QCD tree amplitudes on modern GPUs: A case study for novel event generators

Enrico Bothmann,[a,*] **Joshua Isaacson,**[b] **Max Knobbe,**[a]
Stefan Höche[b] **and Walter Giele**[b]

[a] *Institut für Theoretische Physik, Universität Göttingen, Friedrich-Hund-Platz 1, 37077 Göttingen, Germany*
[b] *Fermi National Accelerator Laboratory, Batavia, IL 60510, USA*

E-mail: enrico.bothmann@uni-goettingen.de

For more than a decade the current generation of CPU-based matrix element generators has provided hard scattering events with excellent flexibility and good efficiency. However, they are a bottleneck of current Monte Carlo event generator toolchains, and with the advent of the HL-LHC and more demanding precision requirements, faster matrix elements are needed, especially at intermediate to large jet multiplicities. We present first results of the new BlockGen family of matrix element algorithms, featuring GPU support and novel color treatments, and discuss the best choice to deliver the performance needed for the next generation of accelerated matrix element generators.

41st International Conference on High Energy physics – ICHEP2022
6–13 July, 2022
Bologna, Italy

[*] Speaker

1. Introduction

The success of particle colliders depends on the availability of detailed numerical simulations. Event generators based on Monte-Carlo integration provide this crucial link between theory and experiment [1]. But the compute efficiency of these highly flexible programs has become an issue, given the rising requirements on the perturbative accuracy, and the trend towards more complex or higher-multiplicity processes. This is in particular a major concern for fulfilling the HL-LHC physics program [2].

Standard-candle processes, such as vector boson and top-antitop-pair production in association with jets, require a large number of generated events to match the experimental statistics. In standard setups in use by experiments, one finds that the major bottleneck is the sampling of phase space and the evaluation of tree-level matrix elements [3, 4]. In other words, the high-energy part of the event generation pipeline that directly probes fundamental physics becomes the dominant fraction of computational costs at higher multiplicity.

One way to deliver much faster matrix element calculations can be the use of massive parallelism in the evaluation, e.g. by using specialized hardware such as GPUs [5], which are also increasingly available in the HPC landscape. Using GPUs has not only been explored for matrix elements at tree and loop level [6] but also for evaluating parton density functions [7]. However, a production-ready GPU-enabled event generator suitable for experimental applications is not yet available.

In these proceedings we report on our efforts to fill this gap. We report on our previous publication [8], where we discuss GPU-enabled algorithms dubbed BlockGen for gluon-only matrix elements, and also for the first time show first results of our ongoing work to generalize this previous work to all relevant LHC standard-candle processes that require particularly large simulated samples, starting with vector boson and top-antitop-pair production in association with jets. This generalization includes the use of recent developments in the minimal color decomposition of matrix elements originally proposed in [9].

2. Algorithmic Choices

All BlockGen algorithms are based on Berends–Giele recursion [10] to evaluate matrix elements. Berends–Giele recursion has a favorable exponential scaling with the number of external states n as compared to the factorial scaling of Feynman diagrams. The computation of color factors needed for the assembly of full matrix elements is either carried out in a factorized form, or embedded in the recursion itself [11, 12]. We refer to the factorized option as a color-ordered Berends–Giele (COBG) recursion and to the embedded one as a color-dressed Berends–Giele (CDBG) recursion. As discussed extensively in [8], where we study both variants, the computational characteristics are very different between the two variants. Generally speaking, CDBG requires more memory, but scales better with the number of particles.

For COBG, the cyclic symmetry of the amplitudes, as well as the Kleiss–Kuijf relations can be used to construct a minimal basis of color factors and corresponding ordered amplitudes. For gluon-only amplitudes, this minimal basis can be constructed using the adjoint representation, and has been known for some time [13]. One finds that this minimal basis contains $(n-2)!$ independent gluon amplitudes. For the general QCD case, i.e. for arbitrary numbers of quarks and gluons, the minimal basis has been found in 2013 [9], and the corresponding color factors have been calculated in 2015 [14]. Its size is $(n-2)!\,\prod_{i=1}^{n_f} k_i/k!$ for $k = \Sigma_{i=1}^{n_f} k_i$ quark pairs, where the k_i are the numbers of quark pairs of flavor i, and n_f is the number of quark flavors. Using a minimal basis of d amplitudes is particularly important because the squared color-summed amplitude has $d(d+1)/2$ terms, i.e. scales with the number of amplitudes *squared*, such that this squared summing can become the bottleneck for large multiplicities n. In [8], we showed for the gluon-only case that the squared summing begins to take more time than the evaluation of the amplitudes for $n > 8$.

3. Computing Performance

In Fig. 1 we reproduce the main result from [8], a GPU-vs-CPU comparison between various variants of BlockGen and the existing Comix and Amegic programs, showing the average time needed to generate one gluon-only event against the final-state multiplicity n_{out}. For the CPU results, all available (hyper)threads of the chip have been used to compete against the parallel evaluation on the GPU, such that in both cases the entire compute capabilities of the chips are used. For more details on the hardware used, see the caption of the figure. One finds that for $n_{\text{out}} \leq 6$, the fastest algorithm is the GPU-accelerated BlockGen-CO$_\Sigma$, which uses color-ordering and -summation, and it is between 4 and 25 times faster than the best competing CPU code. For $n_{\text{out}} > 6$, BlockGen-CD$_{\text{MC}}$ performs best, which uses color-dressing and -sampling. It is about a factor 4 faster than the best competing CPU code. Consulting the ratio plots, one can moreover see that the GPU-acceleration of BlockGen-CO$_\Sigma$ yields a speed-up factor of 20–25, compared to the evaluation on the CPU. For the caveats of these findings and a more detailed discussion in general, we kindly refer the reader to [8].

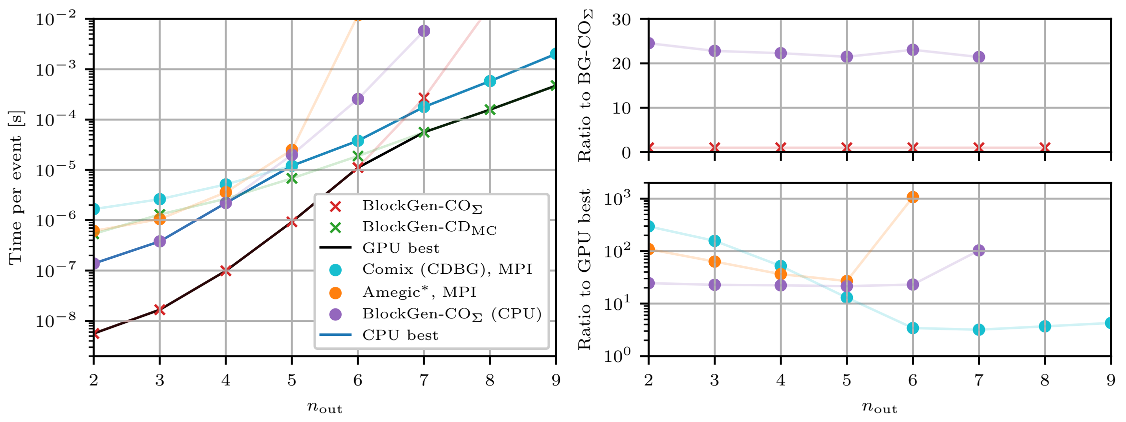

Figure 1: The timings for GPU-based (crosses) and CPU-based (dots) algorithms are compared against each other as a function of gluon multiplicity. The CPU numbers are all generated on an Intel® Xeon® E5-2650 v2 8-core CPU, (2.60 GHz, 20 MB cache), while all the GPU numbers are generated on a NVIDIA V100 (16 GB global memory, 5,120 CUDA cores, 6144 KB L2 cache). The MPI versions are run on 16 threads, and the timing for the color summed algorithm is divided by a factor of 16 to mimic the improvements that would occur from MPI. Furthermore, a modified version of Amegic is used in order to perform helicity sampling. The ratio plots on the right give a better visualization of the relevant speed-ups. Figure reproduced from [8].

In Fig. 2, we present for the first time preliminary BlockGen results after generalizing the implementation to treat full QCD using the minimal basis discussed in Sec. 2 and selected electroweak processes. To our knowledge, this is the first time that the minimal basis is implemented in a code aimed at production. The implementation is not yet ported to use GPU acceleration, such that we compare single-threaded CPU performance only, with the Comix matrix element generator. The left-hand plot shows e^+e^- pair production in association with jets and the right-hand one shows $t\bar{t}$ pair production in association with jets, both at a pp collider. The time per event is given against the number of outgoing jets. One finds that BlockGen performs better than Comix in the e^+e^- case, and equally well in the $t\bar{t}$ case. The relative speed-ups are higher for the e^+e^- case, because there are always at least two fermion pairs, one quark-antiquark and one lepton-antilepton pair, while the $t\bar{t}$ can have one fermion pair less and is thus dominated by the gluonic channels. We have found that channels containing quarks profit strongly from using the minimal basis (we find a 40x speed-up for $d\bar{d} \to usc\bar{u}\bar{s}\bar{c}$ events compared to Comix), while gluonic channels perform similarly.

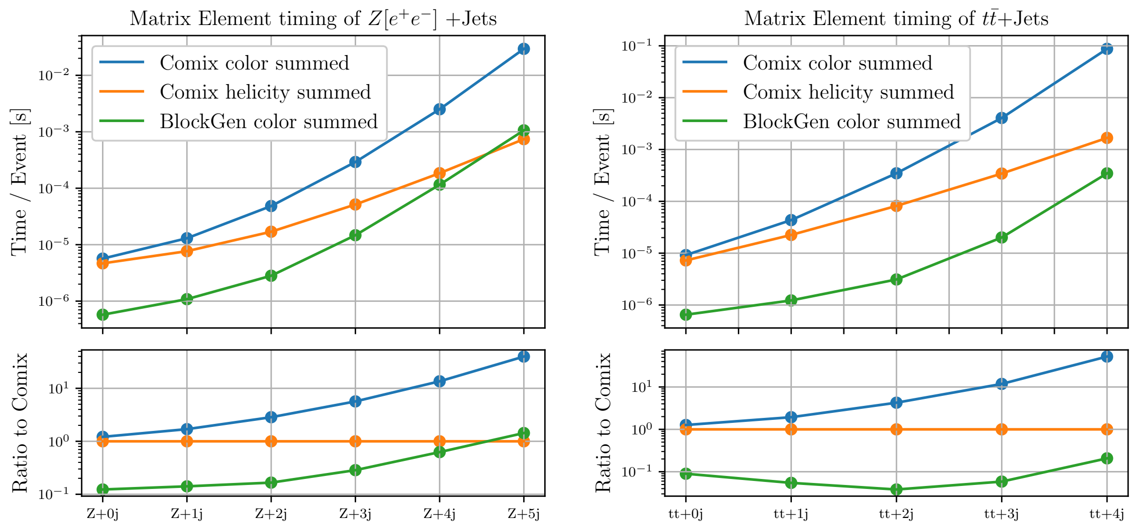

Figure 2: The timings for different single-threaded CPU-based algorithms are compared against each other for $Z[e^+e^-]$ + jets and $t\bar{t}$ + jets generation. The CPU numbers are generated on an Intel® i3-8300 (3.70GHz).

4. Outlook

We have presented the most important results of our previous publication about GPU-accelerated gluon-only matrix element evaluation [8], and we have shown preliminary results from our current effort to generalize our implementation to the full QCD sector and to the most relevant LHC processes, namely e^+e^- and $t\bar{t}$ production in association with jets. In our implementation, the full QCD sector uses a minimal basis of color-ordered amplitudes, which gives us good performance already for our CPU-only implementation. Due to our previous results, we expect a speed-up of 10–20 in a chip-to-chip comparison by our ongoing GPU port.

In the future, we plan to complement the matrix element calculation with a recursive GPU-accelerated phase-space generator, to construct a complete GPU tree-level generator. Since this component is the current bottleneck of LHC event generation of standard candle processes like the above mentioned, we then plan to use our accelerated generator as a plug-in for the Sherpa event generation framework, to remove that bottleneck in time for the remainder of Run-III, and to meet the computational demands of the HL-LHC era.

References

[1] B. Webber, *Monte Carlo Simulation of Hard Hadronic Processes*, *Ann. Rev. Nucl. Part. Sci.* **36** (1986) 253; A. Buckley et al., *General-purpose event generators for LHC physics*, *Phys. Rept.* **504** (2011) 145 [1101.2599].

[2] P. Calafiura et al., *ATLAS HL-LHC Computing Conceptual Design Report*, Tech. Rep. CERN-LHCC-2020-015. LHCC-G-178, CERN, Geneva (Sept., 2020); HSF Physics Event Generator WG collaboration, *Challenges in Monte Carlo Event Generator Software for High-Luminosity LHC*, *Comput. Softw. Big Sci.* **5** (2021) 12 [2004.13687].

[3] S. Höche, S. Prestel and H. Schulz, *Simulation of Vector Boson Plus Many Jet Final States at the High Luminosity LHC*, *Phys. Rev.* **D100** (2019) 014024 [1905.05120].

[4] E. Bothmann, A. Buckley, I.A. Christidi, C. Gütschow, S. Höche, M. Knobbe et al., *Accelerating LHC event generation with simplified pilot runs and fast PDFs*, 2209.00843.

[5] S. Höche et al., *Working Group Report: Computing for Perturbative QCD*, in *Proceedings, 2013 Community Summer Study on the Future of U.S. Particle Physics: Snowmass on the Mississippi (CSS2013): Minneapolis, MN, USA, July 29-August 6, 2013*, 2013, http://www.slac.stanford.edu/econf/C1307292/docs/ComputingFrontier/PQCD-45.pdf [1309.3598]; HEP Software Foundation collaboration, *A Roadmap for HEP Software and Computing R&D for the 2020s*, *Comput. Softw. Big Sci.* **3** (2019) 7 [1712.06982]; A. Dainese et al., eds., *Report on the Physics at the HL-LHC,and Perspectives for the HE-LHC*, vol. 7. 7/2019 of *CERN Yellow Reports: Monographs*, CERN, Geneva, Switzerland (2019), 10.23731/CYRM-2019-007.

[6] K. Hagiwara et al., *Fast calculation of HELAS amplitudes using graphics processing unit (GPU)*, *The European Physical Journal C* **66** (2010) 477; K. Hagiwara et al., *Calculation of HELAS amplitudes for QCD processes using graphics processing unit (GPU)*, *The European Physical Journal C* **70** (2010) 513; J. Kanzaki, *Application of graphics processing unit (GPU) to software in elementary particle/high energy physics field*, *Procedia Computer Science* **4** (2011) 869; K. Hagiwara et al., *Fast computation of MadGraph amplitudes on graphics processing unit (GPU)*, *The European Physical Journal C* **73** (2013) 2608; J. Kanzaki, *Monte Carlo integration on GPU*, *Eur. Phys. J.* **C71** (2011) 1559 [1010.2107]; A. Valassi et al., *Design and engineering of a simplified workflow execution for the MG5aMC event generator on GPUs and vector CPUs*, *EPJ Web Conf.* **251** (2021) 03045 [2106.12631]; W.T. Giele et al., *Thread-scalable evaluation of multi-jet observables*, *The European Physical Journal C* **71** (2011) ; H.-Z. Wu et al., *ZMCintegral: a Package for Multi-Dimensional Monte Carlo Integration on Multi-GPUs*, *Comput. Phys. Commun.* **248** (2020) 106962 [1902.07916]; G. Grasseau et al., *Hybrid implementation of the VEGAS*

Monte-Carlo algorithm, in *Proceedings, GPU Computing in High-Energy Physics (GPUHEP2014): Pisa, Italy, September 10-12, 2014*, pp. 103–108, 2015, DOI; S. Carrazza and J.M. Cruz-Martinez, *VegasFlow: accelerating Monte Carlo simulation across platforms*, *arXiv e-prints* (2020) arXiv:2010.09341 [2010.09341]; S. Carrazza et al., *Towards the automation of Monte Carlo simulation on GPU for particle physics processes*, in *25th International Conference on Computing in High-Energy and Nuclear Physics*, 5, 2021 [2105.10529]; F. Yuasa et al., *Acceleration of Feynman loop integrals in high-energy physics on many core GPUs*, *J. Phys. Conf. Ser.* **454** (2013) 012081; S. Borowka et al., *A GPU compatible quasi-Monte Carlo integrator interfaced to pySecDec*, *Comput. Phys. Commun.* **240** (2019) 120 [1811.11720]; D. Schouten et al., *Accelerated matrix element method with parallel computing*, *Computer Physics Communications* **192** (2015) 54; G. Grasseau et al., *Matrix element method for high performance computing platforms*, *Journal of Physics Conference Series* **664** (2015) 092009; G. Grasseau et al., *Deployment of a Matrix Element Method code for the ttH channel analysis on GPU's platform*, *EPJ Web Conf.* **214** (2019) 06028.

[7] S. Carrazza et al., *PDFFlow: Parton distribution functions on GPU*, *Comput. Phys. Commun.* **264** (2021) 107995 [2009.06635].

[8] E. Bothmann, W. Giele, S. Hoeche, J. Isaacson and M. Knobbe, *Many-gluon tree amplitudes on modern GPUs: A case study for novel event generators*, *SciPost Phys. Codebases* (2022) 3 [2106.06507].

[9] T. Melia, *Dyck words and multiquark primitive amplitudes*, *Phys. Rev. D* **88** (2013) 014020 [1304.7809]; T. Melia, *Dyck words and multi-quark amplitudes*, *PoS* **RADCOR2013** (2013) 031.

[10] F.A. Berends and W.T. Giele, *Recursive calculations for processes with n gluons*, *Nucl. Phys.* **B306** (1988) 759; S. Badger et al., *Comparing efficient computation methods for massless QCD tree amplitudes: Closed analytic formulas versus Berends-Giele recursion*, *Phys. Rev.* **D87** (2013) 034011 [1206.2381].

[11] C. Duhr, S. Höche and F. Maltoni, *Color-dressed recursive relations for multi-parton amplitudes*, *JHEP* **08** (2006) 062 [hep-ph/0607057].

[12] W.T. Giele et al., *Efficient color-dressed calculation of virtual corrections*, *Nuclear Physics B* **840** (2010) 214.

[13] F.A. Berends and W. Giele, *The six-gluon process as an example of Weyl-van der Waerden spinor calculus*, *Nucl. Phys.* **B294** (1987) 700; V. Del Duca et al., *New color decompositions for gauge amplitudes at tree and loop level*, *Nucl. Phys.* **B571** (2000) 51 [hep-ph/9910563]; V. del Duca et al., *Factorization of tree QCD amplitudes in the high-energy limit and in the collinear limit*, *Nucl. Phys.* **B568** (2000) 211 [hep-ph/9909464].

[14] H. Johansson and A. Ochirov, *Color-Kinematics Duality for QCD Amplitudes*, *JHEP* **01** (2016) 170 [1507.00332]; T. Melia, *Proof of a new colour decomposition for QCD amplitudes*, *JHEP* **12** (2015) 107 [1509.03297].