Generating Feynman Diagrams for QED in Julia

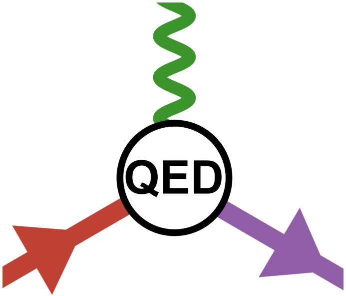
Anton Reinhard[1,2], Simeon Ehrig[1,2], Uwe Hernandez Acosta[1,2]

[1] *Helmholtz-Zentrum Dresden-Rossendorf*, [2] *Center for Advanced Systems Understanding*

30.09.2024

Goal

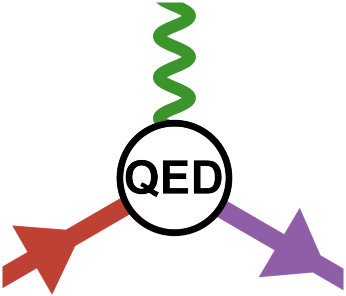
- Generate Feynman diagrams for tree-level perturbative Quantum Electrodynamics (QED) for arbitrary scattering processes



DRESDEN concept HZDR

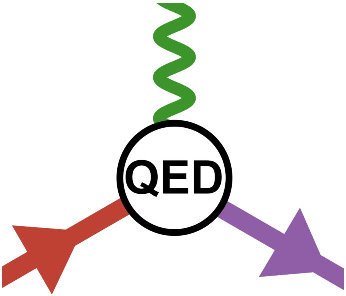
Goal

- Generate Feynman diagrams for tree-level perturbative Quantum Electrodynamics (QED) for arbitrary scattering processes
- Generate computable functions calculating the matrix elements for given particle momenta



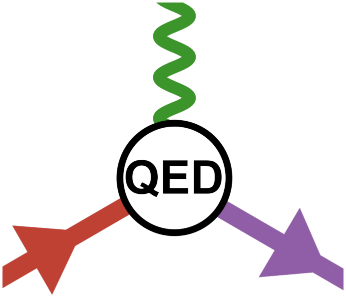
Goal

- Generate Feynman diagrams for tree-level perturbative Quantum Electrodynamics (QED) for arbitrary scattering processes
- Generate computable functions calculating the matrix elements for given particle momenta
- Reuse as much as possible



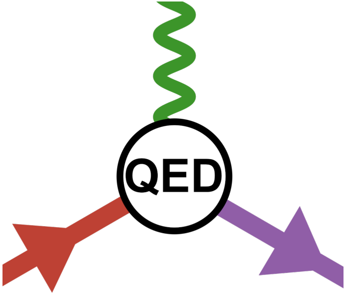
Goal

- Generate Feynman diagrams for tree-level perturbative Quantum Electrodynamics (QED) for arbitrary scattering processes
- Generate computable functions calculating the matrix elements for given particle momenta
- Reuse as much as possible
- Do it in Julia

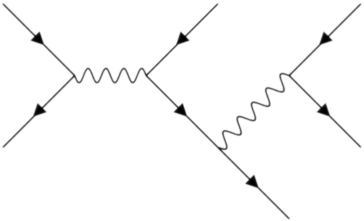


Goal

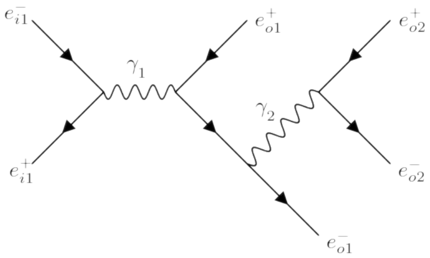
- Generate Feynman diagrams for tree-level perturbative Quantum Electrodynamics (QED) for arbitrary scattering processes
- Generate computable functions calculating the matrix elements for given particle momenta
- Reuse as much as possible
- Do it in Julia
- Benefit from easy CPU and GPU parallelization



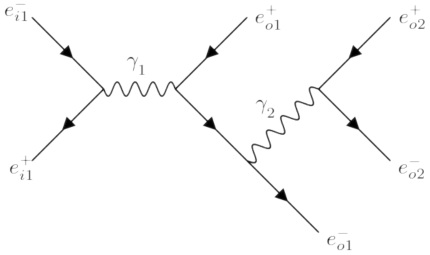
Generating Diagrams in Tree-Level QED - Example

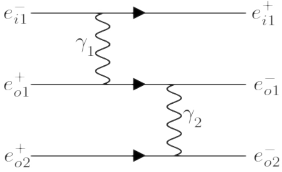


Generating Diagrams in Tree-Level QED - Example

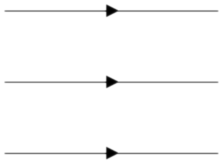


Generating Diagrams in Tree-Level QED - Example



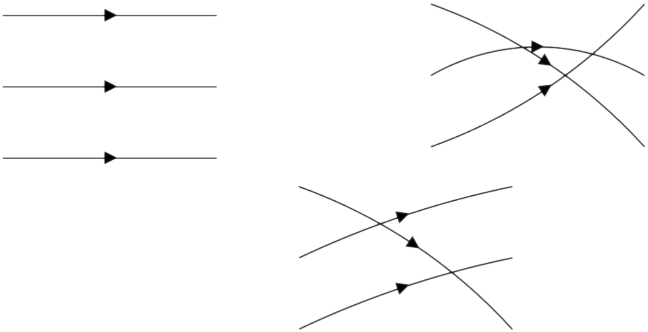


Generating Diagrams in Tree-Level QED - Fermion Permutations

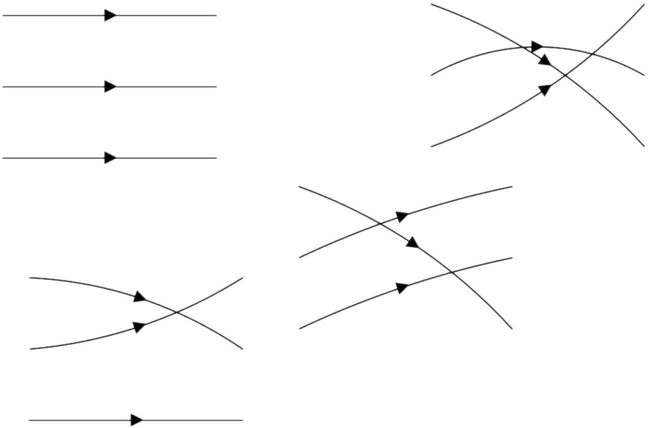


Generating Diagrams in Tree-Level QED - Fermion Permutations

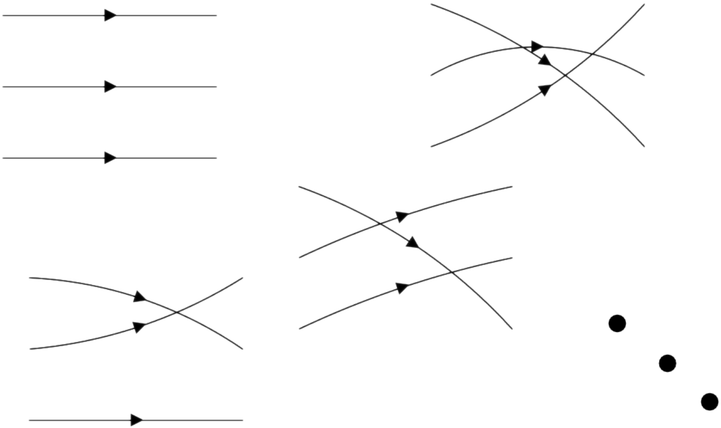
Generating Diagrams in Tree-Level QED - Fermion Permutations



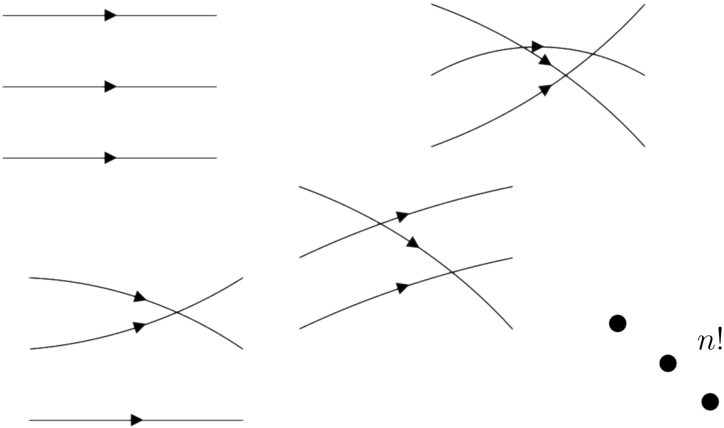
Generating Diagrams in Tree-Level QED - Fermion Permutations



Generating Diagrams in Tree-Level QED - Fermion Permutations

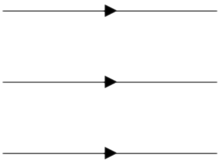


Generating Diagrams in Tree-Level QED - Fermion Permutations

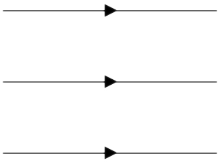


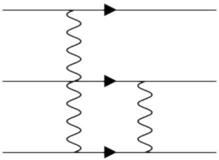
$n!$ permutations

Generating Diagrams in Tree-Level QED - Labelled Plane Trees

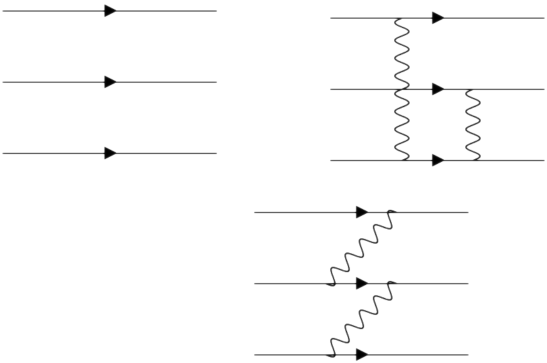


Generating Diagrams in Tree-Level QED - Labelled Plane Trees

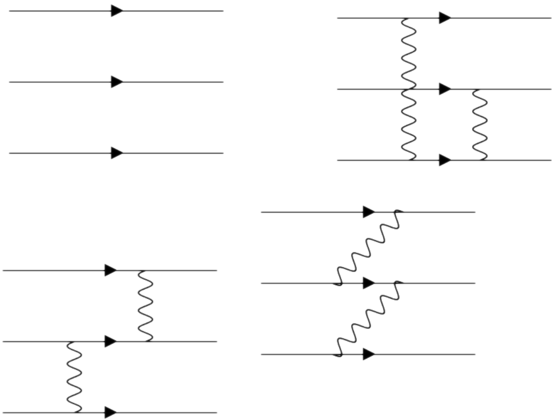




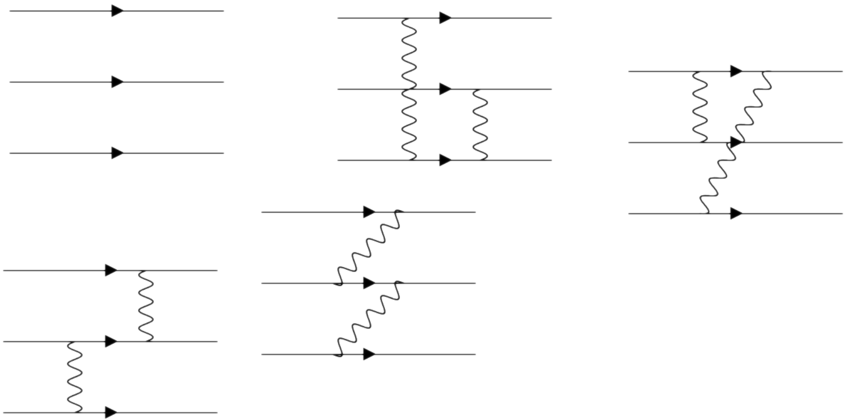
Generating Diagrams in Tree-Level QED - Labelled Plane Trees



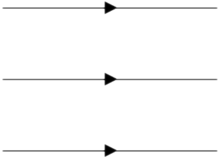
Generating Diagrams in Tree-Level QED - Labelled Plane Trees

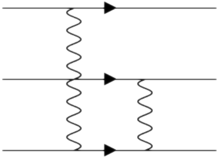


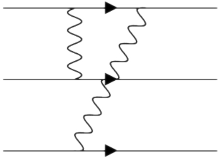
Generating Diagrams in Tree-Level QED - Labelled Plane Trees

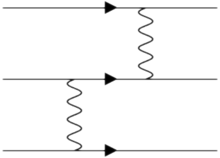


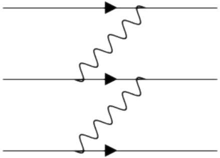
Generating Diagrams in Tree-Level QED - Labelled Plane Trees





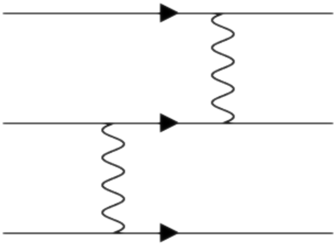




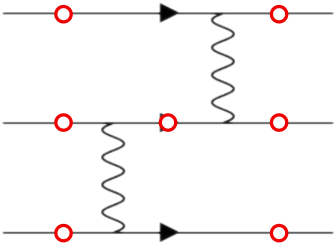


⋱ $\frac{(3n-3)!}{(2n-1)!}$ trees

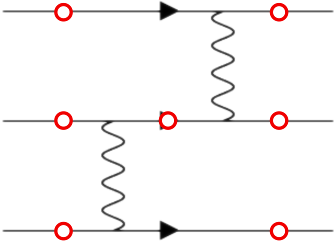
Generating Diagrams in Tree-Level QED - Connecting External Photons



Generating Diagrams in Tree-Level QED - Connecting External Photons

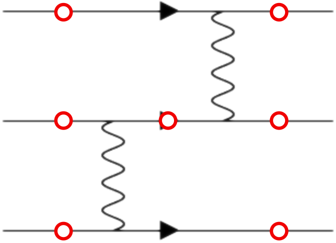


Generating Diagrams in Tree-Level QED - Connecting External Photons



- $\binom{m+3n-3}{3n-3}$ ways to connect external photons
  where n is the number of fermion lines and m is the number of photons

Generating Diagrams in Tree-Level QED - Connecting External Photons



- $\binom{m+3n-3}{3n-3}$ ways to connect external photons
  where n is the number of fermion lines and m is the number of photons
- finally, permute the photons: $m!$

Generating Diagrams in Tree-Level QED - Scaling

$$N_{\text{diags}}(e, u, t, m) = \frac{(m + 3n - 3)!}{(2n - 1)!} \cdot e! \cdot u! \cdot t!$$

where $n := e + u + t$

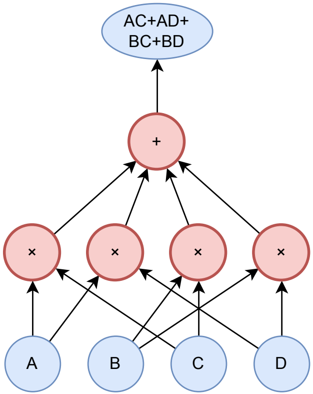
Generating Diagrams in Tree-Level QED - Scaling

$$N_{\text{diags}}(e, u, t, m) = \frac{(m + 3n - 3)!}{(2n - 1)!} \cdot e! \cdot u! \cdot t!$$

where $n := e + u + t$

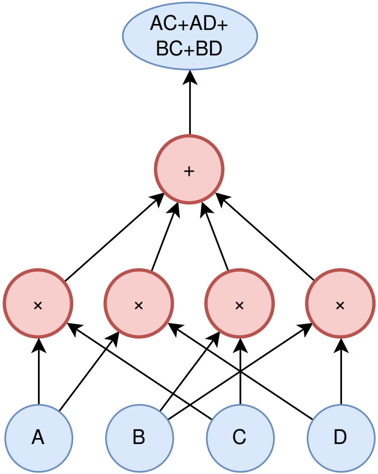
Transform to Computable DAG

- Represent the computation for the matrix element as a directed acyclic graph (DAG)
- Use `ComputableDAGs.jl`
- Allows dynamic construction, analysis, scheduling, and execution (threaded, GPU, etc.)

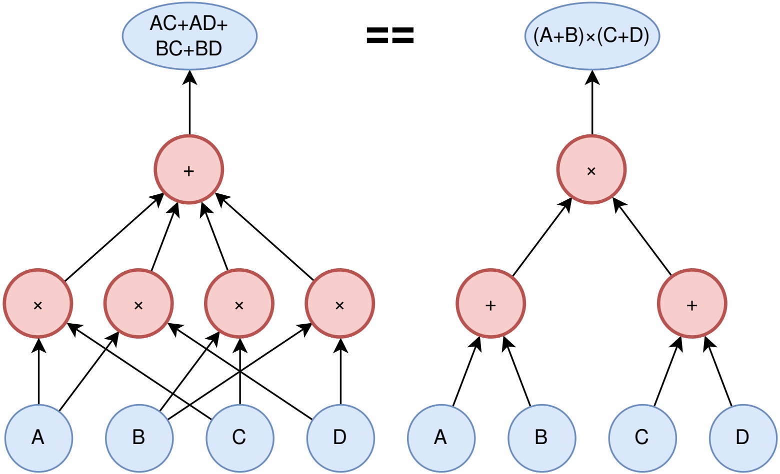


DRESDEN concept HZDR

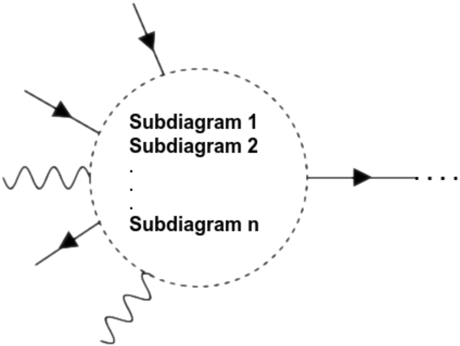
Transform to Computable DAG - Distributivity



Transform to Computable DAG - Distributivity



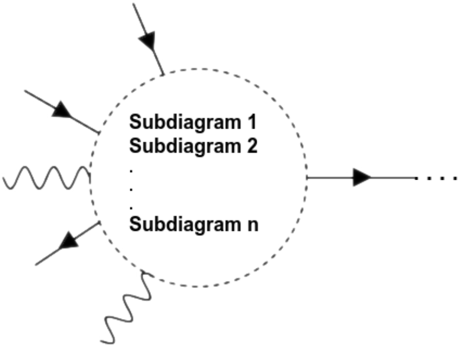
Transform to Computable DAG



- Only $2^{n-1} - 1$ possible inner particle momenta, not factorial[1]

[1] Mauro Moretti, Thorsten Ohl, and Jürgen Reuter. *O'Mega: An Optimizing Matrix Element Generator*. 2001. arXiv: hep-ph/0102195 [hep-ph]. URL: https://arxiv.org/abs/hep-ph/0102195.

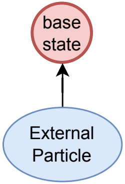
Transform to Computable DAG



- Only $2^{n-1} - 1$ possible inner particle momenta, not factorial[1]
- $\implies$ Consider subdiagrams consisting of particle sets

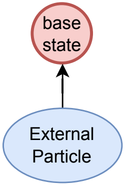
[1] Mauro Moretti, Thorsten Ohl, and Jürgen Reuter. *O'Mega: An Optimizing Matrix Element Generator*. 2001. arXiv: hep-ph/0102195 [hep-ph]. URL: https://arxiv.org/abs/hep-ph/0102195.

Transform to Computable DAG - Base States and Propagators



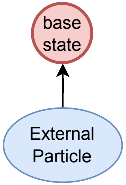
- External particle type, spin or polarization, and momentum as input

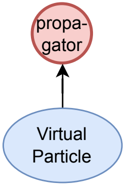
Transform to Computable DAG - Base States and Propagators



- External particle type, spin or polarization, and momentum as input
- Output "propagated" value for the particle set containing only this particle (external leg)

Transform to Computable DAG - Base States and Propagators

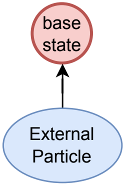


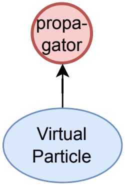


- External particle type, spin or polarization, and momentum as input
- Output "propagated" value for the particle set containing only this particle (external leg)

- Phase space point, momentum contribution map, and virtual particle type as input

Transform to Computable DAG - Base States and Propagators





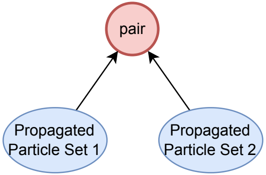
- External particle type, spin or polarization, and momentum as input
- Output "propagated" value for the particle set containing only this particle (external leg)

- Phase space point, momentum contribution map, and virtual particle type as input
- Output a propagator to be used later to propagate the particle set values

DRESDEN concept HZDR

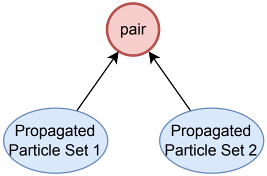
Transform to Computable DAG - Pairing

- Take two disjunct particle sets, multiply them, add vertex term



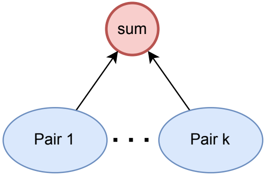
Transform to Computable DAG - Pairing

- Take two disjunct particle sets, multiply them, add vertex term
- Output unpropagated value for the particle set containing the particles of both subsets



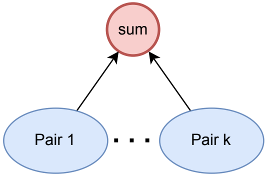
Transform to Computable DAG - Sum Pairs

- Take all unpropagated particle sets of the same contents



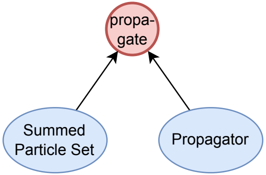
Transform to Computable DAG - Sum Pairs

- Take all unpropagated particle sets of the same contents
- Output unpropagated sum of all values of particle sets containing the given particles



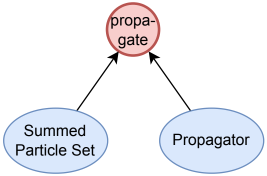
Transform to Computable DAG - Propagate Summed Pairs

- Take a summed unpropagated value for a particle set and the respective virtual particle's propagator



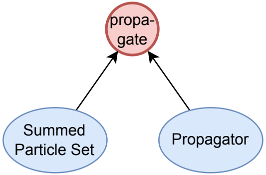
Transform to Computable DAG - Propagate Summed Pairs

- Take a summed unpropagated value for a particle set and the respective virtual particle's propagator
- Output propagated sum of all values of particle sets containing the given particles



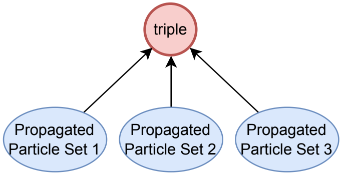
Transform to Computable DAG - Propagate Summed Pairs

- Take a summed unpropagated value for a particle set and the respective virtual particle's propagator
- Output propagated sum of all values of particle sets containing the given particles
- Repeat until subdiagrams contain half of an entire diagram



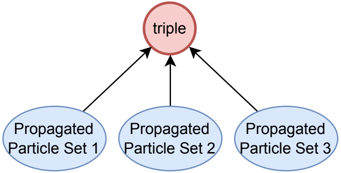
Transform to Computable DAG - Triples

- Like pair, but take three summed and propagated subdiagrams



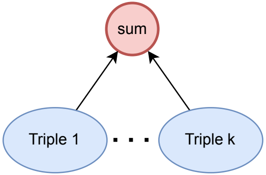
Transform to Computable DAG - Triples

- Like pair, but take three summed and propagated subdiagrams
- Output summed value for a number of diagrams



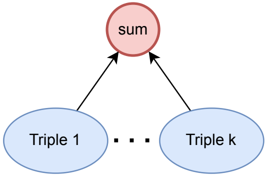
Transform to Computable DAG - Sum Triples

- Like sum pairs, but for the triples



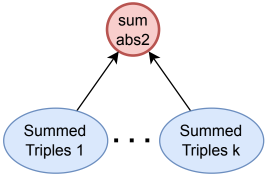
Transform to Computable DAG - Sum Triples

- Like sum pairs, but for the triples
- Output summed value for *all* diagrams, for the given process and one spin and polarization combination



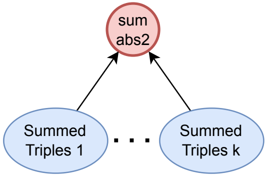
Transform to Computable DAG - Matrix Element

- Finally, `abs2` sum over spin and polarization combinations



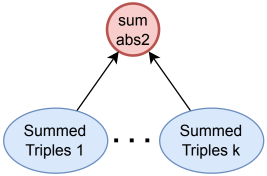
Transform to Computable DAG - Matrix Element

- Finally, `abs2` sum over spin and polarization combinations
- Output the computed squared matrix element



Transform to Computable DAG - Matrix Element

- Finally, `abs2` sum over spin and polarization combinations
- Output the computed squared matrix element



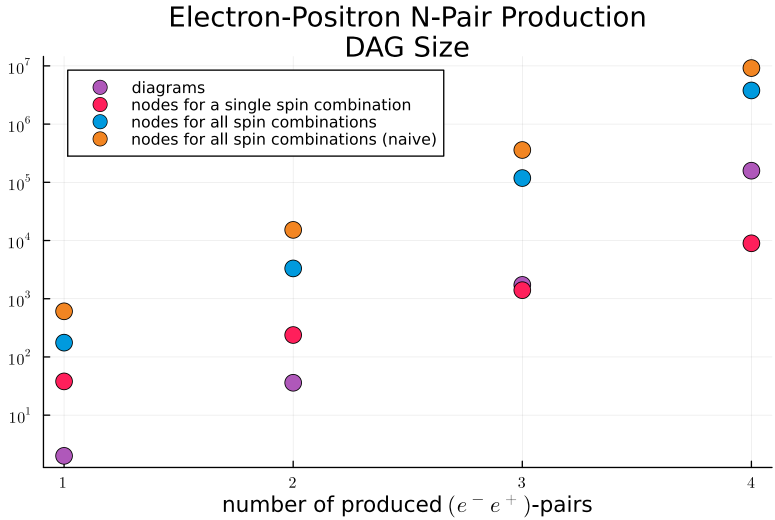
Done!

Results - Reproducibility

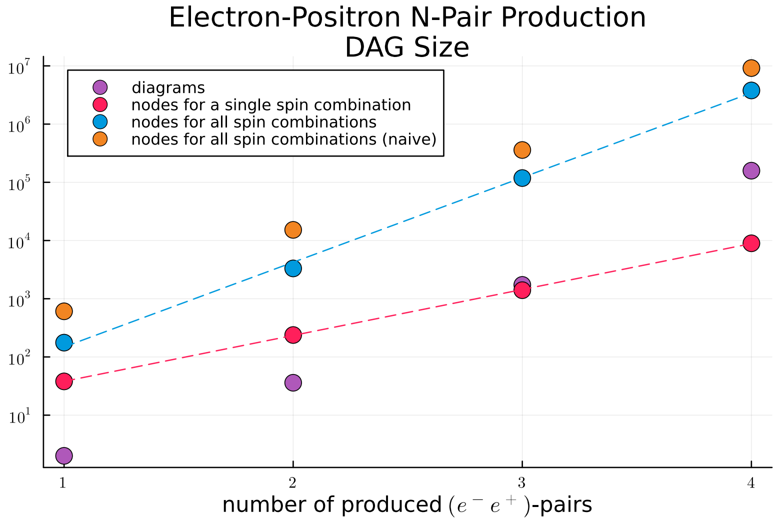
The following results can be reproduced using the Jupyter notebooks at this URL:

`https://github.com/AntonReinhard/QEDFeynmanDiagrams.jl/tree/profiling/profiling`

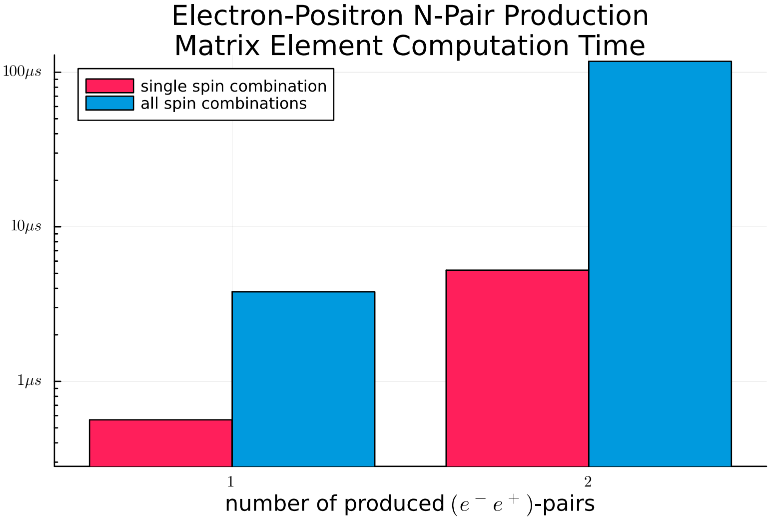
Results - $e^- + e^+ \to n(e^- + e^+)$ - DAG Sizes



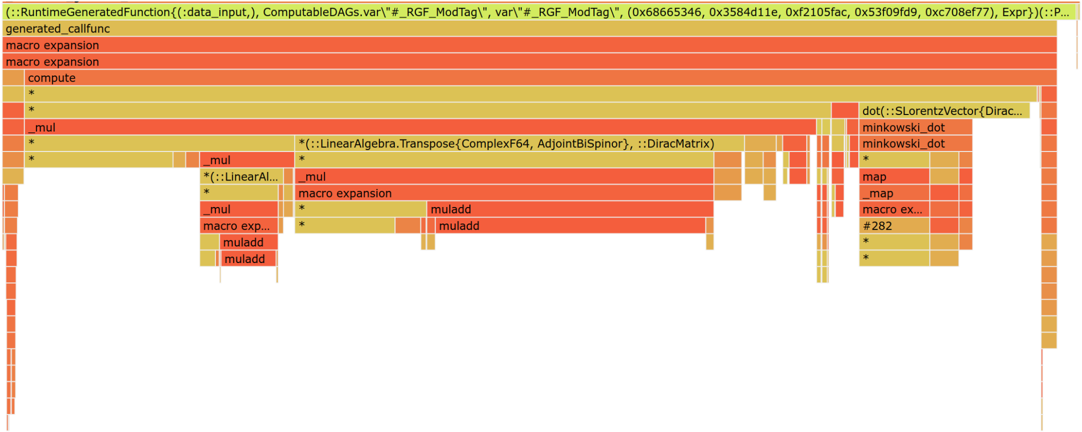
Results - $e^- + e^+ \rightarrow n(e^- + e^+)$ - DAG Sizes



Results - $e^- + e^+ \rightarrow n(e^- + e^+)$ - DAG Computation Time



Results - Profiling Flamegraph



DRESDEN concept HZDR

Summary

The soon™ to be registered package `QEDFeynmanDiagrams.jl` can:

- Generate ComputableDAGs for arbitrary scattering processes in tree-level perturbative QED [2]
- Maximally reuse results, even across different spin and polarization combinations
- Generate matrix elements for synced spins or polarizations with result reuse
- Provide documentation with usage examples

The code is already publicly available

`ComputableDAGs.jl`

`QEDFeynmanDiagrams.jl`

[2] currently excluding muons and tauons until they are implemented in `QuantumElectrodynamics.jl`

Outlook

- Relative negation of diagrams with exchanged fermions is not yet implemented
- Compare to existing solutions (MadGraph5 [2], O'Mega [1], SHERPA [3])
- Extension for other quantum field theories through generalized diagram generation
- Consider vectorization inside the graph
- Find ways to improve startup times

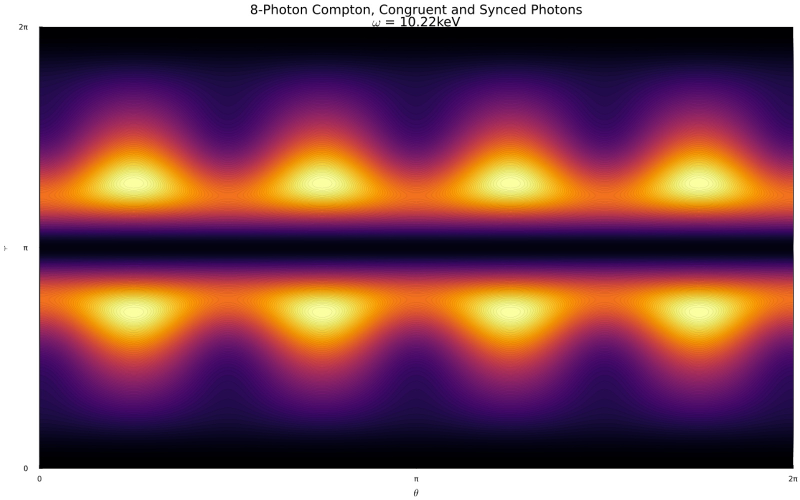
Acknowledgements

Collaborators:

- **Dr. Uwe Hernandez Acosta**[1,2]
- **Simeon Ehrig**[1,2]

[1] Center for Advanced Systems Understanding (CASUS)
[2] Helmholtz-Zentrum Dresden-Rossendorf (HZDR)

Discussion


8-Photon Compton, Congruent and Synced Photons
ω = 10.22keV
2π
π
0
0
π
2π
θ

References

[1] Mauro Moretti, Thorsten Ohl, and Jürgen Reuter. *O'Mega: An Optimizing Matrix Element Generator*. 2001. arXiv: hep-ph/0102195 [hep-ph]. URL: https://arxiv.org/abs/hep-ph/0102195.

[2] Johan Alwall et al. "The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations". In: *Journal of High Energy Physics* 2014.7 (2014), pp. 1–157.

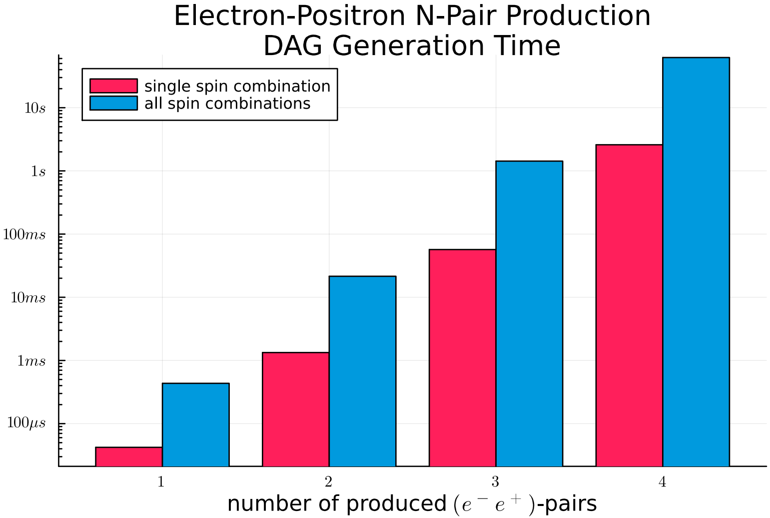
[3] Tanju Gleisberg et al. "Event generation with SHERPA 1.1". In: *Journal of High Energy Physics* 2009.02 (2009), p. 007.

[4] Tim Besard, Christophe Foket, and Bjorn De Sutter. "Effective extensible programming: unleashing Julia on GPUs". In: *IEEE Transactions on Parallel and Distributed Systems* 30.4 (2018), pp. 827–841.

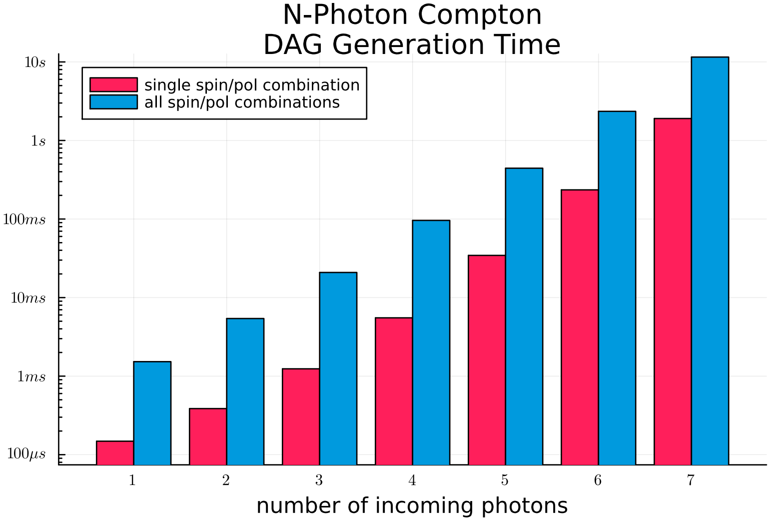
[5] Stefan Karpinski et al. *Why we created julia*. Feb. 2012. URL: https://julialang.org/blog/2012/02/why-we-created-julia/.

[6] Valentin Churavy et al. "Bridging HPC Communities through the Julia Programming Language". In: *arXiv preprint arXiv:2211.02740* (2022).

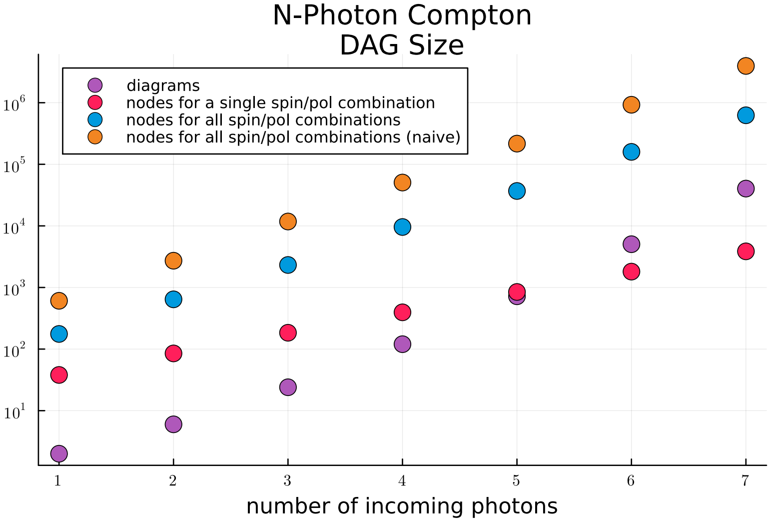
Results - $e^- + e^+ \to n(e^- + e^+)$ - DAG Generation Time



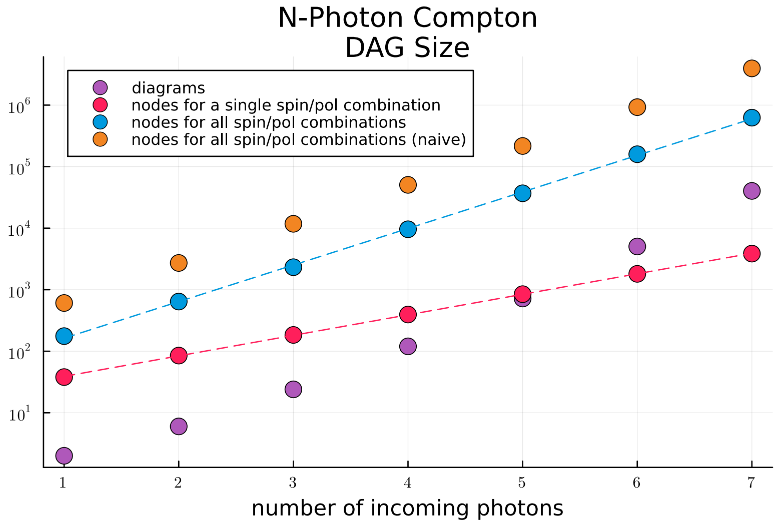
Results - $e^- + k\gamma \rightarrow e^- + \gamma$ - DAG Generation Time



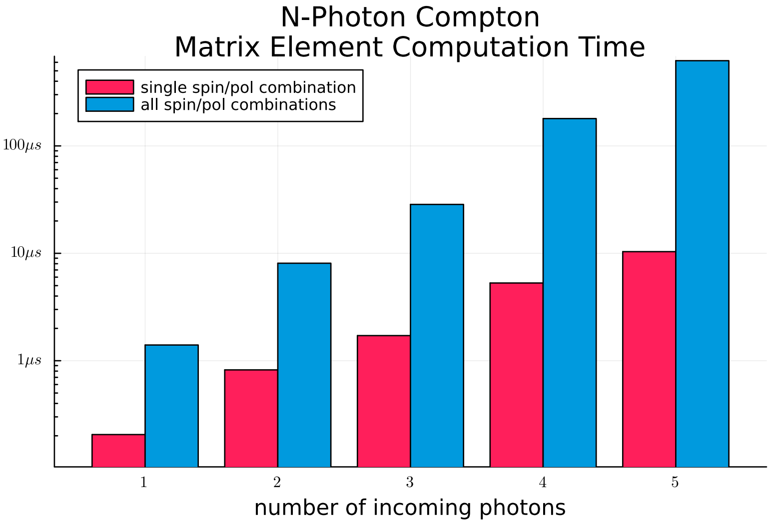
Results - $e^- + k\gamma \to e^- + \gamma$ - DAG Sizes



Results - $e^- + k\gamma \to e^- + \gamma$ - DAG Sizes



Results - $e^- + k\gamma \to e^- + \gamma$ - DAG Computation Time



Benchmarking Machine

Home PC with

- Ryzen 7900X3D
- $2\times$32GB DDR5 RAM @ 6000MHz
- Julia v1.10