Optimizing MC generator usage

Discussion on strategies to avoid MC-limited on HL-LHC results

Discussion topics

1) Reduce CPU requirements by running large scale event generation at LO:

- Generate large detector simulated mainly with LO multijet merging
  ⇒ large statistics LO samples are cheap to generate
- If phase space distributions between LO and (N)NLO differs only by ~constant factor (not true for all processes, but for some e.g. V+jets)
  ⇒ detector simulation unaffected, fixed-order NNLO norm already std
- Produce (weighted) (N)NLO samples only at particle level (or at all?)
- For searches & measurements on unfolded data, detector corrections inferred from high-stat LO
  ⇒ obvious benefit

Discussion topics

1) Reduce CPU requirements by running large scale event generation at LO:

- For searches with detector level predictions, need reweighting of LO sample to (N)NLO sample. Also reweight to e.g. beyond-leading colour.
  ⇒ some gain from reweighting on observable level (observables are pos. definite, event weights are not), but overall the situation will be similar as straight forward detector simulation of (N)NLO samples
- EW corrections: log cancellations missing, but still reweightable.
  (approx. EW corrections can easily incorporated in merged LO sample)

Discussion topics

2) Reducing computing costs by sharing unweighted matrix-element event samples between experiments:

- Sociology and logistics of shared space between experiments
- Format details: gridpacks? LHEs? Sherpa intermediates? HDF5? Hybrid?
- Need to converge on few, but crucial, parameters (PDFs, α_S, scales)
    - Get around this by partial unweighting and storage of coefficients to recompute weight with any input parameters?
      → narrow weight distribution since all PDF sets similar for common scale
- Showering, hadronisation and detector simulation then in each experiment
    - Ideally factor 2 gain in computing… but merging/unweighting also slow?

Discussion topics

3) Other topics

- Suggestion for experiment technical RAs to make “deep” contributions in MC teams, for issues where experiments incentivised and theorists not.
  - How to work in practice? Low-hanging-fruit tasks? E.g. caching/vectorisation in LHAPDF, weights and speed in HepMC + LHE, ??? in POWHEG/Sherpa/…?

- Other ways that generator codes can be better aligned to usage/needs of experiment MC production campaigns?
  - e.g. more powerful final-state biasing, re-hadronisation hooks, advanced cross-section / phase-space sampler biasing for multileg/multistage systems. Avoid wastage? Common API (prototype constructed by summer student)?