GPU accelerated solution of large number of independent ODE systems

Ferenc Hegedűs[1]

[1]Department of Hydrodynamic Systems, Budapest University of Technology and Economics, Budapest, Hungary

In many fields of science, the governing equations of a physical problem can be described by an ordinary differential equation system (ODEs). If the number and size of such a system is smaller, then exploiting the possibilities of massively parallel computing is not feasible, since the time marching of an initial value problem is serial in nature. However, if a large number of parameters are involved during an investigation, the whole problem can be decomposed into a large number of decoupled sub-problems where each parameter combination represents an individual and independent task. These problems are especially suitable for professional GPGPUs using the SIMT (single instruction multiple threads) paradigms. In the last years, many researchers have already implemented their numerical code capable to exploit the high processing powers of GPUs. These are usually specialized codes for the given field of science. General purpose, modular solvers are scarce in the literature. The main aim of the present study/talk is to fill this gap and propose a possible implementation of such a solution technique. An important requirement is the possibility to give the ODE system with as similar syntax as in MATLAB. The code is implemented in C++ and CUDA C software environment. The numerical schemes which can be selected are the 4th order explicit and adaptive Runge-Kutta-Cash-Karp method with 5th embedded error estimation, and the well-known 4th order explicit Runge-Kutta technique with fixed time step. Because the code can handle millions of independent ODE systems, intermediate solutions during the integration are not stored. This is also mandatory from the performance point of view. To be able to extract different properties of a solution (e.g. maximum and minimum of a component, their time instance, etc.), a flexible event handling mechanism is implemented into the code. Moreover, non-smooth dynamics can also be handled easily and efficiently by providing equations describing the dynamics at a switching line (e.g. equations for the impact dynamics); therefor the integration does not have to be stopped.