Computing issues in PHASE SPACE INTEGRALS

Rikkert Frederix
University of Zurich

HO10, CERN, June 21 - July 9, 2010

What we calculate

$$\sigma^{\mathrm{NLO}} = \int_{m+1} d\sigma^R + \int_m d\sigma^V + \int_m d\sigma^B$$

‘Real emission’
NLO corrections

‘Virtual’ or ‘one-loop’
NLO corrections

‘Born’ or ‘LO’
contribution

What we calculate

$$\sigma^{\mathrm{NLO}} = \int_{m+1} d\sigma^R + \int_m d\sigma^V + \int_m d\sigma^B$$

- Phase space integration with possible restrictions in form of cuts: very complicated (nested) integration bounds

- Up to O(10-20) dimensional integrals

- Using Monte-Carlo techniques with VEGAS (or similar in-house routines) for adaptive importance sampling

- Our code (called 'MadFKS') is written in Fortan77. Okay for pure 'number crunching'

The integrand

- Because the integrand has many more peaks than integration variables, a single phase space mapping cannot align integration variables along peaks

- Need for multi-channel integration

Multi channels

- For example

$$\int d\sigma^{\mathrm{R}}(x) = \sum_i \int d\sigma_i^{\mathrm{R}}(x)$$

$$\int d\sigma_i^{\mathrm{R}}(x) = \int \frac{d\sigma^{R}(x)}{\sum_j f_j(x)} f_i(x)$$

- Sum over $f_j(x)$ damps peaks in $d\sigma^R(x)$
- $f_i(x)$ has many less peaks, and therefore we can flatten them using a simple change of integrations variables
- O(50) channels needed

Structure of the code

$$\sigma^{\mathrm{NLO}} = \int_{m+1} d\sigma^R + \int_m d\sigma^V + \int_m d\sigma^B$$

- In fact, each of the contributions contains O(50) subprocesses: we have a single executable for each of them that is put in a separate directory

- Size of the executable (with libraries linked statically) is O(10-20 MB for $d\sigma^R$) and O(0.1-1 GB for $d\sigma^V$)

- Within those, multi-channel integration is used: O(50) channels per directory

- In total, O(1-5k) integration channels for complicated processes

Trivially parallelized

- Each of these integration channels can be executed independently

- That means that we have O(1-5k) jobs

- We cannot lose any job. All results need to be collected and summed to get the correct final answer

Running

1. Jobs are executed with relatively low statistics. Takes 5 min to 1 hour per job

2. Results are collected

3. Jobs with the largest integration uncertainties are resubmitted with higher statistics, starting with the VEGAS grids generated before. A couple of 100 jobs, and can take up to a couple of days per job to finish

4. Results are collected

5. If not satisfied, resubmit again with even higher statistics

Running

- So far, running of the code has been done on a cluster (in Louvain-la-Neuve, Belgium) of O(25) machines with 8 cores each: maximally 200 jobs can be executed in parallel

- Code is compiled on a central node with all libraries linked statically

- All jobs are submitted to the Condor batch system that takes care of queueing, scheduling etc.

- No shared file system: executable is copied to the node when the run starts

Writing out ntuples

- Usually, the ntuples are not written to disk
  - Results are simply accumulated while running and plots for predictions for physical observables are generated on the fly
- If the data is written, it can easily lead to terabytes of data
  - The advantage is clear: generation of plots/cuts can be redone, without redoing the calculation

Summary

- Also in theoretical high energy physics computing demands are increasing

- Clusters with more than 100 machines are needed to generate reliable predictions for the complicated processes at the LHC in a reasonable amount of time (i.e. days)

- Disk space is not a problem when not saving the events

- Grid running?