On the Performance of GPU Accelerated Meshfree Solvers in Fortran, C++, Python, and Julia

Nischay Ram Mamidi, Kumar Prasun, Harivallabha Rangarajan and Anil Nemili
Department of Mathematics, BITS-Pilani, Hyderabad Campus, Hyderabad, 500078

S.M. Deshpande
Engineering Mechanics Unit, JNCASR, Bengaluru, 560064

Bharatkumar Sharma
NVIDIA

The entropy or *q*-variables based Least Squares Kinetic Upwind Method (*q*-LSKUM) is a meshfree method that belongs to the family of kinetic theory-based upwind schemes for the numerical solution of Euler or Navier-Stokes equations that govern the compressible flows. These schemes are based on the moment-method-strategy, where an upwind scheme is first developed at the Boltzmann level and after taking suitable moments, we arrive at an upwind scheme for the governing conservation laws. *q*-LSKUM is capable of operating on any arbitrary distribution of points. The point distributions can be obtained from structured, unstructured, cartesian, chimera, and FAME meshes or even from point generation algorithms such as quadtree and advancing front methods. The meshfree solvers based on *q*-LSKUM are being extensively used to compute flows around many complex configurations, such as wings of an aircraft, turbomachinery blades, flight vehicle configuration with deflected fins, store separation from aircraft and launch vehicles.

This paper presents the development of GPU accelerated meshfree solvers based on *q*-LSKUM for two-dimensional inviscid compressible flows. The GPU solvers are written in both traditional programming languages, Fortran 90 and C++, and also in modern languages, Python and Julia. The programming interface, CUDA is used to perform the calculations on the GPU. Computations are performed on NVIDIA V100 and A100 GPUs.

Numerical results are shown to verify the correctness of the meshfree GPU solvers on standard two-dimensional inviscid test cases. To assess the computational efficiency and to compare the relative performance of the solvers, benchmark calculations are performed to compare the Rate of Data Processing (RDP) values of the GPU codes on several levels of point distribution. Here, RDP is a cost metric, defined as the wall clock time in seconds per iteration per point. To analyse the overall performance, a detailed investigation of the important global kernels that compute *q-variables* and *q-derivatives* (used to achieve higher-order accuracy), *flux residual,* and *state update* is presented. Numerical experiments have shown that the GPU acceleration of the compute intensive kernel *flux residual* has significantly enhanced the speedup of the solver. Apart from RDP, an analysis of other performance metrics such as utilisation of streaming multiprocessors and device memory, occupancy, and arithmetic intensity (ratio of total floating-point operations to total data movement in bytes) is shown. An assessment of the performance of the traditional and modern languages based GPU solvers on V100 and A100 GPUs will be presented.