Progress at NLO
(How to calculate ***PP ➡ ≥8 jets*** @ NLO)

W. Giele, G. Stavenga and J. Winter

1) Progress last years:
   Virtual contributions: enormous...
   Radiative contributions: not so much...

2) Leading Order MC's:
   Using numerical GPU's for parton level MC's

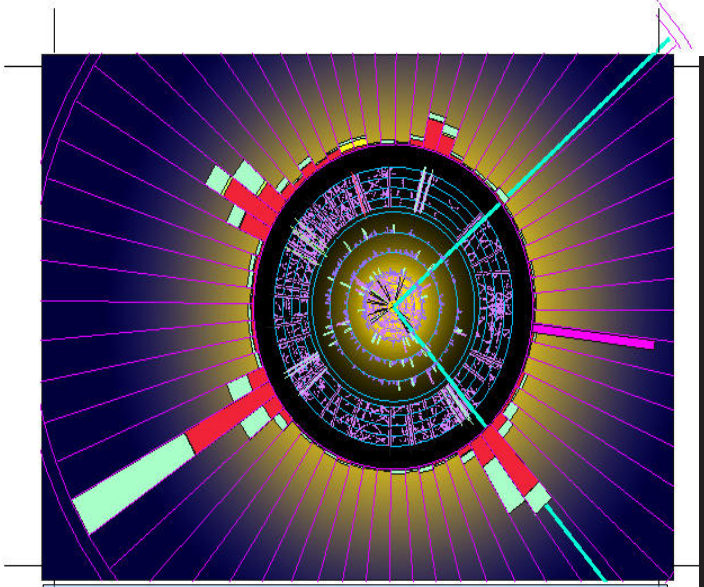
3) NLO MC's:
   NLO for maximal differential jet observables

Progress

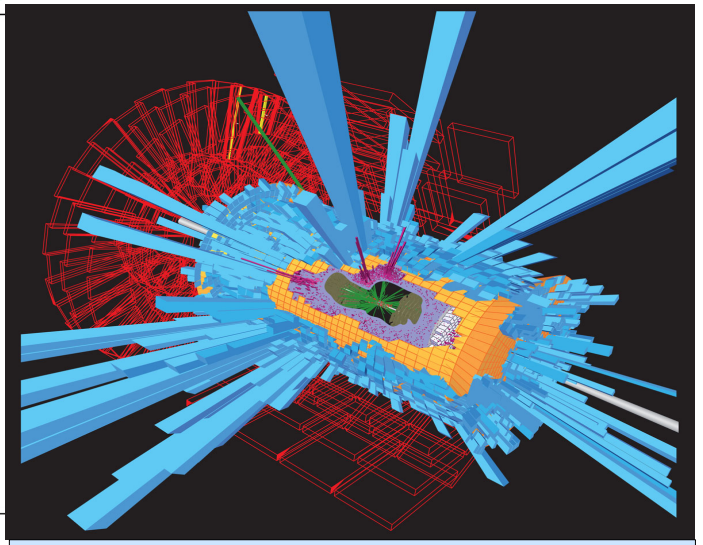
- A few years back virtual lagging radiative evaluations
  - State of the art virtual: ***PP ➡ 2,3 objects***
  - Hold back by virtual contribution evaluations
- Since then great progress in virtual event evaluation
  - State of the art virtual: ***PP ➡ 4,5 objects***
  - Hold back by radiative event evaluation
- Time to start to reverse the order yet again:
  - Calculating ***PP ➡ ≥6 objects***
  - Hold back by virtual contribution evaluations
- This talk deals with how this can be done....

Jets at Hadron colliders

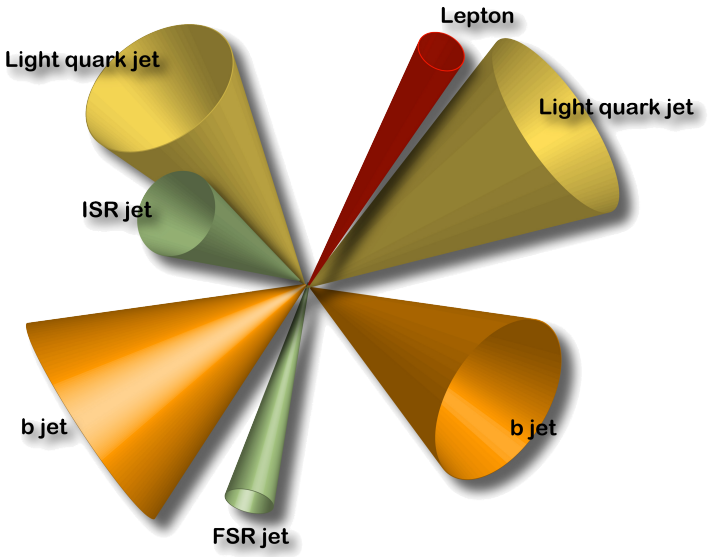
- Hadron colliders have a very “jetty” environment



Top candidate from D0



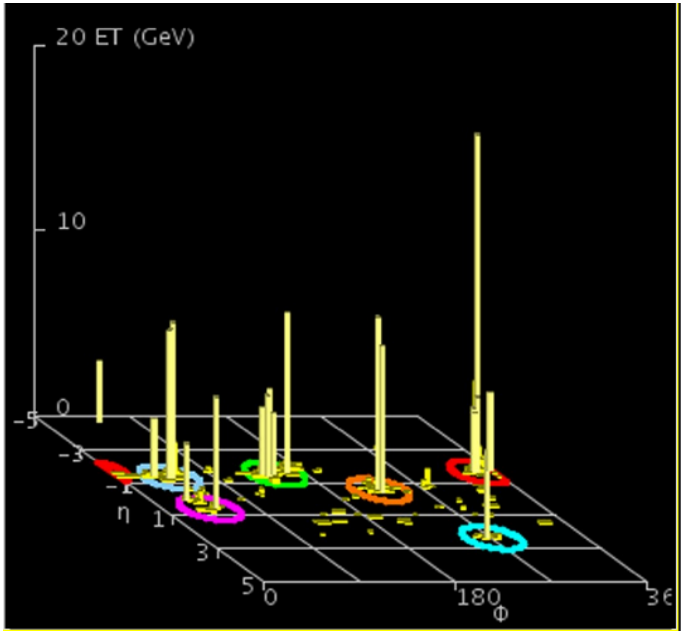
Top candidate from CMS

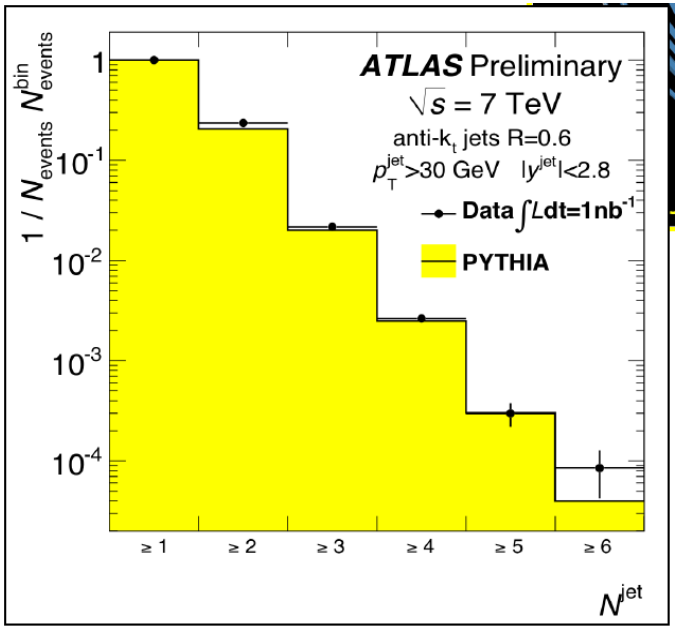


Top candidate from pQCD

- The LHC is even more jetty than we are used to from the TEVATRON.

Why bother?





- Experimental: They will be/are observed

  PP ➞ 2,...,6,...,10,...20,... jets

  so they need to be predicted.

- Theory: if we can do *2➞10 jets* @ NLO,

  We can do *2➞6 jets* @ NLO really well.

Approximations

- We want to develop methods and a proof of principle.
- To do this we restrict ourselves to the all gluonic contribution to $PP \to jets$ (i.e. $gg \to n\,g$).
- Furthermore, we restrict ourselves to leading color.
- The developed methods will not change by adding in quarks and sub-leading color effects.

Using numerical GPU's

- Exciting new hardware is available.....
  - A GPU (Graphical Processing Unit) has the potential to turns your desktop in a PC farm by using massive parallel processing.
  - Potential speed-ups over CPU can be as high as a factor of *200!*
  - GPU farming already in full swing. We have already access to *16* GPU farms (speed-ups of order *16x200=3200)*.

LO on GPU's

- A LO evaluation uses recursion relations:
  - A polynomial algorithm of degree *4*
  - Very memory efficient: *n(n+1)/2 4*-vectors
- These two properties make it very suitable for GPU implementation.
- This can be applied to LO as well as the radiative contributions to NLO.
- (See arXiv:1002.3446 [hep-ph] for LO implementation and vircol.fnal.gov/TESS.html for code.)

GPU Hardware

- The Nvidia Tesla GPU is a plug-in card for your desktop.
- The Tesla GPU is designed for numerical applications and programmable in C/limited C++/FortranXX/Java/Python/...
- The GPU has *30* multi-processors (MP's), each with *1024* processors (threads).
- For us the limiting factor is the fast on-chip shared memory of *16,384 32*-bit registers per MP (accessible by all threads on the MP).
- (There is *4* Gb off-chip slow access memory we use for I/O only.)

$1300, 200W @ peak

Turnkey hybrid GPU-CPU servers for GPU farms

LO on GPU's: Programming Principles

- All *30x1024=30,720* threads execute the same instructions (threads can skip ahead and wait for other threads to catch-up using if statements etc).
- A MC generator is trivially parallel-izable as the evaluation of each event executes the same instructions using different input (e.g. seeds for the random number generator or momenta).
- So, in principle, we can run *30,720* MC generators in parallel, each running *N* events (a speed up of *30,000*!).
- However, fast accessible memory is limiting the number of parallel events.
- Number of events per MP (the Tesla GPU has *30* MP's):

n	4	5	6	7	8	9	10	11	12
events/MP	102	68	48	36	28	22	18	15	13
threads/event	10	15	21	28	36	45	55	66	78

LO on single GPU's: Timing

AMD Phenom(tm) II X4 940 Processor (3 GHz)

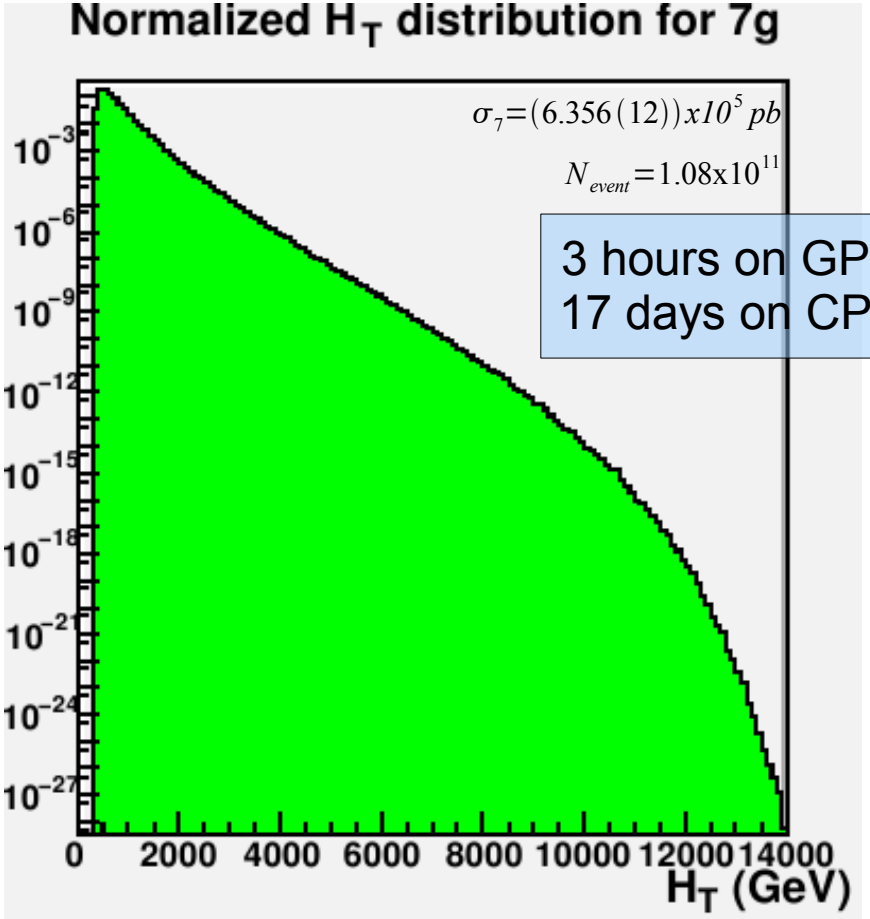
n	T_n^{GPU} (seconds)	$P_n(3)$	T_n^{CPU} (seconds)	$P_n(4)$	G_n
4	2.975×10^{-8}		8.753×10^{-6}		294
5	4.438×10^{-8}	0.91	1.247×10^{-5}	0.87	281
6	8.551×10^{-8}	1.03	1.966×10^{-5}	0.93	230
7	2.304×10^{-7}	1.19	3.047×10^{-5}	0.96	132
8	3.546×10^{-7}	1.01	4.736×10^{-5}	0.98	133
9	4.274×10^{-7}	0.94	7.263×10^{-5}	0.99	170
10	6.817×10^{-7}	1.05	1.044×10^{-4}	0.99	153
11	9.750×10^{-7}	1.02	1.529×10^{-4}	1.00	157
12	1.356×10^{-6}	1.02	2.129×10^{-4}	1.00	158

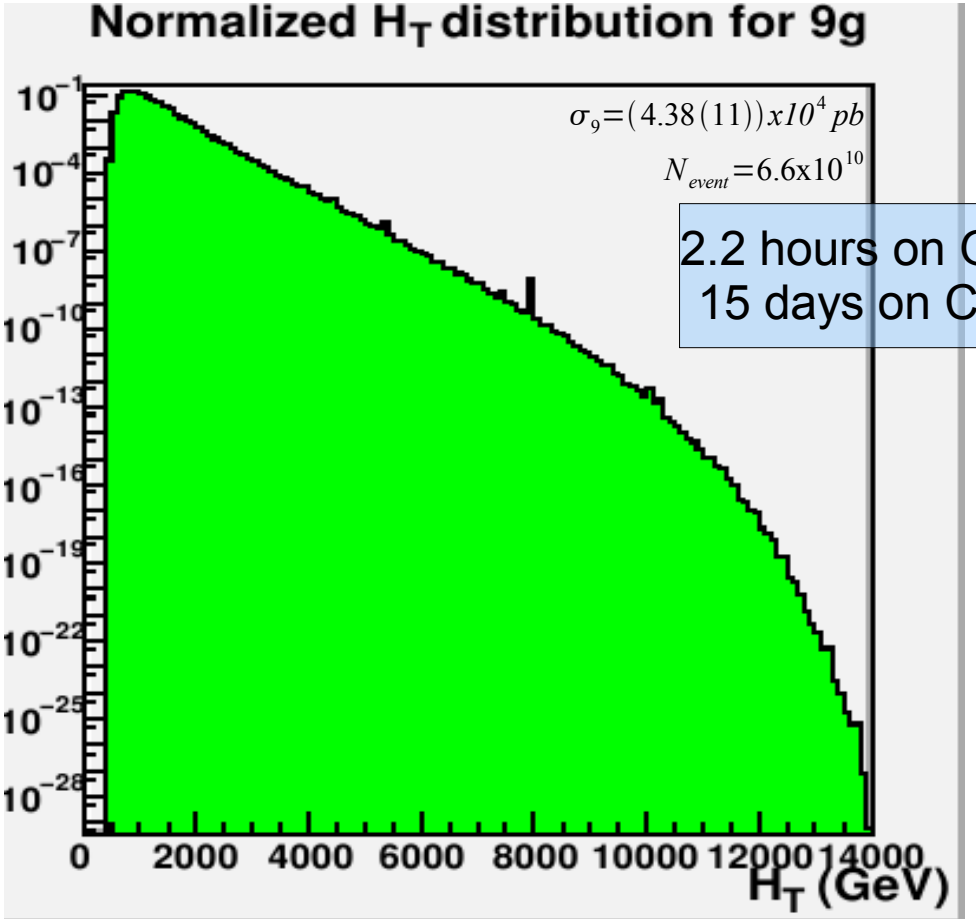
Table 2: The GPU and CPU evaluation times per event, T_n^{GPU} and T_n^{CPU}, given as a function of the number n of gluons for $gg \to (n-2)\,g$ processes. The polynomial scaling measures are also shown, for the GPU, $P_n(3)$, and for the CPU, $P_n(4)$. The $P_n(m)$ are defined as $P_n(m) = [(n-1)/n] \times \sqrt[m]{T_n/T_{n-1}}$. The rightmost column finally displays the gain $G = T_n^{\mathrm{CPU}}/T_n^{\mathrm{GPU}}$.

Kleiss, Stirling, Ellis (1986)

LO on single GPU's: Distributions

- Rambo generated events for *gg➞5 g* and *gg➞7 g* in large color limit at *14* TeV. (Phase space important sampling can give a significant increase in statistics).
- Cuts: Pt(jet)>*60* GeV; |eta(jet)|<*2*; R(jet,jet)>*0.4*





Comparisons

Gleisberg, Hoche (2008)

- Comparisons with Comix:

n	σ_n (pb)	$N_{\text{generated}}$	N_{accepted}	σ_n^{COMIX} (pb)
4	$(2.32421 \pm 0.00047) \times 10^8$	3.06×10^{11}	1.96848×10^{11}	$(2.3283 \pm 0.0023) \times 10^8$
5	$(1.4353 \pm 0.0011) \times 10^7$	2.04×10^{11}	1.12939×10^{11}	$(1.4355 \pm 0.0014) \times 10^7$
6	$(2.84780 \pm 0.00096) \times 10^6$	1.44×10^{11}	6.98918×10^{10}	$(2.8560 \pm 0.0030) \times 10^6$
7	$(6.356 \pm 0.012) \times 10^5$	1.08×10^{11}	4.49985×10^{10}	$(6.408 \pm 0.015) \times 10^5$
8	$(1.608 \pm 0.011) \times 10^5$	8.40×10^{10}	2.93316×10^{10}	
9	$(4.38 \pm 0.11) \times 10^4$	6.60×10^{10}	1.88182×10^{10}	
10	$(1.193 \pm 0.024) \times 10^4$	5.40×10^{10}	1.22356×10^{10}	
11	$(3.550 \pm 0.020) \times 10^4$	4.50×10^{10}	7.88017×10^9	
12	$(9.64361 \pm 0.74) \times 10^3$	3.90×10^{10}	5.13041×10^9	

Table 3: The cross sections σ_n for $gg \to (n-2)\,g$ and their mean standard deviations in pb as calculated by the TESS MC using 10^9 sweeps. The two center columns show the total number of generated events and the number of events passing the jet cuts. For comparison, the cross sections σ_n^{COMIX} in pb as computed by COMIX considering the full-color dependence are also given.

Current work for LO

- 16 GPU farm streamlining (*>10^10* LO event/sec)
- Generic recursion algorithm on GPU (one identical algorithm for any *n*-particle process).
  - For example each thread calculates its own (random) partonic configuration.
  - Only input are the vertices... Can deal with any model expressible in Feynman rules....
- Building in color expansions (*1/Nc^2* for now).
- This all forms the starting point for NLO (radiative events in NLO).

Virtual corrections

- The one-loop calculational techniques have been undergoing intense developments in the past few years.
- Generalized unitarity and parametric integration techniques can be combined in an efficient polynomial algorithm of *D*-dimensional unitarity

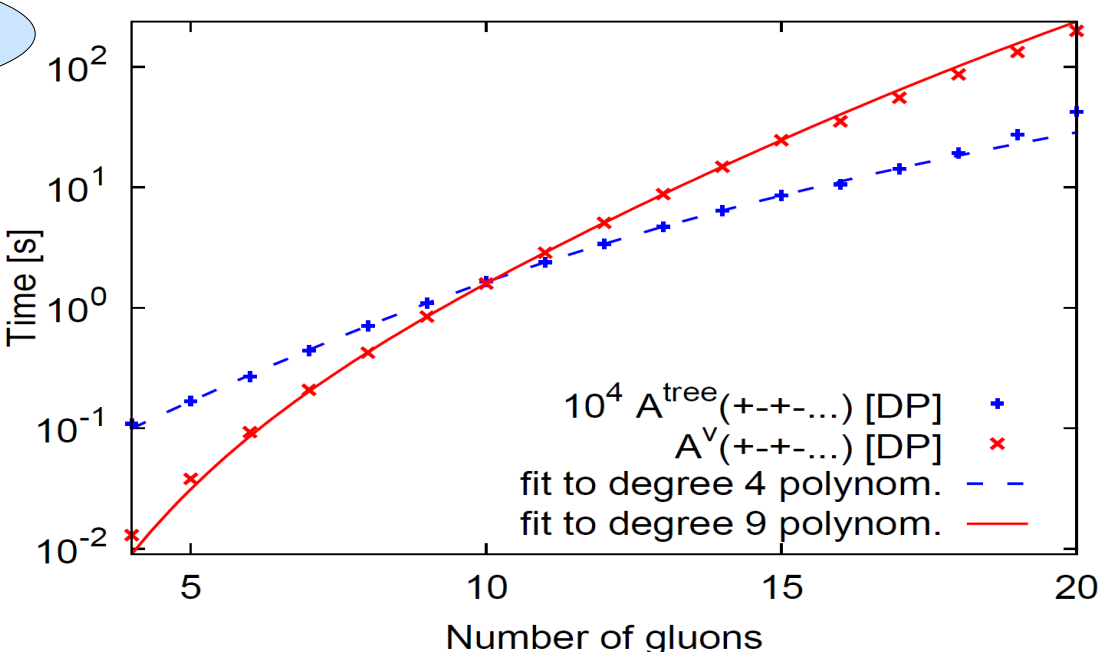
Bern, Dixon, Kosower (1994)
Britto, Cachazo, Feng (2004)

Ossola, Papadopoulos, Pittau (2006)

Ellis, Giele, Kunszt (2007)
Giele, Kunszt, Melnikov (2008

Alternative algorithms have been developed by
van Hameren, Papadopoulos, Pittau (2009) and
Berger,Bern,Dixon,Cordona,Forde,Gleisberg, Ita,Kosower,Maitre (2008)

We use the one loop *n*-gluon code of
Zanderighi, Giele (2008) and
Winter, Giele (2009).



Problems with virtual evaluation

- We can evaluate the radiative contributions to NLO very fast (*10^6-10^8* events/second) on single GPU.
- The virtual corrections are evaluated very slow in comparison (*0.1-100* events/second)
- For example to calculate *PP→8 jets* we have *0.7* virtual and *3x10^7* real events/second.
- While the real events are evaluated fast enough for the large size of phase space, virtual takes way too long for a realistic parton MC.

NLO redefined

- By redefining jets we ***can*** in fact calculate high multiplicity jet cross sections at NLO
- Suppose we can calculate the maximal differential *n*-jet cross section order by order:

$$\frac{d^{(n)}\sigma}{dJ_1 \cdots dJ_n} = \frac{d^{(n)}\sigma_{\mathrm{LO}}}{dJ_1 \cdots dJ_n}\left(1 + \alpha_S K_1 + \alpha_S^2 K_2 + \cdots\right)$$

- Then calculate NLO correction for ***unweighted*** LO
  - Gives small multiplicative correction to LO event.
  - Always positive weight in physical regions of order *1*.
  - Around *60,000 PP→8 jets* event/day on a desktop (on the 16 GPU farm we get ~*10^6* events/day)

Jet Algorithms

- The current jet algorithms use *2→1* clustering.
- This leads to cluster masses at NLO, while LO is massless.
- The result is an ill defined maximal differential jet cross section:

$$\frac{d^{(n)}\sigma}{d\,J_1 \cdots d\,J_n} = \sum_i \left(A_i \delta(m_i) + B_i \, \log^2(m_i) + \cdots \right)$$

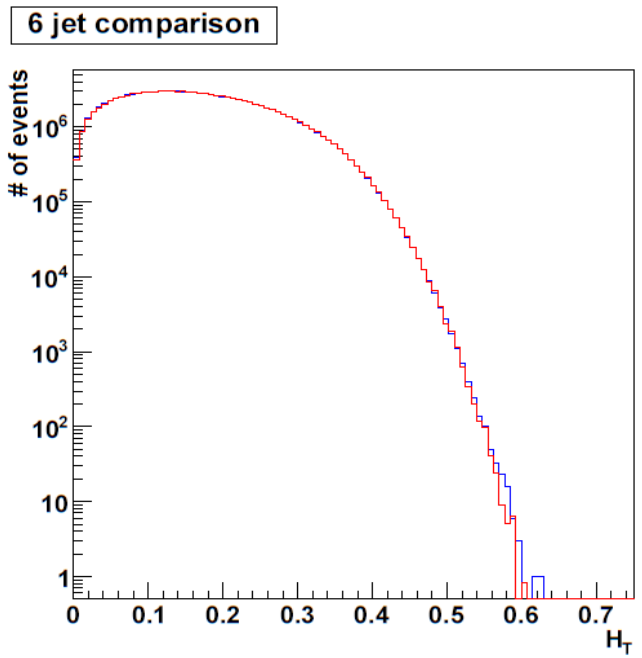
- One way out is to augment the jet algorithm such that the jets remain massless at NLO.
- Another possibility is integrating out mass...

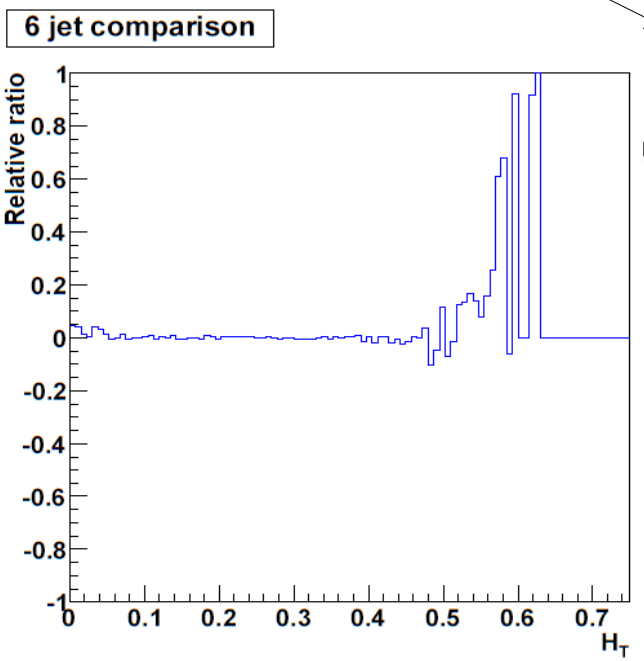
Redefining jets

- For now we use a jet algorithm designed to minimize the theoretical complications.
- Once that all works we will iterate back as close as possible to the existing jet algorithms.
- At NLO we need to match onto LO jets:
  - Clusters remain massless (***3➡2*** clustering)
  - Jets remain Pt-balanced (initial state is only rescaled) or need to introduce beam jets...
- From showers we learn how to write the exact inverse of the jet algorithm: i.e. a ***2➡3*** brancher

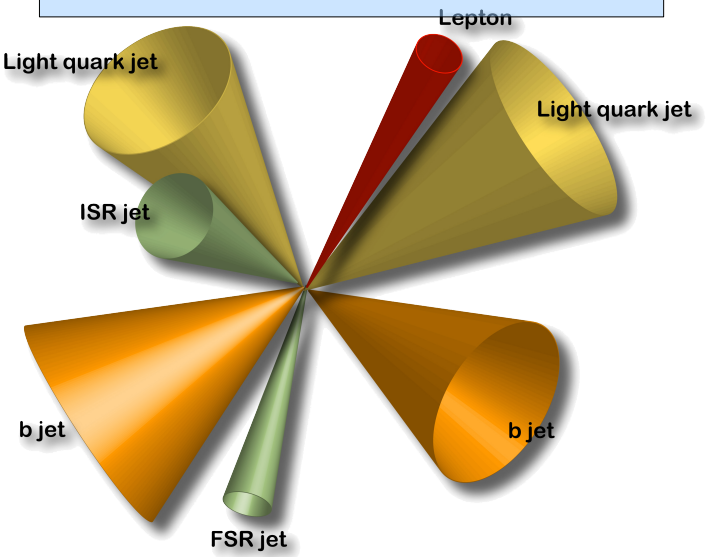
Reversible branchings

- We perform a branching from the *n*-parton to the *(n+1)*-parton phase space which is ***always*** clustered back by the jet algorithm to the initial *n*-parton configuration.
- That is, we can generate the radiative phase space for a maximal differential cross section





Integrates out all parton configurations in cones



The NLO generator

- Given a LO event configuration:
  1. Calculate born and virtual (*1* event) on CPU
  2. Calculate *10^9-10^10* radiative events (single GPU). All reconstruct back to the same LO final state jet configuration (on the GPU in parallel with CPU).
  3. Add results of 1. and 2. together and return value
- Notes:
  - Memory transfer between GPU and CPU minimized, giving even faster evaluations of radiative events.
  - The radiative phase space is just a ***3 parameter*** antenna PS independent of number of jets.

Where are we now/going

- Currently all pieces work, the ensemble not yet.
- We are added the quark virtual corrections (quarks can be chosen massive). This is done in collaboration with K. Melnikov and M. Schulze.
- We are adding in *1/Nc^2* expansion.

"Step one: take a good stiff drink."

Summary NLO

- We have developed a method to efficiently calculate maximal differential multi-jet cross sections.
- Experimental benefits augmented jets:
  - Positive weights at NLO
  - NLO is simply a re-weighting of LO.
  - ME-method defined at NLO
  - Guaranteed IR safe jet observables
- Theoretical benefits augmented jets:
  - Radiative PS reduced to a 3 variable integration.
  - IR/UV cancellations happen at most differential level
  - Well defined and stable multi-jet cross sections

Outlook

- By employing GPU's, desktop evaluations of *PP ➞ ≥8 objects (object={jet,V,H,...})* is very realistic.
- New generations of GPU's have exponential growth in capabilities (4x more on-chip memory, fast double precision, etc).

The new Fermi GPU

- We are close to the first release of the TESS-MC (Threaded Event SimulationS) which will at minimum give NLO estimates of *PP ➞ ≤10 jets*