



Center of Excellence for HPC
Astrophysical Applications

Welcome to gPLUTO !

*A. Mignone¹, M. Rossazza¹, S. Truzzi¹, V. Berta¹,
A. Suriano¹, M. Bugli^{1,2}, G. Mattia³*

¹Physics Department, University of Torino

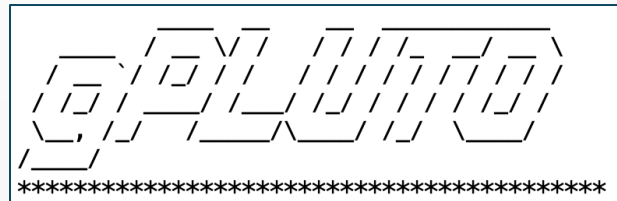
²IAP-CNRS (Paris)

³MPIA (Heidelberg)

Current Status



- The standard version of PLUTO will be maintained although it will no longer receive major upgrades.
- Drastic re-design and re-engineer in order to cope with new exa-scale hardware solution, typically endowed with heterogenous architectural complexity.
- → the PLUTO next line of development:



The *PLUTO* Legacy: *gPLUTO*



- **gPLUTO** is a full rewrite of the PLUTO code targeting exa-scale facilities and new generation hardware.
- **C++**, **OpenACC** (a high-level directive-based programming model developed by NVIDIA) and **OpenMP** chosen as our programming paradigm.
- Started in 2020 → full code rewrite + **NVIDIA** support [except for a few kernels, e.g. initialization, I/O, user interface, etc...]; Aims:
 - Exhaustive porting of the code to GPU;
 - Targeting exa-scale facilities with both CPU and GPUs;
 - Complete code revision (PLUTO is 18 years old !);

The SPACE CoE



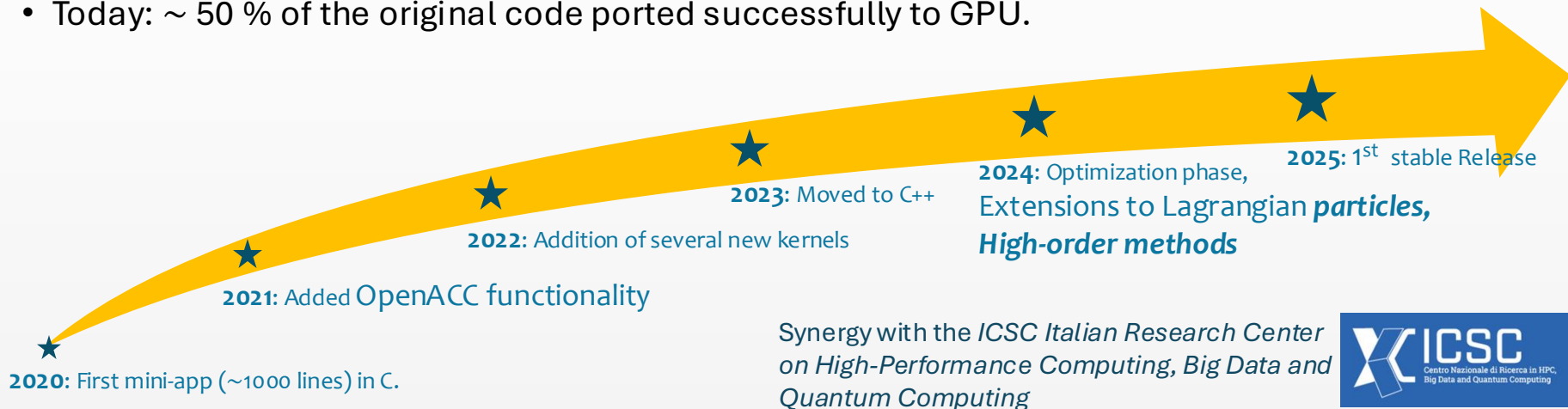
- gPLUTO is developed within the **SPACE** (**S**calable **P**arallel **A**strophysical **C**odes for **E**xascale): a Center of Excellence funded by the European HPC Joint Undertaking (JU).
- The CoE's primary objective is to prepare 7 of the existing state-of-the-art European HPC astrophysics and cosmology codes for the transition to exa-scale on Euro HPC facilities.
- **SPACE** involves co-design activities bringing together scientists, code developers, HPC experts, HW manufacturers and SW developers.



Activities Timeline



- Code rewritten from scratch (!) in 2020: with simple HD module (miniapp, ~1000 lines);
- Incrementally added modules & kernels;
- Switched to C++ to exploit more versatile construct (e.g., templates, classes, vectors);
- Today: ~ 50 % of the original code ported successfully to GPU.






Code Download

- Both PLUTO and gPLUTO may be freely downloaded from the code website;

gPLUTO (v. 0.85)

Standard PLUTO (v. 4.4)

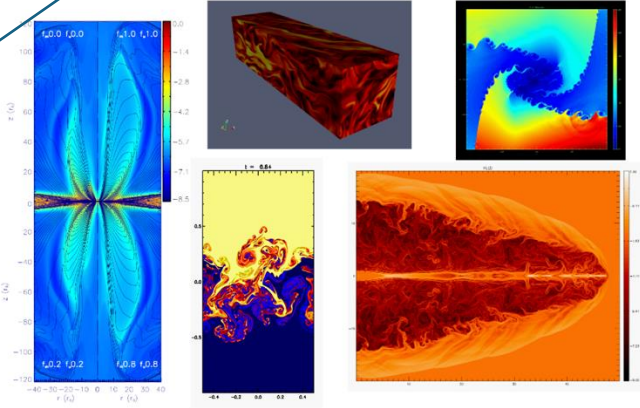
- Before downloading, check if your simulation requisites are satisfied !



The PLUTO Code for Astrophysical GasDynamics

[Home](#) [gPLUTO](#) [Gallery](#) [Documentation](#) [Downloads](#) [PLUTO Symposium 2024](#)

The PLUTO Code



Code Version: 4.4-patch3 (September 2024)

PLUTO is a freely-distributed software for the numerical solution of mixed hyperbolic/parabolic systems of partial differential equations (conservation laws) targeting high Mach number flows in astrophysical plasma dynamics. The code is designed with a modular and flexible structure whereby different numerical algorithms can be separately combined to solve systems of conservation laws using the finite volume or finite difference approach based on Godunov-type schemes.

The CPU version of the code is written in the C programming language while the AMR interface also requires also C++.

PLUTO is a highly portable software and can run from a single workstation up to several thousands processors using the Message Passing Interface (MPI) to achieve highly scalable parallel performance.



PLUTO vs gPLUTO: Physics Modularity



		PLUTO	gPLUTO
Basics Physics	<ul style="list-style-type: none"> [HD] Euler Equations [MHD] Magnetohydrodynamics [RHD] Relativistic Hydrodynamics [RMHD] Relativistic (ideal) MHD [ResRMHD] Relativistic non-ideal MHD 	Yes Yes Yes Yes Yes	Yes Yes Yes Yes Yes
Dissipation Physics	<ul style="list-style-type: none"> Viscosity/Resistivity Thermal Conduction Ambipolar Diffusion Hall MHD 	Yes Yes No Yes	No (planned) No (planned) No (planned) No (planned)
EoS / Thermodynamics	<ul style="list-style-type: none"> Ideal / Isothermal Eos Synge Gas Cooling 	Yes Yes Yes	Yes Yes No (planned)
Cooling	<ul style="list-style-type: none"> Tabulated / Power Law SNEq MINEq 	Yes Yes Yes	Partially No (planned) No
Particles*	<ul style="list-style-type: none"> Cosmic Rays Dust Lagrangian particles 	Yes Yes Yes	Ongoing (planned) No (planned) Yes

PLUTO vs gPLUTO: Algorithm Modularity



		PLUTO	gPLUTO
Time Stepping	<ul style="list-style-type: none"> Hancock / Characteristic Tracing RK (2, 3, 4) 	Yes Yes	No (not planned) Yes
Reconstruction	<ul style="list-style-type: none"> Linear Parabolic WENO3 / WENO5 Monotonicity Preserving (5th) 	Yes Yes Yes Yes	Yes Yes Yes Yes
Riemann Solver	<ul style="list-style-type: none"> Two Shock Roe TVDLF / HLL / HLLC / HLLD GFORCE 	Yes Yes Yes Yes	Yes Yes Yes Yes
High order methods	4th-order finite volume method	No	Yes
$\nabla \cdot \mathbf{B} = 0$	<ul style="list-style-type: none"> 8 Wave Generalized Lagrangian Multiplier (GLM) Constrained Transport 	Yes Yes Yes	Yes Yes Yes
Poisson Solver		No	Yes
FARGO	(Fast Advection Scheme for Rotating Objects)	Yes	Yes
ShearingBox	(Local model of differentially rotating disk)	Yes	No (planned)

PLUTO vs gPLUTO: Grid, I/O, ...



		PLUTO	gPLUTO
Geometry	<ul style="list-style-type: none">• Cartesian / Cylindrical / Spherical	Yes	Yes
Non-Uniform Grid	<ul style="list-style-type: none">• Fixed transformation• Mapped grids	Yes No	No Yes
Adaptive Mesh Refinement		Yes (Chombo lib)	Dev*
Output File Format (full parallel I/O)	<ul style="list-style-type: none">• VTK• Binary• HDF5	Yes Yes Yes	Yes Yes Yes
Input	<ul style="list-style-type: none">• Restart (Binary / HDF5)• Externally produced data files	Yes Yes	Yes Yes
Parallel Communication (MPI / NCCL)	<ul style="list-style-type: none">• Synchronous• Asynchronous (more performant)	Yes No	Yes Yes

* Independent development outside SPACE

Structural Changes



- While **gPLUTO** retains a very similar structure to its predecessor, some important differences:

	PLUTO	gPLUTO
Language	C	C++ / OpenACC / OpenMP
Version	4.4 – patch 3	0.85 (beta)
Runs on CPU	<u>YES</u>	<u>YES</u>
Runs on GPU	<u>NO</u>	<u>YES</u>
Parallel	<u>YES</u>	<u>YES</u>
Adaptive Mesh Refinement	<u>YES</u>	<u>NO</u> (under development)
Particle Support	<u>YES</u>	<u>NO</u> (under development)
Download	pluto-xx.tar.gz archive file	GitLab Repository

gPLUTO: how to configure a problem



- In developing **gPLUTO**, we try to guarantee “almost” backward compatibility;
- Problem configuration still defined in terms of the 3\ usual files,

- **init.cpp**: defines initial conditions, user-def b.c. and analysis
- **definitions.hpp**: problem header file
- **pluto.ini**: runtime configuration file

- Python script used to create the makefile and/or change header file options:

```
> export PLUTO_DIR=$HOME/gPLUTO    # sets your environment variable
> python3 $PLUTO_DIR/setup.py       # run the python script to create definitions.hpp and makefile
> make                             # compile the code
> ./pluto                          # run the code
```

Initial problem configuration: main changes

	PLUTO	gPLUTO
Initial conditions	init.c: <pre>void Init(double *v, double x, double y, double z)</pre>	init.cpp <pre>void Init(double *v, double x, double y, double z, RunConfig *run_config)</pre>
Problem header file	definitions.h:	definitions.hpp <i>→ Remains essentially unchanged (except for some pre-defined macros)</i>
Runtime initialization	pluto.ini [Grid] <pre>X1-grid 1 -1.0 128 u 1.0 X2-grid 1 -0.5 64 u 0.5 X3-grid 1 0.0 1 u 1.0</pre>	pluto.ini <i>→ Single uniform patch (no stretched grids*).</i> [Grid] <pre>X1-grid -1.0 128 1.0 X2-grid -0.5 64 0.5 X3-grid 0.0 1 1.0</pre>

3D Array notation and indexes



	PLUTO	gPLUTO
Data allocation type	Pointer-to-pointer, dynamic memory allocation: <code>static double ****Uc;</code> <code>if (Uc == NULL) Uc = ARRAY_4D(...);</code>	C++ template class with 1D strided array, dynamically allocated. <code>static Ary4D Uc;</code> <code>if (!Uc.isallocated()) Uc.create(...);</code>
Data ordering	k (slowest), j, i (fastest)	i,j,k (internally handled by class)
Cons. variable array	<code>data->Uc[k][j][i][nv]</code>	<code>data->Uc(i,j,k,nv)</code>
Prim. variable array	<code>data->Vc[nv][k][j][i]</code>	<code>data->Vc(i,j,k,nv)</code>
Staggered fields	<code>data->Vs[nv][k][j][i+1]</code>	<code>data->Vs[nv](i+1,j,k)</code>

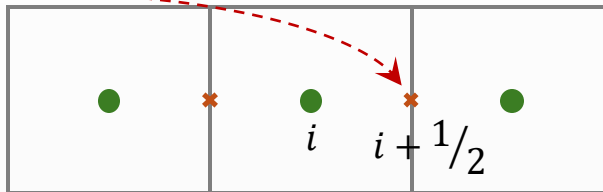
gPLUTO index
ordering is now
i,j,k

Half-integer notation

- During the re-design process, we establish a new convention to access face- (or edge-) centred quantities (e.g. fluxes, staggered fields):

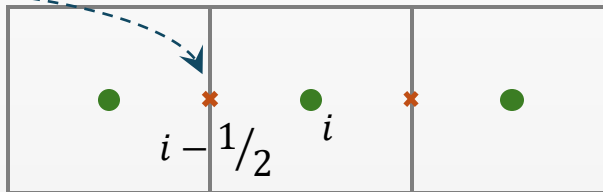
PLUTO: $Vs[BX1s][k][j][i] = B_{x,i+\frac{1}{2},j,k}$

Here $[i][j][k]$ refers to the right interface.



gPLUTO: $Vs[BX1s](i,j,k) = B_{x,i-\frac{1}{2},j,k}$

Now (i,j,k) refers to the left interface.



Other general considerations



- Drastically reduced usage of global variables;
- Extensive use of the BOX_LOOP() macro(s);
- Completely different errors and warnings diagnostic (*);
- Approach to curvilinear grids no longer based on non-equidistant geometric centroid;

The gPLUTO Development Team



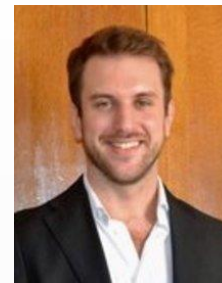
M. Rossazza
(OpenACC Jedi)



G. Mattia
(PyPLUTO, ResRMHD)



M. Bugli
(GR, GitLab expert)



D.M. Fuksman
(Radiation, AMR)



V. Berta
(High Order, ResRMHD, Particles)



S. Truzzi
(Code porting / OpenMP / HPC access)



A. Suriano
(LP Particles)

THE END
