Introduction

The original HIJING [1] (Heavy Ion Jet Interaction Generator) Monte Carlo model was developed by M. Gyulassy and X.-N. Wang with special emphasis on the role of minijets in proton-proton (pp), proton-nucleus (pA) and nucleus-nucleus (AA) reactions at collider energies in a wide range from 5 GeV to 2 TeV. Since the release of the first HIJING version a number of underlying libraries has undergone a major upgrade connected with structural changes, and they got rewritten to C++, becoming a standard now in the high-energy community. Hence, we decided to upgrade HIJING accordingly, to be a genuine, C++ based, modular event generator, with the most recent versions of PYTHIA8 [2] and LHAPDF6 [3], and be compatible with the experimental frameworks. We present the current status of HIJING++ for LHC energies. Here, we summarize the structure and the speed gain due to parallelization of the new program code, also presenting some comparison between experimental data.

The HIJING++ program

The code has several entry points for user provided and built-in features, which will be present in the final, public code:

- HIJING class contains all the physics were coded in the class HIJPhysics, which has several entry points for user provided and built-in modules and optimizations for modularity and compiler’s improved parallel supports.

During the designing we took great care of modularity, portability, maintainability and speed. The colored boxes represent the newly included HIJING modules and modifications neglecting cross-links.

The HIJing class contains all the physics were coded in the FORTRAN subroutines, based on the latest version of HIJING version 2.552 [4]. The high-energy nuclear physics related part (hard collisions, soft collisions, fragmentation, Cronin effect, jet quenching) are moved to the HIJPhysics class, where they can be called modularly, with the possibility to alter them to user supplied modules.

Due to the object oriented being of the C++, the original structure was optimized for modularity and compiler’s improved parallel supports. The HIJQueue and HIJManager classes are responsible for distributing the separate HIJing events in a parallel environment.

A useful new feature is the HIJAnalysis class, being an easy-to-use interface to user defined data collections and Rivet [5] tools within the run. The code has several entry points for user provided and built-in histogram collection modules, which are configurable in the user (main) code with one line commands.

Ongoing developments

HIJING++ is currently under validation, before release. In parallel to the testing of the code, a sophisticated theoretical development of the model is ongoing. Here we list some of these updates and features, which will be present in the final, public code:

- The original HIJING shadowing function, and also its modified, Q2 dependent version are producing too much shadowing, so we are considering to implement other shadowing models.

- A contemporary jet energy loss module is under development. This module is enable users to include various jet-quenching models in the future, such as we include Gyulassy–Lévai–Vitev (GLV) [6] (see poster JET-25 for more details).

- Tuning of parameters relevant for physics and runtime (such as soft-hard separation scale, Cronin effect strength, thread management for parallel architectures such as GPUs) is also on the wish list.

- The code has several entry points for user provided and built-in features, which will be present in the final, public code.

- HIJING++ is currently under validation, before release. In parallel to the testing of the code, a sophisticated theoretical development of the model is ongoing. Here we list some of these updates and features, which will be present in the final, public code:

- The original HIJING shadowing function, and also its modified, Q2 dependent version are producing too much shadowing, so we are considering to implement other shadowing models.

- A contemporary jet energy loss module is under development. This module is enable users to include various jet-quenching models in the future, such as we include Gyulassy–Lévai–Vitev (GLV) [6] (see poster JET-25 for more details).

- Tuning of parameters relevant for physics and runtime (such as soft-hard separation scale, Cronin effect strength, thread management for parallel architectures such as GPUs) is also on the wish list.

- The code has several entry points for user provided and built-in features, which will be present in the final, public code.

- HIJING++ is currently under validation, before release. In parallel to the testing of the code, a sophisticated theoretical development of the model is ongoing. Here we list some of these updates and features, which will be present in the final, public code:

- The original HIJING shadowing function, and also its modified, Q2 dependent version are producing too much shadowing, so we are considering to implement other shadowing models.

- A contemporary jet energy loss module is under development. This module is enable users to include various jet-quenching models in the future, such as we include Gyulassy–Lévai–Vitev (GLV) [6] (see poster JET-25 for more details).

- Tuning of parameters relevant for physics and runtime (such as soft-hard separation scale, Cronin effect strength, thread management for parallel architectures such as GPUs) is also on the wish list.

- The code has several entry points for user provided and built-in features, which will be present in the final, public code.

Summary

Present milestone aims to communicate the status of this software development, moreover, give perspectives for the forthcoming applicabilities and features of the soon-to-be-released open source HIJING++ for the next generation of heavy-ion collision measurement, simulations, and facilities at future colliders.

For more info and updates about the project, preliminary datasets, requests and contact details check our webpage on https://gitlab.kfki.hu/hijing/QuarkMatter2018/.

Acknowledgement

This work was supported by the Hungarian-Chinese cooperation: Wigner HU–CASR–CASR bilateral grant, Hungarian National Research Fund OTKA grant K125640 and THOR COST action 15213. We acknowledge the support of the Wigner GPU laboratory.

References