Computer Physics Communications 244 (2019) 483–486

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

GenASiS Basics: Object-oriented utilitarian functionality for large-scale physics simulations (Version 3)

Reuben D. Budiardja [a,1], Christian Y. Cardall [b,*]

[a] National Center for Computational Sciences, Oak Ridge National Laboratory, Oak Ridge, TN 37831-6354, USA
[b] Physics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831-6354, USA

ARTICLE INFO

Article history:
Received 12 March 2019
Received in revised form 22 May 2019
Accepted 24 May 2019
Available online 6 June 2019

Keywords:
Simulation framework
Object-oriented programming
Fortran 2003

ABSTRACT

GenASiS Basics provides Fortran 2003 classes furnishing extensible object-oriented utilitarian functionality for large-scale physics simulations on distributed memory supercomputers. This functionality includes physical units and constants; display to the screen or standard output device; message passing; I/O to disk; and runtime parameter management and usage statistics. This revision – Version 3 of Basics – includes a significant name change, some minor additions to functionality, and a major addition to functionality: infrastructure facilitating the offloading of computational kernels to devices such as GPUs.

New version program summary

Program Title: SineWaveAdvection, SawtoothWaveAdvection, and RiemannProblem (fluid dynamics example problems illustrating GenASiS Basics); ArgonEquilibrium and ClusterFormation (molecular dynamics example problems illustrating GenASiS Basics)
Program Files doi: http://dx.doi.org/10.17632/6w9ygpygmc.2
Licensing provisions: GPLv3
Programming language: Fortran 2003 (tested with GNU Compiler Collection 8.1.0, Intel Fortran Compiler 18.0.3, Cray Compiler Environment 8.6.5, IBM XL Fortran 16.1.0)
Journal reference of previous version: Computer Physics Communications 214 (2017) 247
Does the new version supersede the previous version?: Yes
Reasons for the new version: This version includes a significant name change, some minor additions to functionality, and a major addition to functionality: infrastructure facilitating the offloading of computational kernels to devices such as GPUs.
Summary of revisions:

The class VariableGroupForm – a major workhorse for handling set of related fields – has been renamed StorageForm.

The ability to use unicode characters in standard output has been added, but is currently only supported by the GNU Compiler Collection (GCC). This capability is used to display exponents as numerical superscripts, as well as symbols such as $\hbar$, $\odot$, and Å in the display of relevant units. It is made operational by the line

```
USE_ISO_10646 = 1
```

which is now included in the machine-specific makefile fragments with a GCC suffix in the Build/-Machines directory.

There are some changes to units and constants. The geometrized units of past releases ($G = c = k = 1$, with a fundamental unit of meter) have been replaced by natural units ($\hbar = c = k = 1$, with MeV as the fundamental unit). Lorentz–Heaviside electromagnetic units are employed (permeability $\mu = 1$; no factors of 4π in the Maxwell equations). This refers to numbers as processed internally by the code; as described in the initial release, users can employ the members of the UNIT singleton for input/output purposes, that is, to specify or display numbers with any available units they wish.

* Corresponding author.
E-mail addresses: reubendb@ornl.gov (R.D. Budiardja), cardallcy@ornl.gov (C.Y. Cardall).

[1] This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

https://doi.org/10.1016/j.cpc.2019.05.014

A number of units have been added, and the specification all units has been put on a more rational basis in keeping with six of the seven standard SI base units (meter, kilogram, second, ampere, kelvin, mole; we have not needed the candela; see [3]). Some physical and astrophysical constants have also been added. All constants have been updated to 2018 values [4].

For notifications to standard output, a few tweaks to ignorability levels have been made in various classes. The default output to screen is now less verbose (ignorability `INFO_1`, our designation for messages of significance just below `WARNING`).

A couple of additions have been made to `MessagePassing`: null subcommunicators are accommodated, and an `AllToAll_V` method has been added to `CollectiveOperation_R_Form`.

Enhancements to timer functionality have been made. The class `TimerForm` now has a member `Level`, which is specified in order to control indentation in screen output. Some functionality has been added to `PROGRAM_HEADER_Singleton` to work with timers. A method `TimerPointer` returns a pointer to a timer with a specified `Handle` (typically a meaningfully named integer). The new members `TimerLevel` and `TimerDisplayFraction` of `PROGRAM_HEADER_Singleton`, which can be set from the command line, can be used to suppress output from timings deemed insignificant, based on timer level or a measured time interval falling below a specified fraction of the total execution time.

The most significant addition in functionality in this release is the addition of infrastructure to offload computational kernels to hardware accelerators such as GPUs using OpenMP device-related directives and runtime library routines in OpenMP 4.5 and later.[2] This infrastructure, implemented in a new subdivision `Devices` (see Fig. 1), provides lower-level routines to perform memory management between the host (CPU) and device (GPU) including data allocation, data movement between host and device, and device-to-host memory address association. The routines are implemented as Fortran wrappers to the OpenMP runtime library and CUDA[3] routines written in C. Additional methods and an option utilizing the lower-level `Devices` routines have been added to our `StorageForm` class. They are: `UpdateHost()` and `UpdateDevice()` to copy data from device to host and host to device, respectively; `AllocateDevice()` to allocate memory on the device mirroring the allocation on the host; and `PinnedOption` as an optional flag to the `Initialize()` method to allocate the host memory in a page-locked region to facilitate faster data transfer between host and device. A detailed description of the implementation of this functionality can be found in [5].

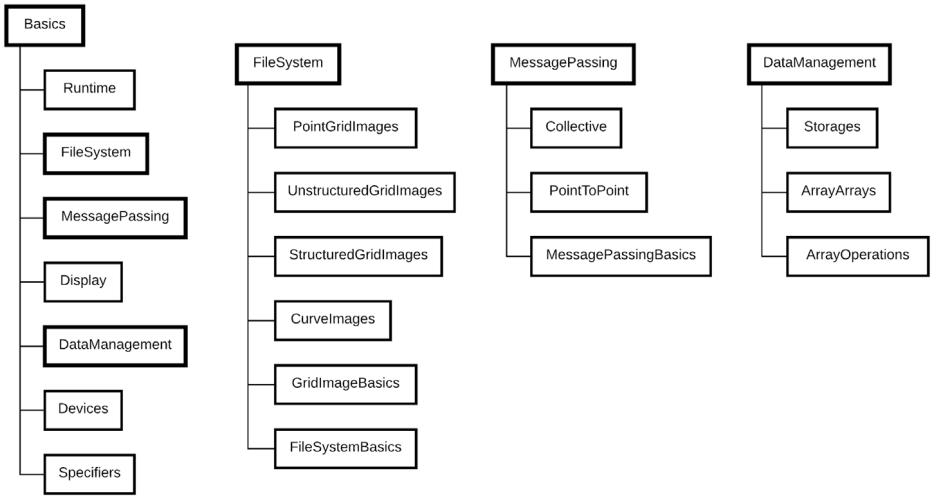

Fig. 1. *Left:* Structure of `Basics` after the introduction of the `Devices` division. *Middle Left:* Substructure of `FileSystem`. *Middle Right:* Substructure of `MessagePassing`. *Right:* Substructure of `DataManagement`. *All:* Solid lines outline the directory hierarchy. Boxes framed with thinner linewidths denote 'leaf' divisions of the code with no additional subdirectories. The compilation order is from bottom to top; thus dependencies essentially flow in reverse, from top to bottom.

To deal with different levels of compiler support for device-related OpenMP directives, we use the preprocessor in some source files in `Devices` to guard against attempted compilation of unsupported features. Preprocessor macro substitution is also utilized in OpenMP directives to switch between multi-threading parallelism on CPUs and offload parallelism to GPUs. Setting the makefile variable `ENABLE_OMP_OFFLOAD` to `1` – which is the default in the machine-specific makefile `Makefile_POWER_XL` for the XL compiler on POWER-based supercomputers – sets the appropriate flags and preprocessing to enable compilation for OpenMP offload parallelism. Alternatively, the command

```
make ENABLE_OMP_OFFLOAD=1
```

sets this variable when `make` is invoked from the command line.

2 https://www.openmp.org/specifications/.
3 https://developer.nvidia.com/about-cuda.

Information regarding the number of devices available to the program, the kind of OpenMP parallelism enabled (i.e. multi-threading or offload), and the selected OpenMP loop scheduling are displayed at runtime by `PROGRAM_HEADER_Singleton`. When offload parallelism is enabled, the loop scheduling is automatically set to `static` with chunk-size of 1. With multi-threading parallelism, the schedule defaults to `guided` but can be overridden at runtime by setting the environment variable `OMP_SCHEDULE` appropriately.

The example problem `RiemannProblem` in the `Examples` directory under the `Basics` division has been modified to exploit the GPUs using this new functionality. The computational kernels for the problem have been annotated with new OpenMP directives (via the appropriate preprocessor macros) such that they are offloaded to the GPUs when offload parallelism is enabled during compilation. In [5] we demonstrate the weak scaling of this example problem up to 8000 GPUs on the Summit supercomputer at the Oak Ridge Leadership Computing Facility.[4] Figs. 2 and 3 show a visualization of the three-dimensional version of `RiemannProblem` at 1280^3 resolution executed with 1000 GPUs.

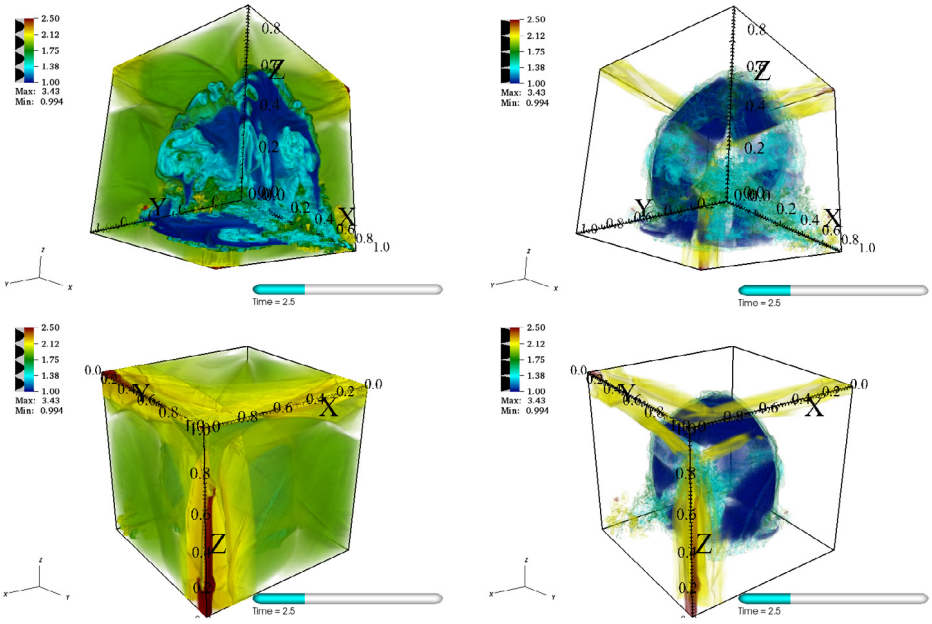

Fig. 2. The polytropic constant (essentially, entropy per baryon) in a three-dimensional version of the `RiemannProblem` example at 1280^3 resolution at time $t = 2.5$. The initial conditions consist of two regions with different pressures and densities separated by a plane of discontinuity intersecting three points: [0.5, 0.0, 0.0], [0.0, 0.75, 0.0], [0.0, 0.0, 1.0] in a computational domain spanning [0, 1] in each dimension. Volume plots from two different perspectives (top and bottom) of with two different transfer functions (left and right) are displayed.

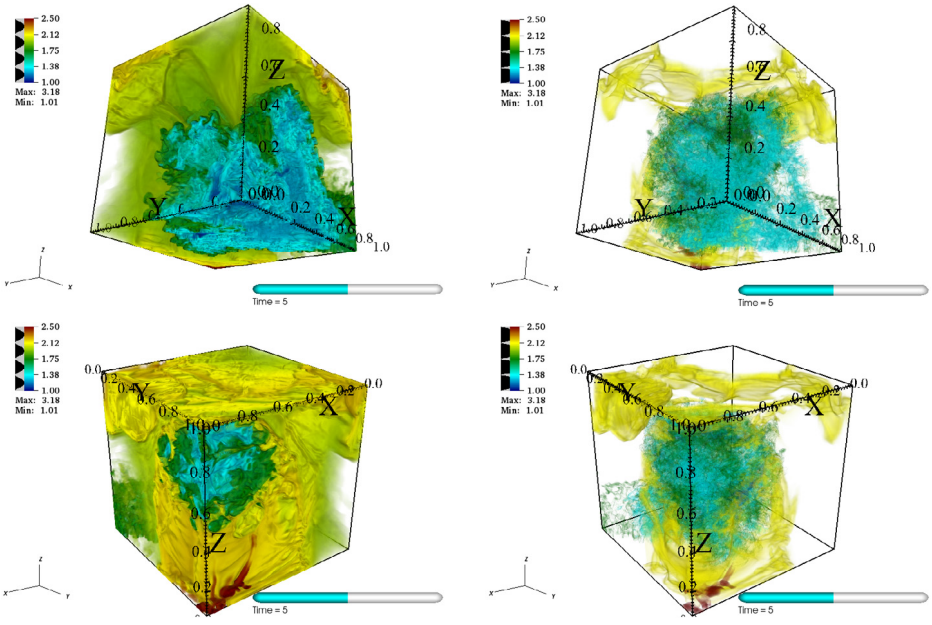

Fig. 3. As in Fig. 2, but at time $t = 5$.

4 https://www.olcf.ornl.gov/olcf-resources/compute-systems/summit/.

Nature of problem: By way of illustrating GenASiS `Basics` functionality, solve example fluid dynamics and molecular dynamics problems.
Solution method: For fluid dynamics examples, finite-volume. For molecular dynamics examples, leapfrog and velocity-Verlet integration.
External routines/libraries: MPI [1] and Silo [2]
Additional comments including restrictions and unusual features: The example problems named above are not ends in themselves, but serve to illustrate our object-oriented approach and the functionality available though GenASiS `Basics`. In addition to these more substantial examples, we provide individual unit test programs for each of the classes comprised by GenASiS `Basics`.

GenASiS `Basics` is available in the CPC Program Library and also at https://github.com/GenASiS.

[1] http://www.mcs.anl.gov/mpi/
[2] https://wci.llnl.gov/simulation/computer-codes/silo
[3] https://en.wikipedia.org/wiki/SI_base_unit
[4] M. Tanabashi et al. (Particle Data Group), Phys. Rev. D 98 (2018) 030001
[5] Budiardja, R.D. and Cardall, C.Y., "Targeting GPUs with OpenMP Directives on Summit: A Simple and Effective Fortran Experience," *submitted for publication Parallel Computing: Systems and Applications*, arXiv:1812.07977 [physics.comp-ph],

Acknowledgments

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Nuclear Physics under contract number DE-AC05-00OR22725 and the National Science Foundation under Grant No. 1535130. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725.