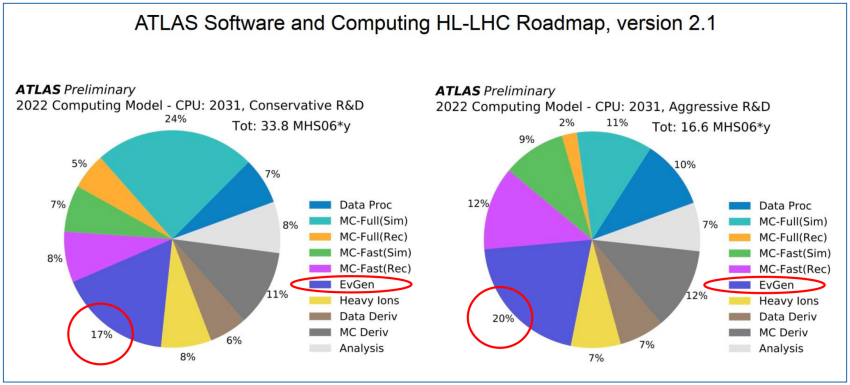
Towards NLO vectorisation in MG5aMC

Hardware acceleration for next-to-leading order event generation within MadGraph5_aMC@NLO

Zenny Wettersten (CERN, TU Wien)
Olivier Mattelaer (UCLouvain)
Stefan Roiser (CERN)
Andrea Valassi (CERN)
Marco Zaro (UniMi, INFN)

Why NLO?

Reminder: Why event generation?



(CERN-LHCC-2022-005, taken from AV's slides)

- Major compute cost; $\sim$ 50% events generated at NLO (or higher)

Comparison between LO and NLO in MG5aMC

- NLO event generation increases theoretical precision by roughly an order of magnitude — important entering the HL-LHC era
- NLO event generation computationally heavier than LO

$p\,p \to t\,\bar{t}$	$+0j$	$+1j$	$+2j$
at LO	9.413	14.853	292.813
at NLO	1.260	75.195	$\sim$ 120200

Tests of total compute time ran on personal machines — only meant for rough comparisons.

Comparison between LO and NLO in MG5aMC

- NLO event generation increases theoretical precision by roughly an order of magnitude — important entering the HL-LHC era
- NLO event generation computationally heavier than LO

$p\,p \to t\,\bar{t}$	$+0j$	$+1j$	$+2j$
at LO	9.413 s	14.853 s	292.813 s
at NLO	1.260 m 75.569 s	75.195 m 4511.698 s	$\sim$ 120200 m $\sim$ 721300 s
t_{NLO}/t_{LO}	8.028$\times$	303.8$\times$	$\sim$ 24600$\times$

Tests of total compute time ran on personal machines — only meant for rough comparisons.

What is NLO?

Perturbative scattering amplitudes

- Can't evaluate QFT probabilities directly – Taylor expansion:

$$\underbrace{\frac{d\sigma}{d\Omega}}_{\text{diff. cross section}} = \underbrace{\left| \frac{\langle 0|\, T\left\{\Pi_k(\zeta_{I,k}) \exp\left[-i\int_{-\infty}^{\infty} H_I(t)\, dt\right]\right\}|0\rangle}{\langle 0|\, T\left\{\exp\left[-i\int_{-\infty}^{\infty} dt H_I(t)\right]\right\}|0\rangle} \right|^2}_{\text{scattering amplitude}}$$

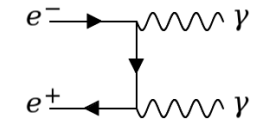
$$\simeq \underbrace{C}_{\text{norm.}} \left| T\left\{\Pi(\zeta_I)\left[-i\underbrace{\int H_I(t)dt}_{\textbf{L}\text{eading }\textbf{O}\text{rder}} + \underbrace{\iint H_I(t)H_I(t')dt'dt}_{\textbf{N}\text{ext-to-}\textbf{LO}} + \ldots\right]\right\}\right|^2$$

- Complexity scales *very quickly*

Feynman diagrams 101: LO contribution

- Taylor expansion in the *coupling constant*
  - Go from *minimal* description of this process to more complicated
- Example: electron annihilation, producing two photons

$$\text{1st term} \propto T\left\{\Pi(\zeta_I)\left[-i\int H_I(t)dt\right]\right\} \equiv$$

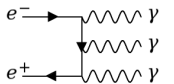


LO Feynman diagram

- *Tree-level* diagrams, momenta of all particles well-defined:
  $\Longrightarrow$ Treat particles as vectors; this is an inner product

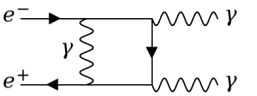
Feynman diagrams 101: NLO contribution

- Second term (NLO): Add one more *coupling* ("interaction")
- Either add additional (unobserved) external particle:

$$\underbrace{T\left\{\Pi(\zeta_l)\left[\int H_I(t)H_I(t')dt'dt\right]\right\}}_{\sim\text{2nd term}} \in \underbrace{\text{  }}_{\text{Real emission}} \Longleftarrow$$

Still tree-level
"Just" algebra!

- Or add completely internal particle:

2nd term $\in$ 

Loop correction

$\Longleftarrow$ Undefined internal momenta p!
Must integrate over internal p

Local subtraction schemes

- Both NLO contributions are *divergent* in phase space
- However, *the divergences locally cancel*

$$\sigma_{NLO} - \sigma_{LO} = \underbrace{\int L \, d\Omega_n}_{\text{loops, have poles}} + \underbrace{\int R \, d\Omega_{n+1}}_{\text{reals, divergent in } \Omega} = \text{finite}$$

- To calculate, two algebraic manipulations:
  - Factorise the $(n+1)$-dimensional phase space $d\Omega_{n+1} = d\Omega_n d\Omega_1$
  - Add 0 in the form of **c**ounter-**t**erms (**CTs**) $C_L = \int C_R \, d\Omega_1$

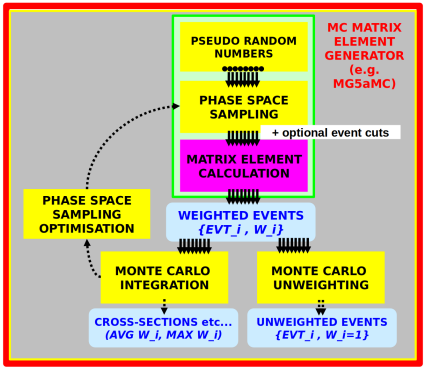
$$\sigma_{NLO} - \sigma_{LO} = \int (L + C_L) \; d\Omega_n + \int (R - C_R) \; d\Omega_{n+1}$$

FKS subtraction (arXiv:hep-ph/9512328, hep-ph/9706545)

- Evaluating NLO $\simeq$ choosing appropriate CTs
- One standard choice: **FKS subtraction scheme**
  - Combines particles with phase space divergences
  - Splits phase space into *FKS sectors* based on these particles
  - For each sector, define local CT from particles' kinematics
  - Total σ given by sum over FKS sectors
- MC: Generate a "proper event" and a "counterevent" and sum
- Note: These subtraction schemes are **not** the (main) contributor to negatively weighted events at NLO
  (these primarily come from matching procedures)

NLO event generation in MG5aMC

Event generation structure



(taken from AV's slides)

- As mentioned, MG5aMC: Determines σ and distribution; generates unweighted events
- At NLO, generate event and counterevent to sum
- Born, real, and loop diagrams
- Born and real diagrams are tree-level diagrams (same as LO)
- Loop diagrams integrate over internal undefined momenta

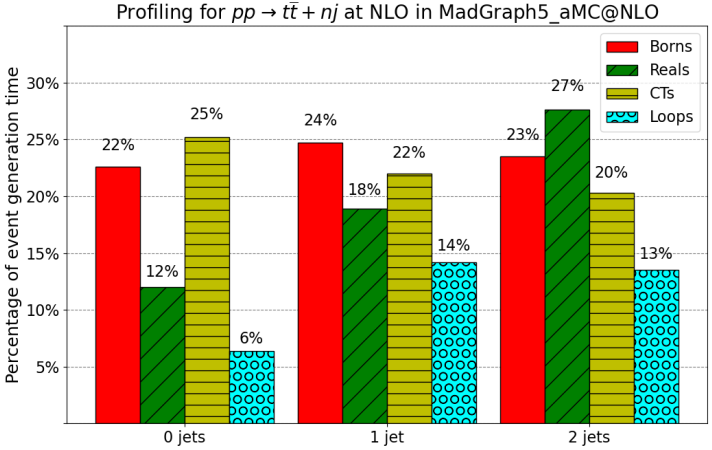
Single event pseudocode

```
1:  FKS ← setFKSsector( )
2:  p ← generateMomentum(FKS)
3:  if passCuts_nBody(p) then                          ▷ Branch
4:      M(Born) ← BornMatrix(p)                        ▷ Born (LO!) amplitude
5:      M(Loop) ← LoopMatrix(p)                        ▷ Loop correction
6:  if passCuts_Real(p) then                           ▷ Branch
7:      M(Real) ← RealMatrix(p)                        ▷ Real emission
8:  if passCuts_Soft(p) then                           ▷ Branch
9:      M(Soft) ← SoftMatrix(p)                        ▷ Soft CT
10: if passCuts_Coll(p) then                           ▷ Branch
11:     M(Coll) ← CollMatrix(p)                        ▷ Collinear CT
```

Differences between LO and NLO

- For NLO calculations, phase space generation and amplitude evaluation cannot be cleanly separated
  - With FKS subtraction, amplitude routines not identical in phase space
  - Additionally, different terms may have different cuts
- $\Longrightarrow$ LO and NLO have distinct code bases
  - NLO uses a modified version of MINT (arXiv:0709.2085)
- Loops: Finds integrand formula, passes to external loop libraries
  - Due to instabilities, try loop libraries until one gives stable result
  - Generally, *might require quadruple precision* for very unstable points
    - No major GPU vendors support quadruple precision

Runtime profile for NLO event generation



Runtime spent in each type of amplitude evaluation.

- Borns, reals, and **C**ounter **T**erms are all tree-level scattering amplitudes
  - Tree-level has been vectorised for the LO GPU port!
- CTs collectively refer to soft and collinear terms
- For tested processes, **loops subdominant**
- Not necessarily general — more profiling needed

Vectorising NLO event generation

Working plan

1. First step: Modify algorithm to multi-event interface
   - Keep in mind that end goal is vector instructions
2. Implement this for *fixed order calculations* (σ integration)
   - Codebases for fixed order and event generation similar but not identical
3. Simulate vectorisation with multithreading $\Longleftarrow$ We are here!
   - Allows for testing the algorithm without SIMD instructions
4. Replace amplitude evaluation code with vectorised versions
   - Tree-level amplitudes already vectorised in LO port
5. Port from fixed order to event generation

- Stages 4 and 5 can be exchanged
  - Multithreading could be beneficial at NLO even without full vectorisation

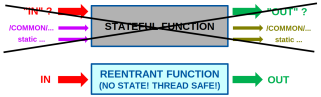
Modified single event pseudocode

1: $FKS \leftarrow \textsc{setFKSsector}()$
2: **while** $!\textsc{passAnyCuts}(p)$ **do**
3: $\quad p \leftarrow \textsc{genMomentum}(FKS)$
4: $\mathrm{M}(Born) \leftarrow \textsc{BornMatrix}(p)$
5: $\mathrm{M}(Real) \leftarrow \textsc{RealMatrix}(p)$
6: $\mathrm{M}(Soft) \leftarrow \textsc{SoftMatrix}(p)$
7: $\mathrm{M}(Coll) \leftarrow \textsc{CollMatrix}(p)$
8: $\mathrm{M}(Loop) \leftarrow \textsc{LoopMatrix}(p)$
9: **for** $a \in [B, R, S, C]$ **do**
10: $\quad$ **if** $!\textsc{passCuts}(a, p)$ **then**
11: $\quad\quad \mathrm{M}(a) \leftarrow 0$

- Calls all amplitudes for each phase space point
- Computationally less efficient, but mostly "branch-free"
- Lines $4 - 7$ $\sim$use$\sim$ routines vectorised for LO port
  - Some internal differences, but fundamentally same
- As LO port shows, parallel tree-level calls become cheap
  - On e.g. GPUs, likely cheaper to "waste"

Status report

- Earlier this year, work into NLO vectorisation started
- Currently working on multithreading fixed order calculations
  - Some bug(s) remaining in phase space generation
  - Ignoring loops for now, but they can also be multithreaded
    although they would be problematic to vectorise
- Once working: Step back to refactor codebase
  - As AV said, stateful functions taxing



  - Locality makes amplitudes modular; can e.g. plug-and-play loop libraries
- When refactored, two steps remain:
  - Port refactoring from fixed order to event generation
  - Add proper vectorised routines for (tree-level) amplitudes

Future prospects

Plausible acceleration in amplitudes

- Borns, reals, and CTs are all tree-level amplitudes
  $\Longrightarrow$ Can recycle work from LO port!
  - Profiling shows majority of runtime spent in tree-level amplitudes — can modify NLO interface to call these functions instead of standard routines
  - Could reduce total runtime by $\sim 50 - 70\%$
- Loop amplitudes generally need quadruple precision, unsuited for vectorisation and (especially) GPUs
  - MG5aMC can approximate loops by the Born amplitude
    $\Longrightarrow$ Only call full loops for some phase space points!
  - Collaborators at ANL looking into porting existing loop libraries to GPUs
    $\Longrightarrow$ Could remove another $\sim 5 - 15\%$
- Possible acceleration less than for LO; but still sizable!

Outlook

- Vectorising NLO next step in accelerating simulation
- In MG5aMC, heaviest NLO parts can be recycled from LO port
- Our team is currently putting extensive effort into this work!
- Others work in parallel — collaborative and synergistic projects
- Hopefully, by next CHEP we have something to show :-)

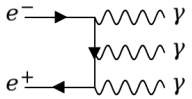
Many thanks to Stefano Frixione and Rikkert Frederix for their assistance in both the theoretical and numerical aspects of this work.


CERN

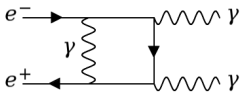
Backup slides

NLO contributions

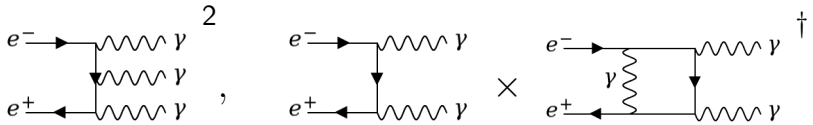
- Taylor expansion in *coupling order*
  - Real emission adds single vertex,



  - Loop adds *two* vertices,



- *Absolute square*: Include *square* terms with *two* extra vertices



Negative weights in NLO generation

- For NLO calculations, three likely negative weight sources:
  - Local subtraction scheme; parton shower variables; matching procedure
- Matching procedure can greatly reduce negative fraction
  - Matching: Subtracting the *approximated* NLO σ from the parton shower

$$\underbrace{d\sigma^{(\mathbb{H})}}_{\text{matched emission } \sigma} = \underbrace{d\sigma^{(\text{NLO,real})} - d\sigma^{PS}}_{\text{analytic } \sigma - \text{ shower } \sigma},$$

$$\underbrace{d\sigma^{(\mathbb{S})}}_{\text{matched Born-like } \sigma} = \underbrace{d\sigma^{PS} + \sum d\sigma^{\text{NLO,CTs}}}_{\text{shower } \sigma + \text{ CT } \sigma}$$

- See e.g. arXiv:2002.12716, arXiv:2310.04160 for more details

Approximating loop contributions

- Instead of evaluating loop for each phase space point, use an approximator A,

$$\int V \, d\Omega = \int (V - A) \, d\Omega + \int A \, d\Omega$$

- MC over RHS — if $(V - A) \approx 0$, always use A instead of V
  - Subtraction — possible source of negative weights if $V < A$

Stability in loop integrals

- MG5aMC tests loop stability by Lorentz transformations
  - Lorentz invariance: New reference frame should yield same loop integral
  - Loop integral V at phase space point p invariant under e.g. rotation R

$$\text{Stability check:} \qquad V(p) \equiv V(R(p))$$

- If check fails: Jump to next loop integrator
- If all fail at double precision, move to quad

Multithreaded acceleration for NLO

- HEP processes have multiple parton configurations

$$p\,p \to t\,\bar{t} \in \begin{cases} g\,g \to t\,\bar{t} \\ q\,\bar{q} \to t\,\bar{t} \end{cases}$$

- In MG5aMC, each parton config. run as separate process
- $\Longrightarrow$ If one subprocess slow, single job runs for long time
  - Not an issue at LO; speed difference noticeable but not major bottleneck
- Multithreading subprocesses:
  $\Longrightarrow$ Splitting bottlenecks across all cores