BlackHat Numerics

Darren Forde (CERN)

HO10 Workshop July 2010

In collaboration with C. Berger, Z. Bern, L. Dixon, F. Febres Cordero,
H. Ita, D. Kosower, D. Maître

Overview

- What is BlackHat.
- Brief overview of techniques used and their implementation.
- Speed and Numerical stability studies.

NLO Computation

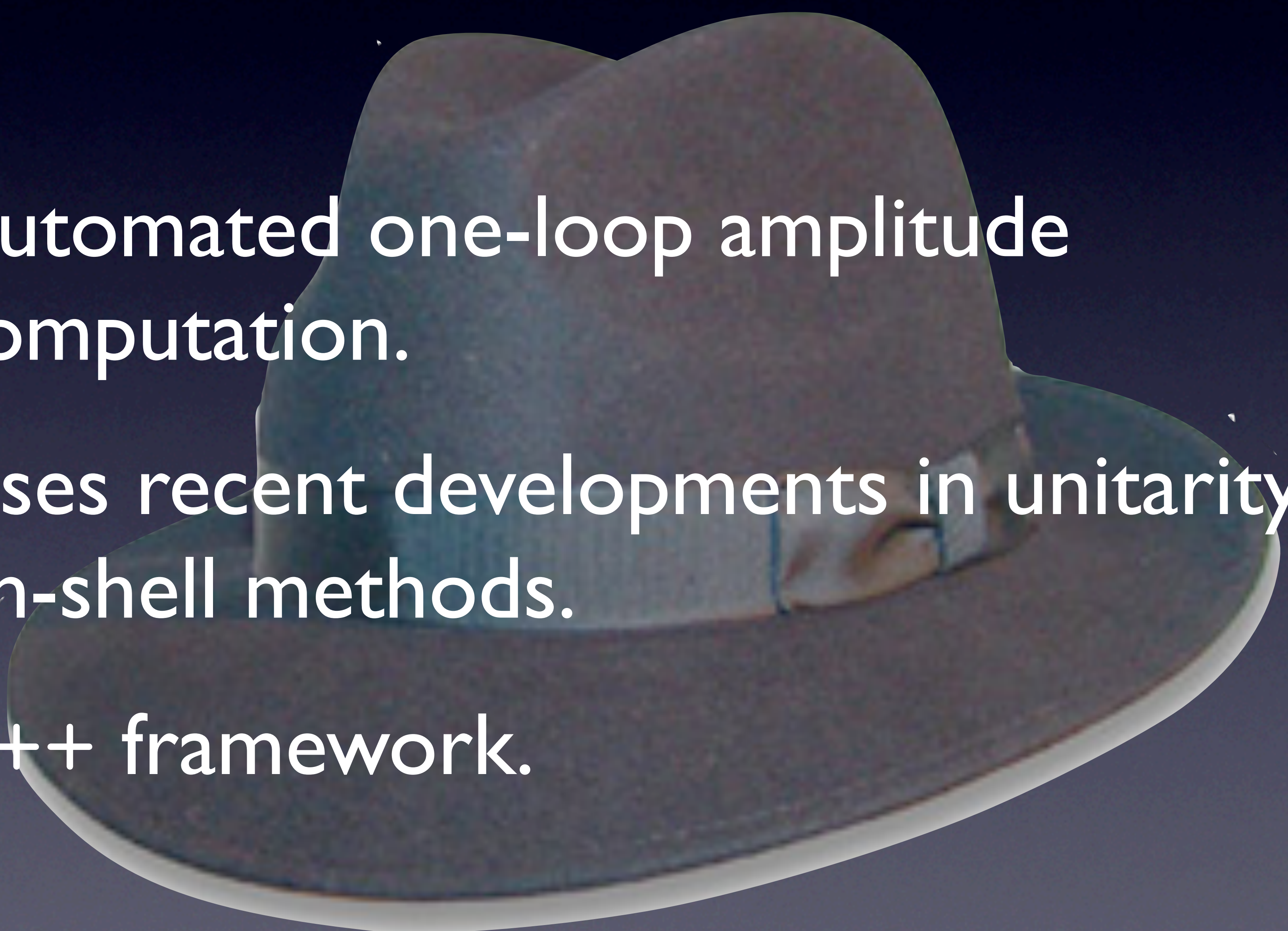
- Three pieces are needed for an NLO computation

$$\sigma_n^{\text{NLO}} = \int_n \sigma_n^{\text{tree}} + \int_n \sigma_n^{\text{virtual}} + \int_{n+1} \sigma_{n+1}^{\text{real}}$$

- I will focus on the virtual piece here.
- Specifically the automated One-loop computation package `BlackHat`.
- The remainder of the NLO computation is done with SHERPA. [Gleisberg, Hoeche, Krauss, Schoenherr, Schumann, Siegert, Winter]

BlackHat

[Berger, Bern, Dixon, DF, Febres Cordero, Ita, Kosower, Maître]



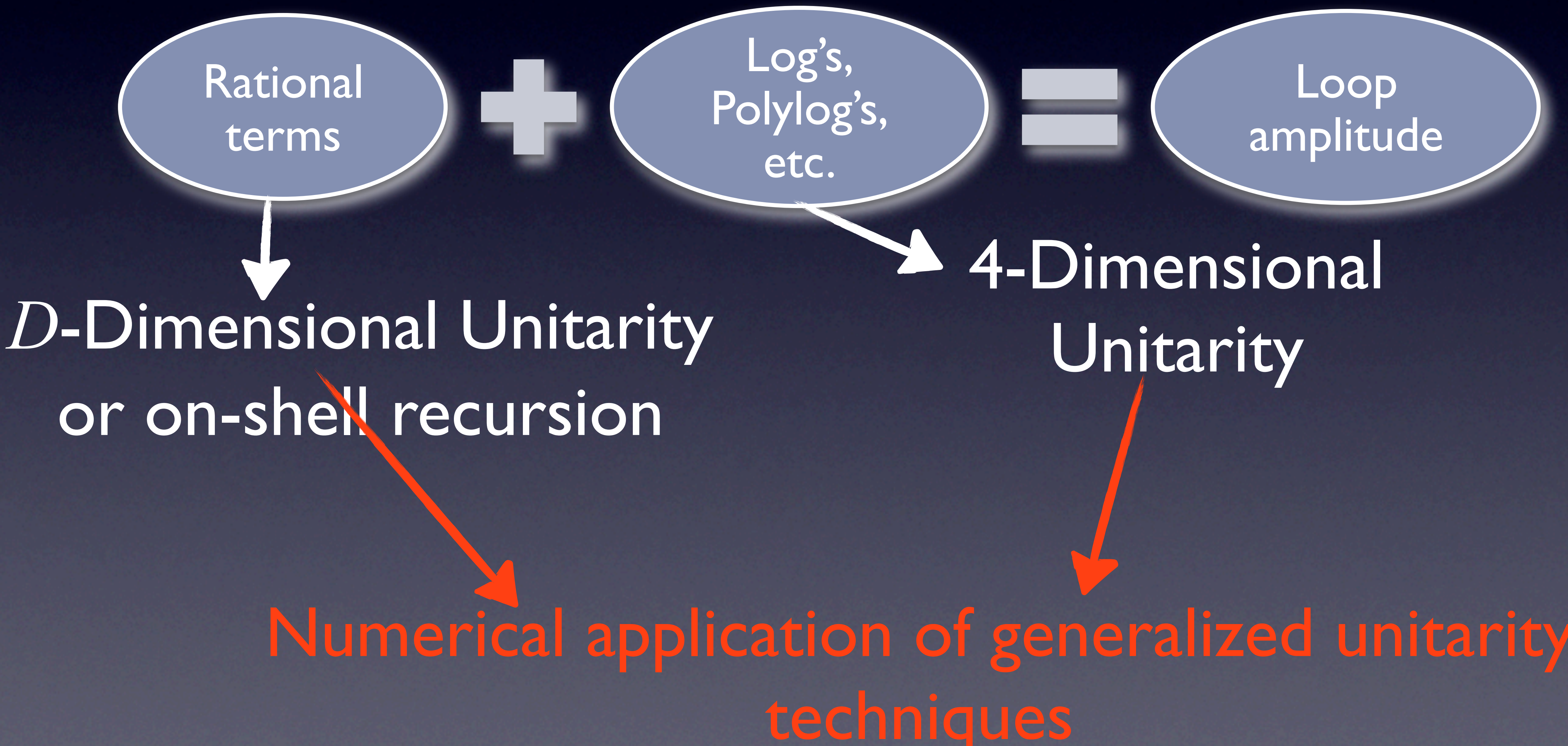
- Automated one-loop amplitude computation.
- Uses recent developments in unitarity & on-shell methods.
- c++ framework.

Typical Computing Load

- In an NLO computation we must compute the real and the virtual part, time spent is usually split evenly.
- Typically this means that we evaluate at (e.g. Z+3 jets at NLO)
  - Real: ~10^8 points.
  - Virtual: ~10^6 points.
- Virtual is about ~10^3 times slower per point.
- Typical running time for Z+3 jets for an LHC study is about a day on ~200 cores.
- Use n-tuples to reduce this computing load for further study.

Anatomy of a One-Loop Amplitude

- Split the computation of the amplitude into two parts, choose the optimal technique for each piece.



Generalized Unitarity

- Eliminates the need for tensor reductions.
- Performing a cut reduces the number of integrals.
  - e.g. for a triangle, using a specific loop momentum parameterisation we have, [DF]

$$\int dt \, \frac{\tilde{C}_{-3}}{t^3} + \frac{\tilde{C}_{-2}}{t^2} + \frac{\tilde{C}_{-1}}{t} + C_0 + t\tilde{C}_1 + t^2\tilde{C}_2 + t^3\tilde{C}_3 \qquad \frac{d}{\left(l(t) - K\right)^2}$$

- Remove the poles of the higher order terms (e.g. Boxes).
- Then extract the coefficient (e.g. C_0) from the series.

[Bern, Dixon, Kosower] [Britto, Cachazo, Feng] [DF]

Generalized Unitarity

- Eliminates the need for tensor reductions.
- Performing a cut reduces the number of integrals.
  - e.g. for a triangle, using a specific loop momentum parameterisation we have, [DF]

$$\int dt \, \frac{\tilde{C}_{-3}}{t^3} + \frac{\tilde{C}_{-2}}{t^2} + \frac{\tilde{C}_{-1}}{t} + C_0 + t\tilde{C}_1 + t^2\tilde{C}_2 + t^3\tilde{C}_3 \qquad \frac{d}{\left(l(t)-K\right)^2}$$

- Remove the poles of the higher order terms (e.g. Boxes).
- Then extract the coefficient (e.g. C_0) from the series.

[Bern, Dixon, Kosower] [Britto, Cachazo, Feng] [DF]

Numerical Direct Extraction

- Extract a particular coefficient of the integrand.

$$\int dt \frac{\tilde{C}_{-3}}{t^3} + \frac{\tilde{C}_{-2}}{t^2} + \frac{\tilde{C}_{-1}}{t} + C_0 + t\tilde{C}_1 + t^2\tilde{C}_2 + t^3\tilde{C}_3$$

- Extract using a discrete Fourier projection.

$$C_0 = \frac{1}{2p+1} \sum_{j=-p}^{p} A_1\left(t_0 e^{2\pi i j/(2p+1)}\right) A_2\left(t_0 e^{2\pi i j/(2p+1)}\right) A_3\left(t_0 e^{2\pi i j/(2p+1)}\right)$$

- Sample at least as many points as there are possible coefficients. This gives an exact result up to numerical stability issues.

Numerical Stability

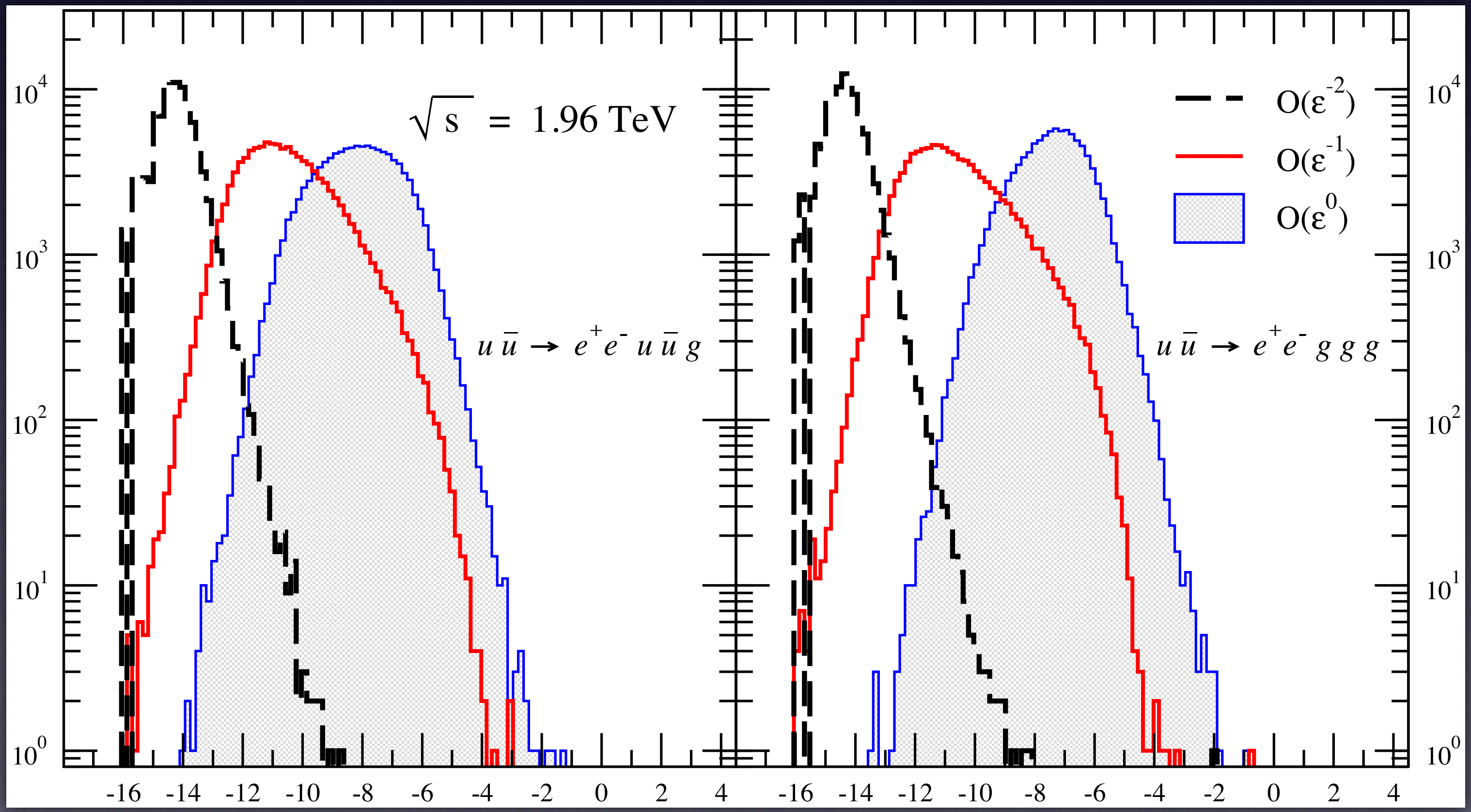
- Advantage to sampling more points is that you can compute as many coefficients as you have sampled points.
  - So extra sampling means we can test higher coefficients.
- Higher coefficients should vanish.
- How close they approach zero is a very good test of numerical accuracy, e.g. $t^4 = 10^{-8} \Rightarrow \sim 8$ digits.

Minimal Re-Computation

- Error handling strategy,
  - If a coefficient fails a test recompute it in higher precision.
  - Currently use the **`qd`** package for double-double and quad-double types.
- Minimizes the amount of time spent using higher precision, this can be a factor of 10 or more slower than double precision.
- Very rarely need to recompute entire amplitude e.g. only when cut part and rational part are very large and cancel or when the IR and UV singularities do not come out right.

Numerical Stability

- Check at a significant number of points the numerical accuracy.
- Testing over 10^5 actual phase space points for two Z+3 jets sub processes gives



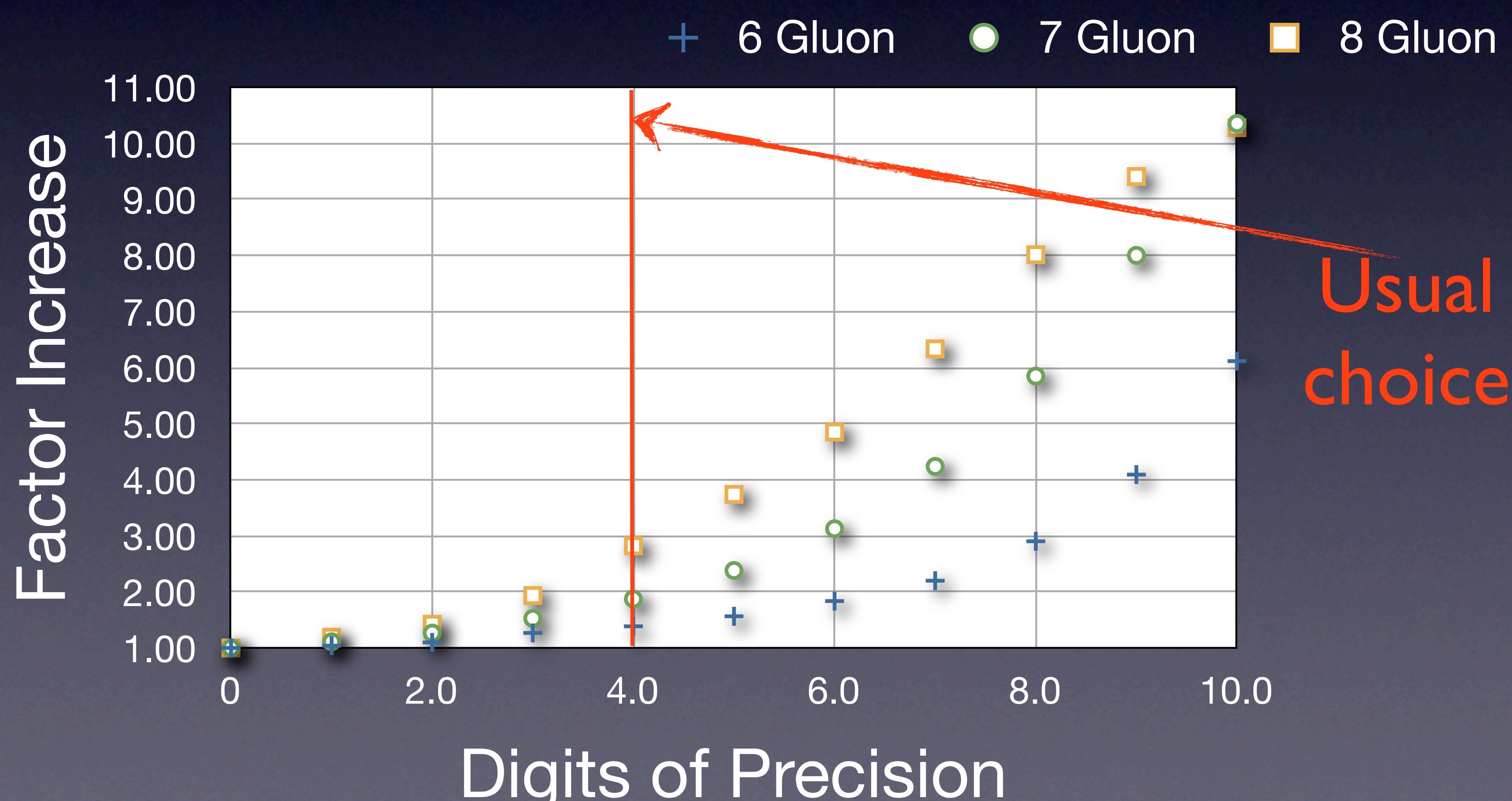
$$\log_{10}\left(\frac{\left|d\sigma_V^{\mathrm{BH}} - d\sigma_V^{\mathrm{target}}\right|}{\left|d\sigma_V^{\mathrm{target}}\right|}\right)$$

- Extremely good control over numerical stability.

Effect on Timing

- Investigate how much extra time is spent computing an amplitude as we demand more precision.

The more complex the process the more time spent at higher precision.



Timing for Gluons

- Average computation time of a single colour ordered amplitude for a specific helicity configuration.

	Timing (ms) no checks	Timing (ms) 4 digits accuracy
6 Gluons	1.6 (0.95)	2.2 (1.2)
7 Gluons	5.9	11
8 Gluons	15	50

Use on-shell recursion when it is faster.

- ~50 times faster than initial code 2 years ago.

All timing on a Core 2 Duo 2.93Ghz

Timing for W+3jets

- Average computation time of a single colour ordered amplitude for a specific helicity configuration.

	Timing (ms) no checks	Timing (ms) 4 digits accuracy
qqgggll (LC)	6.6	7.6
qqgggll (SLC)	36	50
qqQQgll	18	21

- Additional time spent depends upon the process.

Actual Runtime

- For each point we do not compute every sub process.
- Split up the amplitude into Leading Colour and Sub-Leading Colour pieces.
- Sub leading colour is much slower ~7 times, but contributes ~10 times less.
- For a particular statistical error need only call it ~1/100 of the time.
  - In practice to be conservative we call it ~1/10 of the time.
- This only doubles the total running time.

Where is the Computation Time Spent?

- Comparing the speed of the equivalent tree to the one-loop amplitude we see $\sim 10^3$ speed difference.
- Number of trees computed in a one-loop amplitude computation $\sim 10^3$.
- Profiler output tells us that ~80-90% of the time is spent computing trees.
- Main area to focus on is speeding up the trees.

Using Analytic Formulae

- BlackHat uses on-shell recursion to compute trees, our experience has shown that for up to 8-9 legs this gives the best results. (see also [Dinsdale,Ternick,Weinzierl])
- Increase speed by using analytic formulas wherever possible.
- BlackHat automatically uses any analytic formula added to its libraries for,
  - Trees.
  - Rational and cut parts.
- Trivial to add new processes, general interface for doing this.

Conclusion

- BlackHat is now a mature one-loop-amplitude code.
- Numerical instabilities are well understood and controlled.
- The efficiency of the code has increased dramatically and we hope to continue improving in the future.