+p)**2", "k**2-msq"),
        loop_momenta = ["k"],
        powerlist = [1, 2],
        regulators = ["eps"],
        replacement_rules = [("p*p", "psq")])

    # find the regions and expand the integrals using expansion
    # by regions
    sum_terms = loop_regions(
        name = "bubble1L_dotted_m",
        loop_integral = li_m,
        smallness_parameter = "msq",
        expansion_by_regions_order = 1)

    # generate code that will calculate the sum of all regions
    # and the requested orders in the smallness parameter
    sum_package("bubble1L_dotted_m", sum_terms,
        regulators = li_z.regulators,
        requested_orders = [0],
        real_parameters = ["psq", "msq"],
        complex_parameters = [])
```

Define dotted 1-loop bubble

Find regions assuming small msq/psq

Generate code

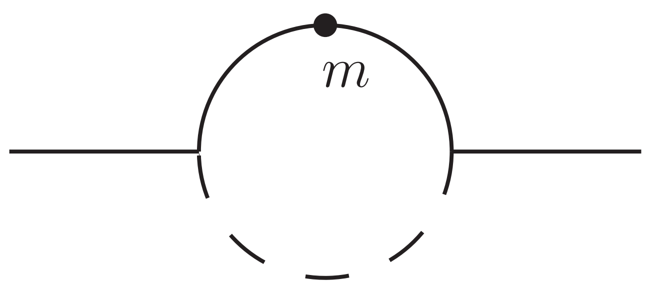
Output region vectors:

$\mathbf{v}_1 = (0, -1, 1)$

$\mathbf{v}_2 = (0, 0, 1)$

Also outputs a C++ code which can numerically compute the sum of the expanded integrals

pySecDec: EBR Example



$$G = \mu^{2\epsilon} \int \frac{\mathrm{d}^d k}{i\pi^{d/2}} \frac{1}{(k+p)^2(k^2-m^2)^2}$$

Step 2: Integrate

```
from pySecDec.integral_interface import IntegralLibrary, series_to_sympy
import sympy as sp

if __name__ == "__main__":

    psq, msq = 4, 0.002
    name = "bubble1L_dotted_m"
    real_parameters = [psq, msq]

    # load the library
    intlib = IntegralLibrary(f"{name}/{name}_pylink.so")
    intlib.use_Qmc(transform="korobov3")

    # integrate
    integral_without_prefactor, prefactor, integral_with_prefactor = \
        intlib(real_parameters)

    # convert the result to sympy expressions
    result, error = \
        map(sp.sympify, series_to_sympy(integral_with_prefactor))

    # access and print individual terms of the expansion
    print("Numerical Result")
    for power in [-2, -1, 0]:
        val = complex(result.coeff("eps", power))
        err = complex(error.coeff("eps", power))
        print(f"eps^{power:<2} {val: .5f} +/- {err:.5e}")
```

Step 3: Generate, Compile, Run

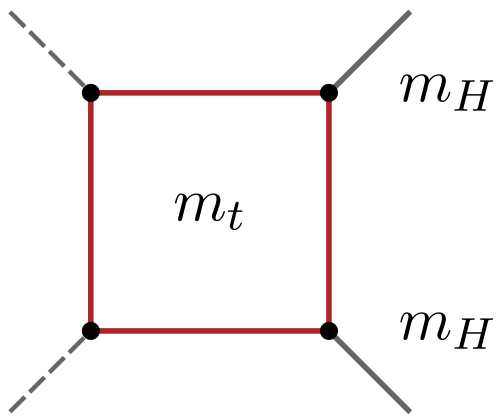
```
$ python3 generate_bubble1L_dotted_m_ebr.py
$ export CXX=nvcc
$ export SECDEC_WITH_CUDA_FLAGS="-gencode arch=compute_70,code=sm_70"
$ make -C bubble1L_dotted_m -j 8
$ time python3 integrate_bubble1L_dotted_m_ebr.py
> Numerical Result:
eps^-2  0.00000+0.00000j +/- 0.00000e+00+0.00000e+00j
eps^-1  0.00000+0.00000j +/- 1.87110e-17+0.00000e+00j
eps^0   1.90010-0.78540j +/- 8.70722e-17+1.03484e-16j
...

real    0m1.062s
user    0m2.175s
sys     0m0.270s
```

We have determined the Feynman parametrisation of our integral, applied expansion by regions and generated integrals which we then numerically computed

Of course, this works also for higher loop integrals…

pySecDec: EBR Box Example

Example: 1-loop massive box expanded for small $m_t^2 \ll s, |t|$



Requires the use of analytic regulators

Can regulate spurious singularities by adjusting propagators powers

$$G_4 = \mu^{2\epsilon} \int_{-\infty}^{\infty} \frac{d^D k}{i\pi^{D/2}} \frac{1}{[k^2 - m_t^2]^{\delta_1}[(k+p_1)^2 - m_t^2]^{\delta_2}[(k+p_1+p_2)^2 - m_t^2]^{\delta_3}[(k-p_4)^2 - m_t^2]^{\delta_4}}$$

Can keep $\delta_1, \ldots, \delta_4$ symbolic or $\delta_1 = 1 + n_1/2, \delta_2 = 1 + n_1/3, \ldots$ and take $n_1 \to 0^+$

Output region vectors:

$\mathbf{v}_1 = (0,0,0,0,1)$

$\mathbf{v}_2 = (-1,-1,0,0,1)$

$\mathbf{v}_3 = (0,0,-1,-1,1)$

$\mathbf{v}_4 = (-1,0,0,-1,1)$

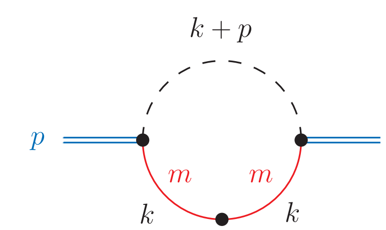
$\mathbf{v}_5 = (0,-1,-1,0,1)$

Result: s = 4.0, t = -2.82843, $m_t^2 = 0.1, m_h^2 = 0)$

$$I = -1.30718 \pm 2.7 \cdot 10^{-6} + (1.85618 \pm 3.0 \cdot 10^{-6})\, i$$

$$+\mathcal{O}\left(\epsilon, n_1, \frac{m_t^2}{s}, \frac{m_t^2}{t}\right)$$

Transform the expression for the full integral:



$$F = \int\limits_{k\in D_h} \mathrm{D}k\, I + \int\limits_{k\in D_s} \mathrm{D}k\, I = \sum_i \int\limits_{k\in D_h} \mathrm{D}k\, T_i^{(h)} I + \sum_j \int\limits_{k\in D_s} \mathrm{D}k\, T_j^{(s)} I$$

$$= \sum_i \Bigg(\int\limits_{k\in\mathbb{R}^d} \mathrm{D}k\, T_i^{(h)} I - \sum_j \int\limits_{k\in D_s} \mathrm{D}k\, T_j^{(s)} T_i^{(h)} I \Bigg) + \sum_j \Bigg(\int\limits_{k\in\mathbb{R}^d} \mathrm{D}k\, T_j^{(s)} I - \sum_i \int\limits_{k\in D_h} \mathrm{D}k\, T_i^{(h)} T_j^{(s)} I \Bigg)$$

The expansions commute: $\boxed{T_i^{(h)} T_j^{(s)} I = T_j^{(s)} T_i^{(h)} I \equiv T_{i,j}^{(h,s)} I}$

$\Rightarrow$ **Identity:** $F = \underbrace{\sum_i \int \mathrm{D}k\, T_i^{(h)} I}_{\boldsymbol{F^{(h)}}} + \underbrace{\sum_j \int \mathrm{D}k\, T_j^{(s)} I}_{\boldsymbol{F^{(s)}}} - \underbrace{\sum_{i,j} \int \mathrm{D}k\, T_{i,j}^{(h,s)} I}_{\boldsymbol{F^{(h,s)}}}$

All terms are integrated over the whole integration domain $\mathbb{R}^d$ as prescribed for the expansion by regions $\Rightarrow$ location of boundary Λ between D_h, D_s is irrelevant.

Slide from: **Bernd Jantzen, High Precision for Hard Processes (HP2) 2012**

The general formalism (details)

Identities as in the examples are generally valid, under some conditions.

Consider

- a (multiple) integral $F = \int \mathrm{D}k\, I$ over the domain D (e.g. $D = \mathbb{R}^d$),
- a set of N regions $R = \{x_1, \ldots, x_N\}$,
- for each region $x \in R$ an expansion $T^{(x)} = \sum_j T_j^{(x)}$ which converges absolutely in the domain $D_x \subset D$.

Conditions

- $\bigcup_{x \in R} D_x = D \qquad [D_x \cap D_{x'} = \emptyset \; \forall x \neq x']\,.$
- Some of the expansions commute with each other. Let $R_\mathsf{c} = \{x_1, \ldots, x_{N_\mathsf{c}}\}$ and $R_\mathsf{nc} = \{x_{N_\mathsf{c}+1}, \ldots, x_N\}$ with $1 \leq N_\mathsf{c} \leq N$. Then: $T^{(x)} T^{(x')} = T^{(x')} T^{(x)} \equiv T^{(x,x')} \; \forall x \in R_\mathsf{c}\,,\; x' \in R\,.$
- Every pair of non-commuting expansions is invariant under some expansion from R_c: $\forall x_1', x_2' \in R_\mathsf{nc},\; x_1' \neq x_2',\; \exists\, x \in R_\mathsf{c} : \; T^{(x)} T^{(x_2')} T^{(x_1')} = T^{(x_2')} T^{(x_1')}\,.$
- $\exists$ regularization for singularities, e.g. dimensional ($+$ analytic) regularization. $\hookrightarrow$ All expanded integrals and series expansions in the formalism are well-defined.

Slide from: Bernd Jantzen, High Precision for Hard Processes (HP2) 2012

The general formalism (2)

Under these conditions, the following **identity** holds: $\left[F^{(x,\ldots)} \equiv \sum_{j,\ldots} \int \mathrm{D}k\, T_{j,\ldots}^{(x,\ldots)} I\right]$

$$\boxed{F = \sum_{x\,\in\,R} F^{(x)} - \sum_{\{x'_1,x'_2\}\,\subset\,R}^{\langle R_c+1\rangle} F^{(x'_1,x'_2)} + \ldots - (-1)^n \sum_{\{x'_1,\ldots,x'_n\}\,\subset\,R}^{\langle R_c+1\rangle} F^{(x'_1,\ldots,x'_n)} + \ldots + (-1)^{N_c} \sum_{x'\,\in\,R_{nc}} F^{(x',x_1,\ldots,x_{N_c})}}$$

where the sums run over subsets $\{x'_1, \ldots\}$ containing at most one region from R_{nc}.

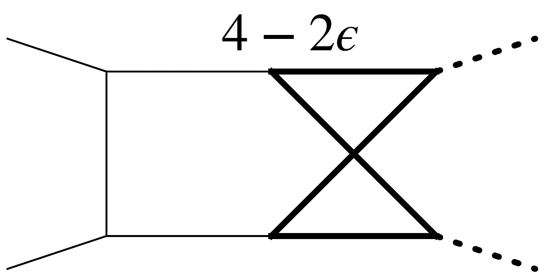
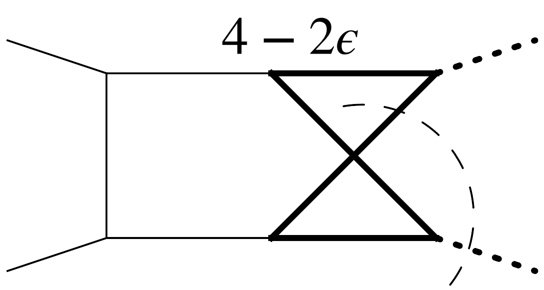
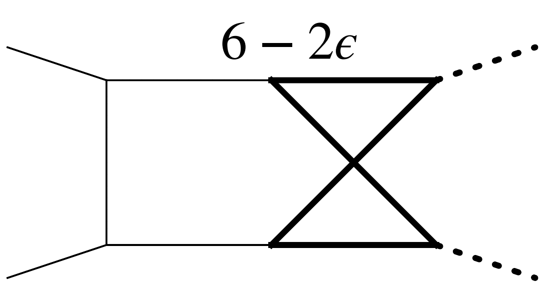
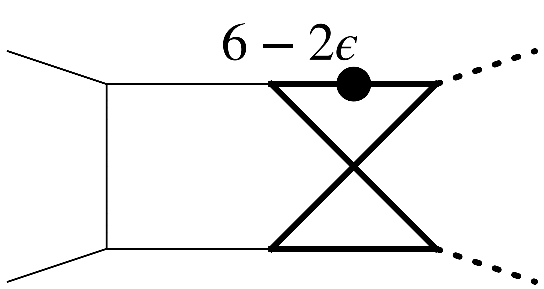
Comments

- This identity is exact when the expansions are summed to all orders. ✓
  Leading-order approximation for $F \rightsquigarrow$ dropping higher-order terms.
- It is independent of the regularization (dim. reg., analytic reg., cut-off, infinitesimal masses/off-shellness, ...) as long as all individual terms are well-defined.
- Usually regions & regularization are chosen such that multiple expansions $F^{(x'_1,\ldots,x'_n)}$ $(n \geq 2)$ are scaleless and vanish.
  [✓ if each $F_0^{(x)}$ is a *homogeneous* function of the expansion parameter with *unique scaling*.]
- If $\exists\, F^{(x'_1,x'_2,\ldots)} \neq 0 \rightsquigarrow$ relevant overlap contributions ($\rightarrow$ "zero-bin subtractions").
  They appear e.g. when avoiding analytic regularization in SCET. e.g. Manohar, Stewart '06; Chiu, Fuhrer, Hoang, Kelley, Manohar '09; ...

Slide from: Bernd Jantzen, High Precision for Hard Processes (HP2) 2012

Numerical Integration

Finite Integrals (example from $gg \to ZZ$)

	Finite	ϵ Order	Rel Err	Timing (s)
	No	0	$2 \cdot 10^{-3}$	45
	No	0	$4 \cdot 10^{-2}$	63
	Yes	1	$8 \cdot 10^{-6}$	60
	Yes	1	$8 \cdot 10^{-4}$	55
Linear combination (numerator insertion)	Yes	1	$1 \cdot 10^{-4}$	18

Figures & Timings: Bakul Agarwal

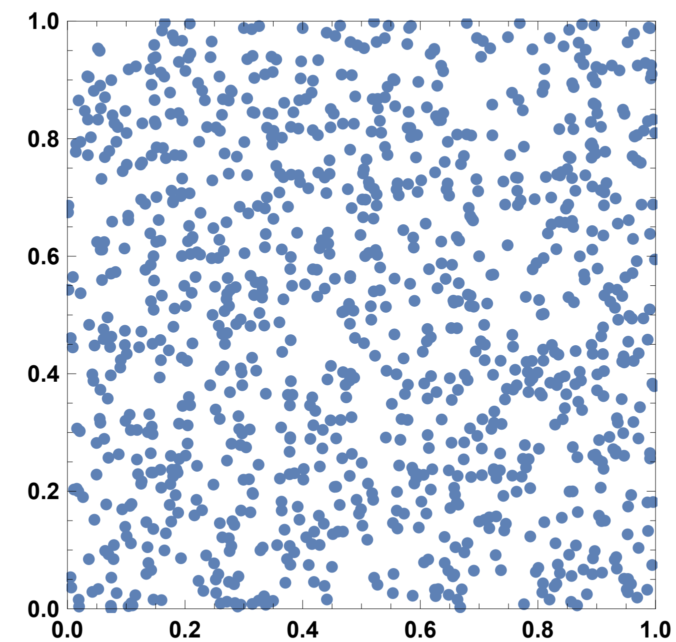
pySecDec + QMC
Nvidia Tesla V100

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

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

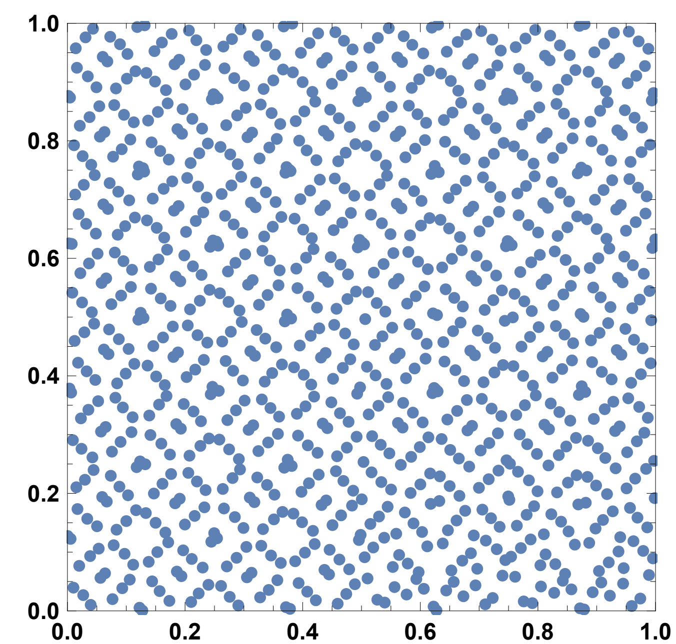


Monte Carlo:

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

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

Poor performance for large d

Both methods implemented in Cuba Hahn 04; Hahn14

Quasi-Monte Carlo (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

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

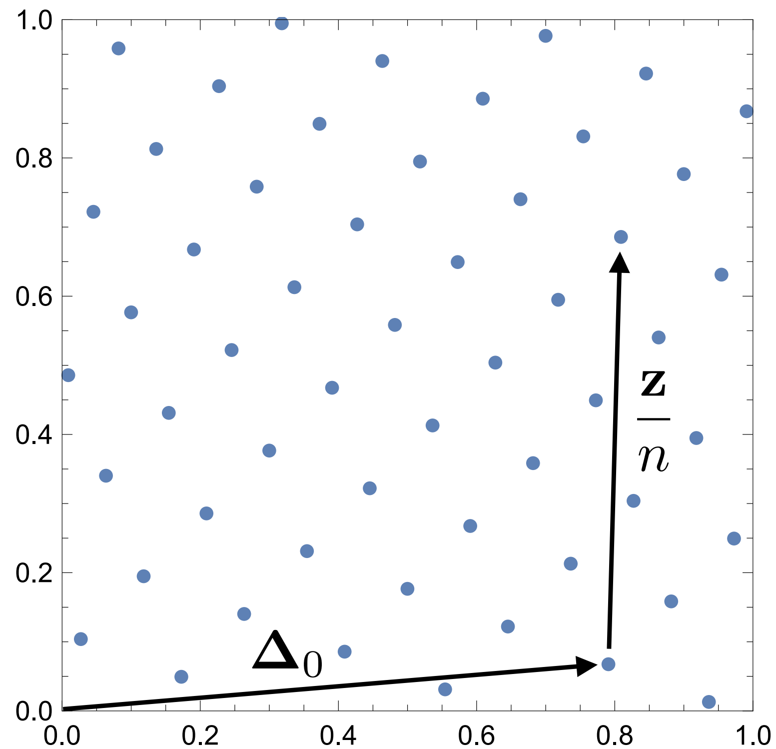
$\mathbf{z}$ - Generating vec.

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

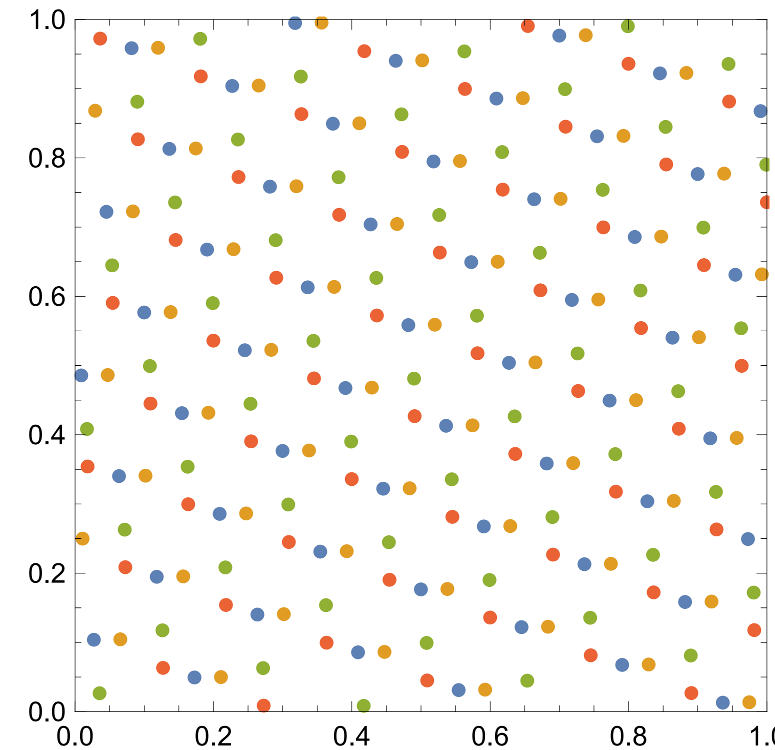
$\{\}$ - Fractional part

n - # Lattice points

m - # Random shifts



1 Shift



4 Shifts

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

Weighted Function Spaces

Assign weights $\gamma_{\mathfrak{u}}$ to each subset of dimensions $\mathfrak{u} \subseteq \{1, \ldots, d\}$ Review: Dick, Kuo, Sloan 13

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

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

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

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

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

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

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

Generating vector $\mathbf{z}$ precomputed for a **fixed** number of lattice points, chosen to minimise the worst-case error, we use component-by-component (CBC) construction

Nuyens 07

In our public code, we distribute lattice rules generated using product weights:
$\gamma_{\mathfrak{u}} = \prod_{i \in \mathfrak{u}} \gamma_i, \; \gamma_i = 1/d$ produced for a Korobov space with $\alpha = 2$

Periodising Transforms

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

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

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

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

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

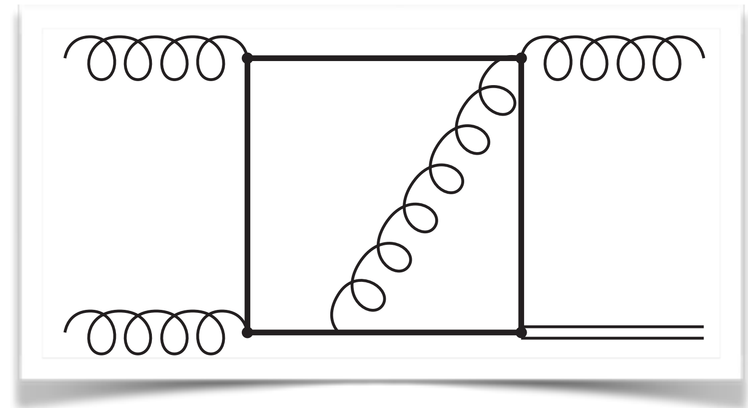
Baker transform: $\phi(u) = 1 - |2u - 1|$

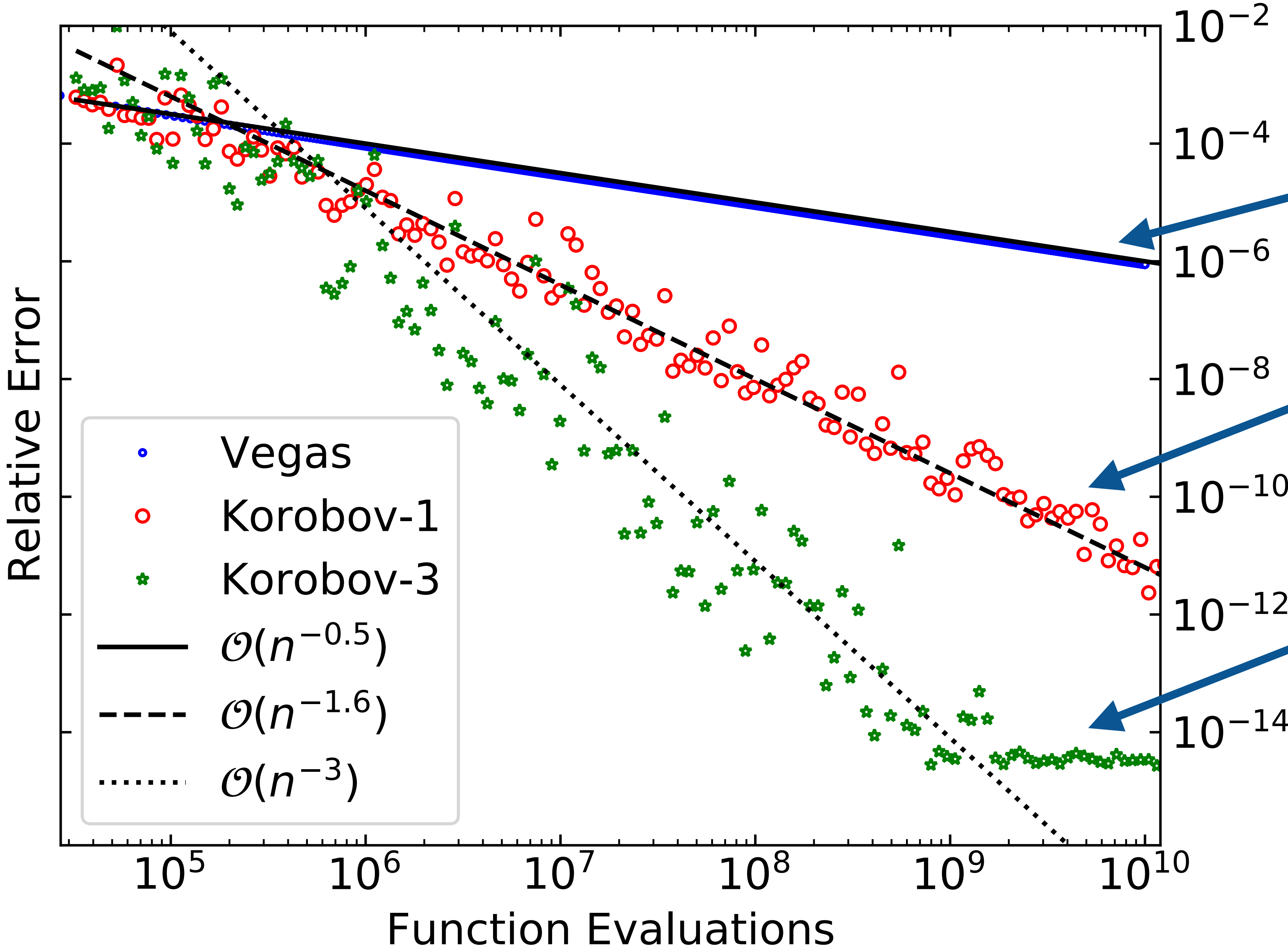
Scaling

Example:

Sector Decomposed HJ Integral

(Note: example stolen from previous calculation)





Monte Carlo scaling

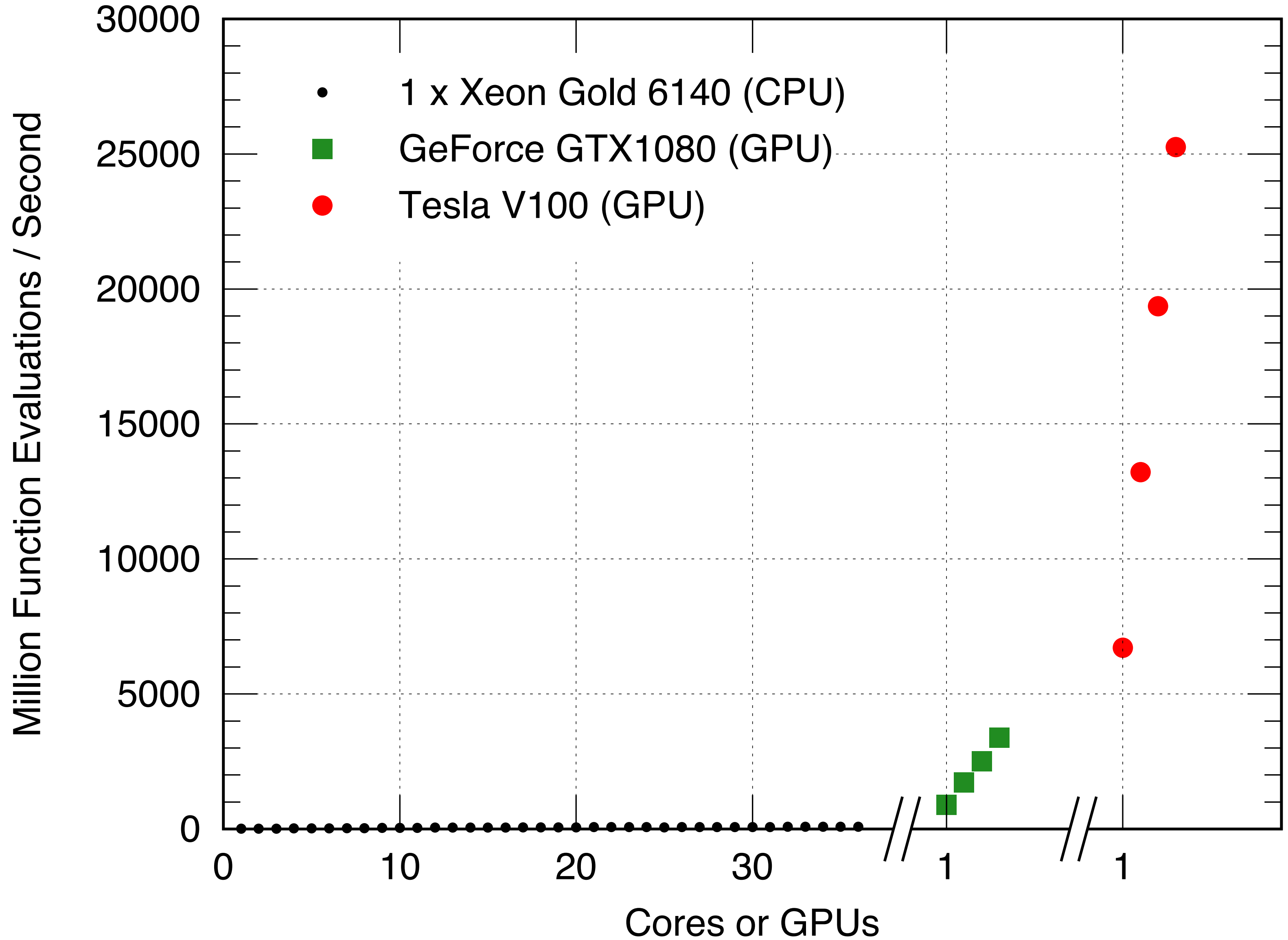
Better than ``guaranteed" n^{-1} scaling

Limited by machine precision (double)

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)