APPENDIX A
ARTIFACT DESCRIPTION APPENDIX: EULERIAN ALGORITHMS FOR THE DISCRETIZATION OF (PLASMA) KINETIC EQUATIONS

A. Abstract

While fluid models are common tools in the study of plasmas, many of these systems, whether in astrophysics or the lab, are only weakly collisional and far from equilibrium, making them more accurately described by kinetic equations. Kinetic equations can be computationally demanding due to the need to solve for the distribution function of the particles in a higher dimensional phase space, with position and velocity coordinates. Despite this challenge, the motivation for solving the plasma kinetic equation is large as there remains a vast array of questions concerning collisionless dynamics in real plasma systems. Here we present algorithms in an Eulerian framework for the discretization of the plasma kinetic equation, using a high-order discontinuous Galerkin finite element method due to its arithmetic intensity and parallelizability. Scaling and performance of the algorithm are discussed, especially in comparison to the traditional particle-in-cell method. Benchmarks of the algorithm are presented as well.

B. Description

1) Check-list (artifact meta information):

- **Algorithm: Discontinuous Galerkin Finite Element Method**
- **Program: Gkeyll, C++, LuaJIT**
- **Compilation: Intel C Compiler, LuaJIT compiler**
- **Hardware: Intel Xeon Platinum 8160 (“Skylake”), Intel Xeon Phi 7250 (“Knight?s Landing”)**
- **Output: ADIOS (.bp files)**
- **Publicly available?: Yes**

2) How software can be obtained : The complete source code can be downloaded from: https://bitbucket.org/ammarhakim/gkyl/src/default/
The executable, and all the software dependencies can also be installed through the anaconda distribution via the command “conda install -c gkyl gkyl”

3) Hardware dependencies: None

4) Software dependencies: C++, LuaJIT, MPI (must have MPI-3), ADIOS, Eigen

5) Datasets: None

C. Installation

Installation on the Stampede 2 cluster, where all the results shown were obtained, can be achieved by simply running the mkdeps.stampede.sh script in the machines directory in the source code upon pulling the source code from bitbucket. After all the dependencies are built, installation can be completed via running the shell script configure.stampede.sh and then using the waf build system via the command “./waf build install”

D. Experiment workflow

Not applicable

E. Evaluation and expected result

Similar plots to those shown in the paper can be produced using the postgkyl post-processing package, which can also be downloaded through the anaconda distribution via “conda install -c gkyl postgkyl”
Upon running the sample two-stream instability input file using the command “ /gkylsoft/gkyl/bin/gkyl two_stream.lua” (or wherever you ultimately elect to install gkyl and whatever you call the input file) which can be found on the documentation for the code: http://gkyl.readthedocs.io/en/latest/gkyl/es-two-stream.html, the phase space plot of the distribution function can be produced with the following postgkyl command line command: “pgkyl -f two-stream_elc_2.bp interpolate -b ms -p 2 plot”
Note that the file handle after the “-f” flag should be whatever you name the input file.

F. Experiment customization

The relevant compiler flags are included in the configure.stampede.sh but for reference, the optimization flags used are “-O3,-axCORE-AVX512,” and the compiler employed is the Intel 18 C Compiler