Project Identification	
Convocatoria	Ciencia Básica y de Frontera 2026
Título (EN)	JURHIATA: Next-generation GPU-accelerated moving-mesh code for Astrophysical Magnetohydrodynamics
Clave	846
Responsable Técnico/a	Manuel Zamora-Avilés

JURHIATA: Next-generation GPU-accelerated moving-mesh code for Astrophysical Magnetohydrodynamics

JURHIATA Development Team

March 31, 2026

ID	Participants	Institution	Expertise / Role
PI	Dr. Manuel Zamora-Avilés	SECIHTI / INAOE	MHD and numerical simulations (project lead in all stages).
coPI	Dr. Javier Ballesteros-Paredes	IRyA (UNAM)	MHD and numerical simulations; co-PI and physics leadership.
P1	Dr. Jose Franco	IA (UNAM)	Theoretical astrophysics and stellar-feedback modeling.
P2	Dr. Gilberto C. Gómez-Reyes	IRyA (UNAM)	Computational astrophysics and fluid dynamics (involved in all stages).
P3	Dr. Enrique Vázquez-Semadeni	IRyA (UNAM)	MHD, interstellar-medium physics, molecular clouds, and star formation.
P4	Dra. Adriana Gazol	IA (UNAM)	(Magneto)hydrodynamic models of the interstellar medium.
P5	Dra. Griselda Arroyo Chavez	University of Arizona	MHD simulations, star formation, scientific visualization, and SPH methods (all stages).
P6	Dr. Raúl Naranjo-Romero	INAOE	Scientific computing, cluster operations, and science outreach.
P7	Dr. Abraham Luna Castellanos	INAOE	Millimeter-band astronomy, observational polarimetry, and astronomical instrumentation.
P8	Dr. Ruben Guerrero Gamboa	IRyA (UNAM)	MHD and adaptive-mesh codes.
P9	Dra. Vianey Edaly Camacho Pérez	National Taiwan Normal University	Numerical simulations and star formation.
P10	M.C. Josué Gerardo López Castillo	INAOE	Numerical-simulation and MHD analysis (student researcher; all stages).

Continued on next page

ID	Participants	Institution	Expertise / Role
P11	M.C. Fabián Alberto Quesada Zúñiga	INAOE	Numerical-simulation and MHD analysis (student researcher; all stages).
P12	M.C. Luis Andrés Hernández Cruz	INAOE	Numerical-simulation analysis, MHD, and synthetic observations (student researcher; all stages).
P13	M.C. Alejandro Edmundo Aguilar Torrez	INAOE	Numerical simulations, data analysis, and stellar clusters (student researcher; all stages).
P14	M.C. Karla Alejandra Gutierrez Davila	IRyA (UNAM)	Simulation analysis and galactic evolution.

Acronyms: SECIHTI: Secretaría de Ciencia, Humanidades, Tecnología e Innovación; INAOE: Instituto Nacional de Astrofísica, Óptica y Electrónica; IA: Instituto de Astronomía; IRyA: Instituto de Radioastronomía y Astrofísica; UNAM: Universidad Nacional Autónoma de México.

Abstract

JURHIATA[1] is a new parallel finite-volume code for astrophysical fluid dynamics under development at INAOE and UNAM. By employing a moving Voronoi mesh, it preserves the shock-capturing accuracy of Godunov-type schemes while providing quasi-Lagrangian adaptivity, which significantly reduces advection errors in highly dynamical flows. The current baseline is fully operational on CPUs with hybrid OpenMP/MPI parallelization. The project is organized into three stages: **Stage 1 (2026)** focuses on GPU parallelization of the hydrodynamic core; **Stage 2 (2027)** implements ideal MHD and the physical modules for star formation (self-gravity, radiative processes, and stellar feedback); and **Stage 3 (2028)** concludes with scientific exploitation through full-physics simulations of a Milky Way-like galaxy. The software architecture is explicitly designed for GPU-based high-performance clusters, including national infrastructure such as the *Coatlicue*[2] supercomputer, aligning the project with Mexican Government strategic initiatives. A central objective is to achieve high performance through systematic GPU-oriented optimization of the numerical algorithms. JURHIATA is also designed as an open community resource: a high-performance astrophysical simulation code that integrates modern numerical methods, advanced physical modules, and scalable parallel performance, providing an efficient, modular and extensible code for the numerical study of complex astrophysical systems.

This proposal is organized as follows. Section 1 motivates the scientific and computational need for JURHIATA. Section 2 documents the current HD baseline and completed OpenMP+MPI parallel implementation, including numerical method details and validation tests. Section 3 presents the three-stage project plan (GPU parallelization, additional physics modules, and scientific exploitation). Section 4 summarizes execution governance, team responsibilities, budget/computing-resource justification, open-science strategy, and broader impact.

For dissemination and community access, project updates and public-facing information are available at https://mixtli.inaoep.mx/mixtli/jurhiata.html.

Contents

[1] JURHIATA, whose name means *Sun* in the Purépecha language, takes its name from the Meseta Purépecha region of Michoacán (Mexico), the place of origin of the project's PI.

[2] Official federal announcement page: https://www.gob.mx/

1 Introduction

1.1 Numerical methods in astrophysical fluid dynamics

Numerical simulations have become an essential tool for studying complex phenomena in astrophysical fluid dynamics, including turbulence in the interstellar medium, star formation, stellar feedback, galaxy formation and evolution, etc. These systems involve highly nonlinear interactions between shocks, turbulence, gravity, and magnetic fields across a wide range of spatial and temporal scales. Accurately modeling such processes requires numerical methods capable of capturing discontinuities, preserving conservation laws, and maintaining stability and accuracy in highly dynamical flows.

Two main numerical paradigms have traditionally been used in astrophysical hydrodynamics and magnetohydrodynamics (MHD): Eulerian grid-based methods and Lagrangian particle-based methods. Grid-based codes solve the fluid equations on fixed or adaptively refined meshes using finite-volume Godunov schemes with approximate Riemann solvers. Prominent examples include `FLASH` (Fryxell et al., 2000), `RAMSES` (Teyssier, 2002), and `ATHENA` (Stone et al., 2008). These approaches provide robust shock-capturing capabilities and have been widely used in a broad range of astrophysical applications.

However, purely Eulerian methods can suffer from significant advection errors when large bulk velocities are present relative to the computational grid. In contrast, Lagrangian approaches such as Smoothed Particle Hydrodynamics (SPH; Monaghan, 1992), implemented in widely used codes such as `GADGET` (Springel, 2005) and `PHANTOM` (Price, 2012), represent fluids through particles that *move* with the flow. This formulation naturally eliminates grid advection errors and provides adaptive spatial resolution. Nevertheless, traditional SPH implementations have been shown to encounter difficulties in accurately capturing fluid instabilities, multiphase mixing, and contact discontinuities.

These limitations have motivated the development of hybrid numerical methods that combine the advantages of Eulerian and Lagrangian approaches. A complementary middle-ground alternative to both fixed-grid Eulerian schemes and classical SPH, and therefore an alternative to the moving-mesh strategy discussed next, is the class of meshless Godunov methods introduced by Hopkins (2015). In these methods, inter-element fluxes are computed through Riemann problems between moving resolution elements without a fixed mesh topology. These meshless finite-mass/finite-volume schemes combine strong conservation and shock-capturing properties with Lagrangian adaptivity.

1.2 Moving-mesh methods and existing codes

Moving-mesh methods represent one of the earliest and most successful hybrid approaches for solving the MHD equations in astrophysical simulations. In these schemes, the computational domain is discretized

using a dynamic Voronoi tessellation whose generating points move with the local fluid velocity. The mesh therefore evolves continuously in time, providing a quasi-Lagrangian description of the flow while preserving the geometric flexibility of Eulerian finite-volume methods.

The fluid equations are solved using Godunov-type schemes in which local Riemann problems are evaluated at cell interfaces. This formulation enables accurate shock capturing while reducing advection errors and improving Galilean invariance. Because the mesh follows the bulk motion of the gas, numerical diffusion associated with advection across fixed grids is significantly reduced, while the finite-volume framework ensures strict conservation of mass, momentum, and energy.

These properties make moving-mesh schemes particularly well suited for astrophysical applications involving multiphase gas dynamics, self-gravitating and turbulent flows, and magnetized plasmas.

The most prominent implementation of the moving-mesh technique in astrophysics is the AREPO code of Springel (2010, hereafter S10). AREPO introduced a finite-volume Godunov scheme on a dynamically evolving Voronoi mesh and demonstrated that moving-mesh methods can achieve both high accuracy and excellent performance in complex astrophysical simulations. Since its introduction, AREPO has been widely used in large cosmological simulations, including major galaxy formation projects such as Illustris and IllustrisTNG (Vogelsberger et al., 2014; Pillepich et al., 2018). At the same time, the public AREPO release provides a reduced subset of the full internal physics stack, with available subgrid recipes primarily configured for cosmological workflows rather than as a broad star-formation-focused physics framework (Weinberger et al., 2020).

Beyond AREPO, there are other moving-mesh and Voronoi-based developments. RICH (Yalinewich et al., 2015) and SHADOWFAX (Vandenbroucke & De Rijcke, 2016) are publicly available moving-mesh codes aimed at studying astrophysical gas dynamics. However, these codes do not include the physics framework for star-formation studies. The MANGA solver in the CHANGA ecosystem (Chang et al., 2017) is methodologically close to AREPO, but it is not publicly available. In contrast, TESS (Duffell & MacFadyen, 2011) targets relativistic hydrodynamics on a moving Voronoi mesh, demonstrating that moving-mesh techniques can be extended to relativistic regimes.

Despite their success, most (if not all) existing moving-mesh implementations are designed primarily for CPU-based high-performance computing systems using distributed-memory parallelism. These limitations reduce the accessibility and adaptability of moving-mesh techniques for a broader range of astrophysical applications.

1.3 High-performance computing and the GPU gap

Over the past decade, the architecture of high-performance computing systems has undergone a major transformation. Modern supercomputers increasingly rely on heterogeneous architectures that combine traditional CPUs with graphics processing units (GPUs) to achieve high computational throughput and improved energy efficiency.[3]

Exploiting these architectures efficiently requires numerical algorithms and data structures specifically designed for massive parallelism and optimized memory access patterns. While GPU acceleration has been successfully adopted in several astrophysical grid-based codes, including GAMER-2 (Schive et al., 2018), FARGO3D (Benítez-Llambay & Masset, 2016), and ATHENAK (Stone et al., 2024), moving-mesh methods present additional algorithmic challenges. These include the dynamic construction of Voronoi tessellations, the continuous evolution of mesh topology, and the irregular communication patterns associated with distributed mesh updates.

Despite the growing importance of GPU-accelerated computing in modern astrophysical simulations, there is currently no publicly available moving-mesh code designed from the outset to exploit GPU-based high-performance computing architectures. This represents a significant gap in the available computational tools for astrophysical fluid dynamics.

1.4 Motivation for the JURHIATA code

The JURHIATA project aims to address this gap through the development of a new moving-mesh computational framework for astrophysical magnetohydrodynamics specifically designed for GPU-accelerated systems.

The code is based on a finite-volume Godunov scheme implemented on a moving Voronoi mesh and incorporates modern Riemann solvers for magnetohydrodynamic flows. In the current implementation, the mesh construction relies on the VORO++ library (Rycroft, 2009), while the numerical infrastructure is

[3]The most powerful supercomputers have hybrid architecture (CPU + GPU), see https://www.top500.org/.

designed around a Structure-of-Arrays (SoA; see 2.5) memory layout that enables efficient memory access patterns and coalesced memory transactions on GPU architectures.

By combining the numerical advantages of moving-mesh methods with a software architecture tailored for heterogeneous high-performance computing platforms, JURHIATA aims to provide a next-generation simulation tool capable of performing high-resolution studies of astrophysical fluids dynamics on modern GPU-based supercomputing systems. Moreover, the Mexican government has recently announced the construction of a new GPU-based national supercomputer, Coatlicue, which is expected to become the most powerful public computing system in Latin America. The system is projected to reach a performance of about 314 petaflops and to integrate 14,480 GPUs, providing a major expansion of the country's high-performance computing capabilities for scientific research and large-scale numerical simulations. The facility is expected to become operational within the next two years and will constitute a key national infrastructure for computational science.[4]

2 Numerical methods and current code implementation

This section describes the numerical methods and software architecture currently implemented in the JURHIATA code. The present version of the code focuses on solving the equations of compressible hydrodynamics using a finite-volume method on a moving Voronoi mesh. The numerical infrastructure is designed to enable efficient execution on distributed-memory high-performance computing systems while maintaining a modular structure that facilitates future extensions.

2.1 Hydrodynamic equations

JURHIATA solves the Euler equations of compressible hydrodynamics in conservative form:

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{U}) = 0, \tag{1}$$

where the vector of conserved variables is

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho \mathbf{v} \\ \rho e \end{pmatrix}, \tag{2}$$

and the corresponding flux tensor is

$$\mathbf{F}(\mathbf{U}) = \begin{pmatrix} \rho \mathbf{v} \\ \rho \mathbf{v}\mathbf{v} + P\mathbf{I} \\ (\rho e + P)\mathbf{v} \end{pmatrix}. \tag{3}$$

Here ρ denotes the gas density, $\mathbf{v}$ the velocity vector, P the gas pressure, and e the total energy per unit mass,

$$e = u + \frac{1}{2}v^2, \tag{4}$$

with $u = P/\rho(\gamma - 1)$ the thermal energy per unit mass, being γ the adiabatic index (typically $\gamma = 1.4$ for diatomic gas or $\gamma = 5/3$ for monatomic gas). The sound speed is given by:

$$c_s = \sqrt{\frac{\gamma P}{\rho}}. \tag{5}$$

For a moving control volume (moving Voronoi cell), the finite-volume form used in S10 is

$$\mathbf{Q}_i = \int_{V_i} \mathbf{U}\, dV, \qquad \frac{d\mathbf{Q}_i}{dt} = -\int_{\partial V_i} \left[\mathbf{F}(\mathbf{U}) - \mathbf{U}\mathbf{w}^T\right] \cdot d\mathbf{n}, \tag{6}$$

where $\mathbf{Q}_i$ is the cell-integrated conserved-state vector, $d\mathbf{n} = \hat{\mathbf{n}}\, dA$ is the outward surface element on ∂V_i, and $\mathbf{w}$ is the local face velocity of the moving mesh.

These equations are discretized using a second-order finite-volume Godunov scheme in which numerical fluxes are computed through approximate Riemann solvers at cell interfaces; Eq. (6) is the compact moving-mesh conservation form used throughout the implementation discussion.

[4] Official announcement web page (https://www.gob.mx).

2.2 Domain discretization: voronoi tessellation

The computational domain is discretized using a Voronoi tessellation defined by a set of mesh-generating points (particles). Each point defines a Voronoi cell containing all spatial locations that are closer to that generator than to any other generator. That is, a Voronoi cell C_i for a point $\mathbf{r}_i$ is defined as:

$$C_i = \{\mathbf{x} \in \mathbb{R}^3 : |\mathbf{x} - \mathbf{r}_i| \leq |\mathbf{x} - \mathbf{r}_j|, \forall j \neq i\}. \tag{7}$$

The Voronoi tessellation used is generated using the VORO++ library (Rycroft, 2009), which efficiently calculates the cell quatities required for the numerical solver.*i)* cell volumes, V_i; *ii)* face areas, A_{ij}; *iii)* face centroids, $\mathbf{f}_{ij}$; *iv)* face normal vectors, $\mathbf{n}_{ij}$, (pointing from cell i to cell j); and *v)* cell center of mass $\mathbf{s}_i$.

Hydrodynamic variables are stored as cell-averaged quantities within each Voronoi cell. Fluxes are computed across the polygonal faces separating neighboring cells by solving local Riemann problems in the rest frame of the moving face S10. This approach significantly reduces advection errors and improves Galilean invariance compared to static grid methods.

2.3 Numerical workflow

JURHIATA implements the MUSCL-Hancock scheme for second-order accuracy in both space and time. As a general reference for finite-volume Godunov methods and Riemann-solver-based shock capturing, we follow Toro (2009). The scheme consists of the following steps.

2.3.1 Timestep calculation (CFL condition)

The timestep is determined by the Courant-Friedrichs-Lewy (CFL) condition to ensure numerical stability. For each cell, we compute a characteristic length,

$$h_i = \frac{V_i}{\max_j A_{ij}}, \tag{8}$$

where the maximum is taken over all faces of cell i. The local timestep is

$$\Delta t_i = \mathrm{CFL} \times \frac{h_i}{|\mathbf{v}_i| + c_{\max,i}}, \tag{9}$$

where $c_{\max,i}$ is the maximum characteristic speed in cell i.[5] The global timestep is the minimum over all cells,

$$\Delta t = \min_i \Delta t_i. \tag{10}$$

Typical CFL values range from 0.25 to 0.5. The CFL condition ensures that information does not travel more than one cell per timestep (e.g., Toro, 2009).

2.3.2 Gradient computation

For second-order MUSCL-Hancock reconstruction on an unstructured moving mesh, we adopt the MANGA-style strategy (Chang et al., 2017). We first compute accurate cell-centered gradients and then apply geometric monotonicity limiting. Unlike AREPO, gradients are evaluated for the conserved state components $\mathbf{U}$, which is generally more robust in strong-gradient flows (Chang et al., 2017). Primitive-variable reconstruction remains available as an optional mode in smooth-regime studies. Given a cell-centered value ϕ_i and cell center of mass $\mathbf{s}_i$, the local piecewise-linear representation is

$$\phi(\mathbf{r}) = \phi_i + \langle \nabla \phi \rangle_i \cdot (\mathbf{r} - \mathbf{s}_i), \tag{11}$$

where $\langle \nabla \phi \rangle_i$ is the mean gradient in cell i.

To keep linear-order consistency on irregular Voronoi geometry (including the mismatch between mesh-generating points and cell centers), we use the improved gradient construction used in MANGA code (e.g., Steinberg et al., 2016; Chang et al., 2017). In practice, this is implemented through an improved Green–Gauss/least-squares compatible operator that preserves second-order behavior in smooth regions while remaining stable near strong deformation of cells.

[5] For pure hydro, $c_{\max} = c_s$ (sound speed), whereas for MHD, $c_{\max} = c_f$ (fast magnetosonic speed).

2.3.3 Slope limiting

Following the geometric limiter formulation introduced in the moving-mesh framework (S10; Chang et al. (2017)), the gradient is first scaled as

$$\langle \nabla \phi \rangle_i^{\mathrm{S10}} = \alpha_i \langle \nabla \phi \rangle_i, \qquad 0 \le \alpha_i \le 1. \tag{12}$$

For each neighbor j, define

$$\Delta\phi_{ij} = \langle \nabla \phi \rangle_i \cdot (\mathbf{f}_{ij} - \mathbf{s}_i), \tag{13}$$

where $\mathbf{f}_{ij}$ is the face centroid. With local extrema $\phi_i^{\max}$ and $\phi_i^{\min}$ over cell i and neighbors, compute

$$\psi_{ij} = \begin{cases} (\phi_i^{\max} - \phi_i)/\Delta\phi_{ij}, & \Delta\phi_{ij} > 0, \\ (\phi_i^{\min} - \phi_i)/\Delta\phi_{ij}, & \Delta\phi_{ij} < 0, \\ 1, & \Delta\phi_{ij} = 0, \end{cases} \tag{14}$$

and set

$$\alpha_i = \min\left(1, \min_j \psi_{ij}\right). \tag{15}$$

This guarantees interface reconstructions remain within neighbor-defined bounds.

2.3.4 Reconstruction to face centers

For each face, we reconstruct the left and right states at the face centroid using the limited gradients:

$$\phi_L = \phi_i + (\nabla\phi)_i^{\mathrm{lim}} \cdot (\mathbf{c}_{ij} - \mathbf{s}_i) \tag{16}$$

$$\phi_R = \phi_j + (\nabla\phi)_j^{\mathrm{lim}} \cdot (\mathbf{c}_{ij} - \mathbf{s}_j), \tag{17}$$

where $\mathbf{s}_i$ and $\mathbf{s}_j$ are cell centers of mass of cells i and j, respectively. This reconstruction is then mapped to the variables expected by the Riemann solver (primitive or conserved representation, depending on the active mode).

2.3.5 Predictor step

The reconstructed states are evolved forward by $\Delta t/2$:

$$\mathbf{W}_L^* = \mathbf{W}_L + \frac{\Delta t}{2}\left(\frac{\partial \mathbf{W}}{\partial t}\right)_i, \tag{18}$$

where $\mathbf{W}$ denotes the vector of primitive hydrodynamic variables, typically $\mathbf{W} = (\rho, \mathbf{v}, p)^T$. Here, the subscript L refers to the left state reconstructed at the face from cell i, and the superscript $*$ indicates the predicted state at time level $t^n + \Delta t/2$.

2.3.6 Riemann solver: HLLC solver for hydrodynamics

For pure hydrodynamics, the HLLC (Harten–Lax–van Leer with Contact discontinuity restoration) approximate Riemann solver (Toro, 2009; Toro et al., 1994) is used to compute fluxes at each face. The solver estimates wave speeds,

$$S_L = \min(v_{L,n} - c_{s,L}, v_{R,n} - c_{s,R}) \tag{19}$$

$$S_R = \max(v_{L,n} + c_{s,L}, v_{R,n} + c_{s,R}) \tag{20}$$

where S_L and S_R are estimates of the left- and right-going signal velocities that bound the HLLC Riemann fan. Here, the subscripts L and R denote the reconstructed left and right states at the interface, $v_{L,n}$ and $v_{R,n}$ are the normal components of the left and right velocities relative to the face, and $c_{s,L}$ and $c_{s,R}$ are the corresponding left and right sound speeds.

The contact wave speed is:

$$S_* = \frac{p_R - p_L + \rho_L v_{L,n}(S_L - v_{L,n}) - \rho_R v_{R,n}(S_R - v_{R,n})}{\rho_L(S_L - v_{L,n}) - \rho_R(S_R - v_{R,n})} \tag{21}$$

The HLLC flux is then:

$$\mathbf{F}_{\mathrm{HLLC}} = \begin{cases} \mathbf{F}_L & \text{if } 0 \le S_L \\ \mathbf{F}_L^* & \text{if } S_L \le 0 \le S_* \\ \mathbf{F}_R^* & \text{if } S_* \le 0 \le S_R \\ \mathbf{F}_R & \text{if } S_R \le 0 \end{cases} \tag{22}$$

2.3.7 Flux update

The conserved variables are updated using the computed fluxes:

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n - \frac{\Delta t}{V_i} \sum_j A_{ij} \mathbf{F}_{ij} \tag{23}$$

where $\mathbf{U} = (\rho, \rho\mathbf{v}, E)^T$ is the vector of conserved variables.

2.3.8 Mesh motion

Finally, the mesh-generating points are advected:

$$\mathbf{r}_i^{n+1} = \mathbf{r}_i^n + \mathbf{v}_i^{n+1} \Delta t \tag{24}$$

In the moving-mesh approach adopted by JURHIATA, the mesh-generating points move with a velocity that is typically chosen to approximate the local fluid velocity. As a result, the computational mesh evolves dynamically in time, providing a quasi-Lagrangian representation of the flow while preserving the geometric flexibility of Eulerian finite-volume schemes. This completes one timestep of the MUSCL-Hancock scheme.

Figure 1 summarizes the complete second-order MUSCL-Hancock update cycle used in JURHIATA, including hydro/MHD branching, conservative updates, divergence-control path, and timestep loop closure.

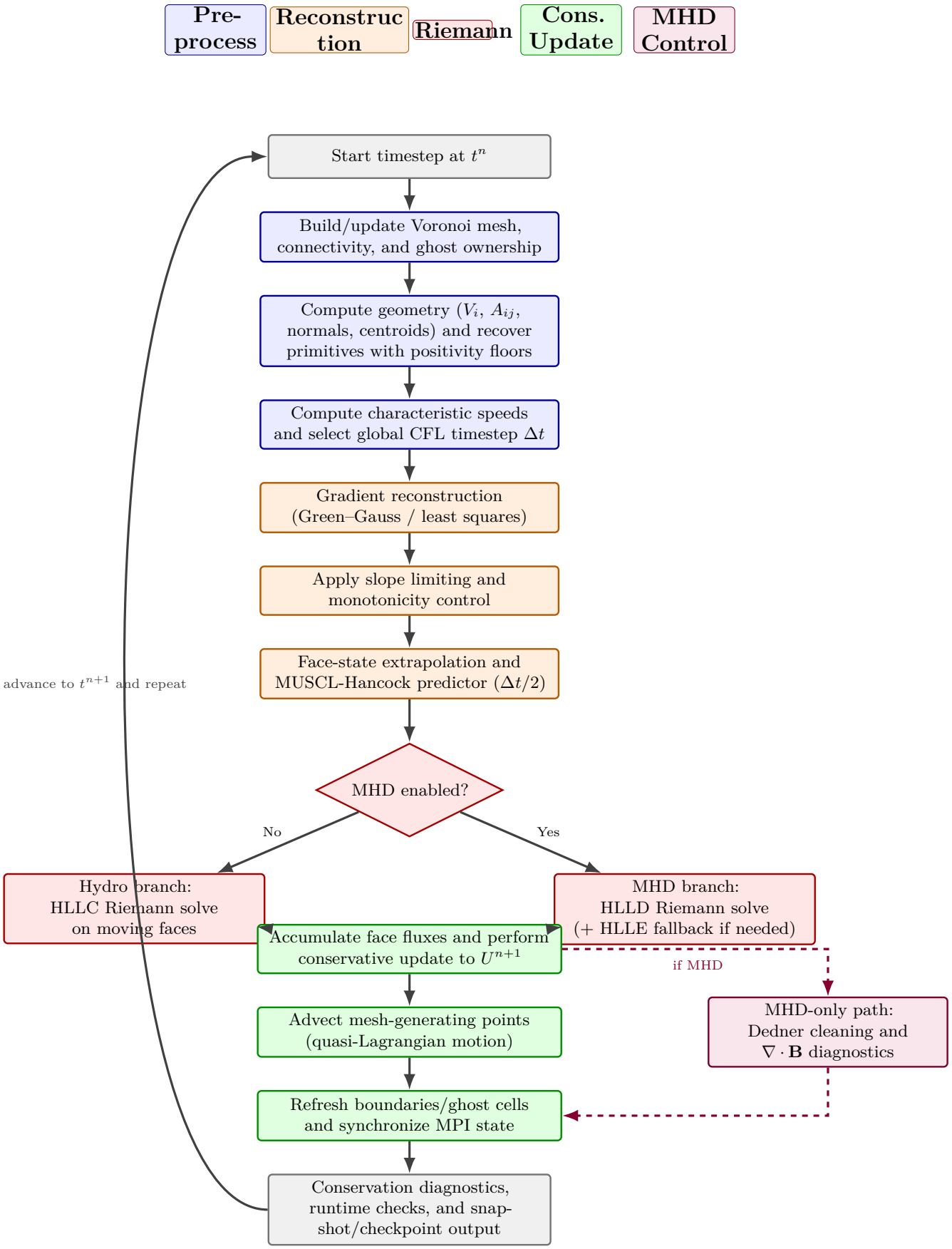

Figure 1: Full MUSCL-Hancock workflow in JURHIATA (second-order in space and time), including hydro/MHD branching, conservative update, optional MHD divergence-control path, and closed timestep loop.

2.4 Boundary conditions

JURHIATA supports three standard boundary-condition classes:

- **Reflecting (solid wall):** the normal velocity component is inverted at the boundary, while tangential components are preserved, enforcing no-penetration walls.
- **Periodic:** opposite domain faces are topologically identified, so fluid states wrap across the domain boundaries; this is natively supported by VORO++ periodic-domain handling.
- **Outflow (transmissive):** zero-gradient boundary states are applied, allowing material to exit the computational domain without artificial reflection.

For MHD runs, boundary states will be applied consistently to magnetic variables and cleaning fields to maintain stable divergence-control behavior at domain edges.

2.5 Data layout and software architecture

The internal data structures of JURHIATA use a Structure-of-Arrays (SoA) layout. In this approach, each field (positions, hydrodynamic variables, and MHD variables when enabled) is stored in contiguous memory blocks instead of embedding all fields in per-cell structs (Array of Structures, AoS, as in AREPO). This improves memory locality, vector efficiency, and GPU readiness. In distributed runs, SoA also allows communication of only the fields required by a given kernel or exchange stage, whereas AoS-oriented layouts often move full structures, which can increase communication volume.

The architecture is modular by construction: numerical kernels, mesh management, physics modules, and parallel communication are separated into explicit components. This separation reduces coupling, improves maintainability, and simplifies staged extension from the current HD baseline toward full multi-physics runs.

Some of the benefits of SoA for MHD include:

- **SIMD[6] vectorization**: contiguous data enables efficient use of AVX/AVX-512[7] instructions for parallel operations on multiple particles.
- **Cache efficiency**: operations on single fields (e.g., computing fast speed from density, pressure, and B-field) access fewer cache lines.
- **MHD-specific advantages**: characteristic speed calculations, gradient computations, and flux calculations benefit significantly from vectorized operations.
- **Reduced memory bandwidth**: only the fields needed for a computation are loaded into cache

When MHD is enabled, the particle state is augmented with magnetic variables, divergence diagnostics, cleaning scalar(s), and derived characteristic quantities (Alfvén, fast, and slow speeds). Separate primitive, conserved, and face-flux containers are used so reconstruction, Riemann solves, and conservative updates operate on explicitly typed MHD states.

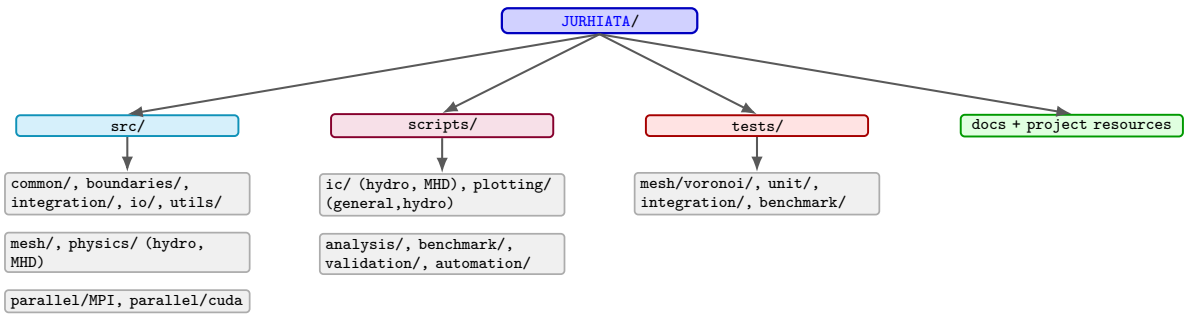

Figure 2: JURHIATA codebase tree with de-cluttered levels for readability. Colors separate core source modules (cyan), tooling (purple), validation (red), and documentation/project resources (green).

[6] SIMD stands for *Single Instruction, Multiple Data*. It refers to vector-style parallelism in which one instruction operates simultaneously on multiple data values, which is important for modern CPU and GPU performance.

[7] AVX (*Advanced Vector Extensions*) and AVX-512 are CPU SIMD vector instruction sets that allow one instruction to process multiple data elements simultaneously.

2.6 Repository structure tree

Figure 2 provides a curated and visually structured map of the JURHIATA codebase, emphasizing the core solver layers, MHD modules, parallel engine, tooling scripts, and validation assets used in this proposal.

To make the internal organization of the source tree more explicit, Figure 2 expands the `src/` subtree into the shared data/configuration layer, tessellation infrastructure, physics kernels, MPI/distributed execution layer, and runtime-support modules that structure the current baseline.

2.7 Validation tests

We validate the HD solver of JURHIATA with two canonical hydrodynamic benchmarks, the Sod shock tube and the Sedov-Taylor blast wave. In both cases, the resulting profiles reproduce the expected physical behavior, and the current baseline already has quantitative diagnostics that motivate the staged acceptance gates used in this proposal.

The end-to-end workflow used for these simulations is already operational: initial-condition generation is handled using Python scripts that construct random particle distributions and simulation-ready HDF5 inputs, followed by automated execution and post-processing.

The validation strategy is explicitly structured around detailed physics tests with analytic solutions whenever available.Beyond Sod and Sedov problems, the following tests are planned during the development of the JURHIATA project:

- **Hydrodynamics (HD):** Gresho vortex, Kelvin–Helmholtz instability, Rayleigh–Taylor instability, Noh strong-shock problem, interacting blast waves, double Mach reflection, and advection of contact discontinuities at high bulk velocity (Galilean-stress test).
- **Magnetohydrodynamics (MHD):** Brio–Wu shock tube, Orszag–Tang vortex, circularly polarized Alfvén wave convergence, MHD rotor, magnetic field-loop advection, MHD blast wave, current-sheet/reconnection setups, and driven-decaying MHD turbulence.
- **Self-gravity / star-formation modules:** Jeans collapse and Evrard collapse (gravity accuracy and energy conservation), stellar particle creation/accretion convergence tests, and controlled feedback-injection tests (blast radius/momentum budget recovery).

These physical tests are designed to jointly assess shock capturing, contact preservation, instability growth, magnetic divergence control, and robustness under strong advection, which are the core performance dimensions required by the scientific goals of this project.

2.7.1 Sod shock tube

Although Sod and related Riemann problems are conceptually 1D benchmarks, our validation runs are executed in 3D domains (quasi-1D setups) because VORO++ is used in 3D mode and the production solver itself is fully three-dimensional.

The Sod setup is initialized in a quasi-1D rectangular domain with transverse thickness:

$$x \in [0, 1], \quad y, z \in [0, 0.1], \quad \gamma = 1.4.$$

A discontinuity is imposed at $x = 0.5$:

$$(\rho, p, v_x)_L = (1.0, 1.0, 0), \qquad (\rho, p, v_x)_R = (0.125, 0.1, 0).$$

The ICs are built with 10k particles with random positions to better stress reconstruction/limiting behavior. The run uses CFL $= 0.3$ and evolves to $t_{\max} = 0.2$. JURHIATA captures the rarefaction wave, contact discontinuity, and shock with the expected ordering and propagation. Even in this low-resolution case, the simulation provides sharper gradients and reduced numerical broadening. Therefore, JURHIATA passes the Sod test (see Figure 3).

2.7.2 Sedov-Taylor blast

The Sedov test is initialized in a cubic domain:

$$x, y, z \in [0, 1], \quad \gamma = 1.4.$$

The ambient medium is uniform:

$$\rho_0 = 1, \qquad p_0 = 10^{-6}, \qquad \mathbf{v}_0 = \mathbf{0}.$$

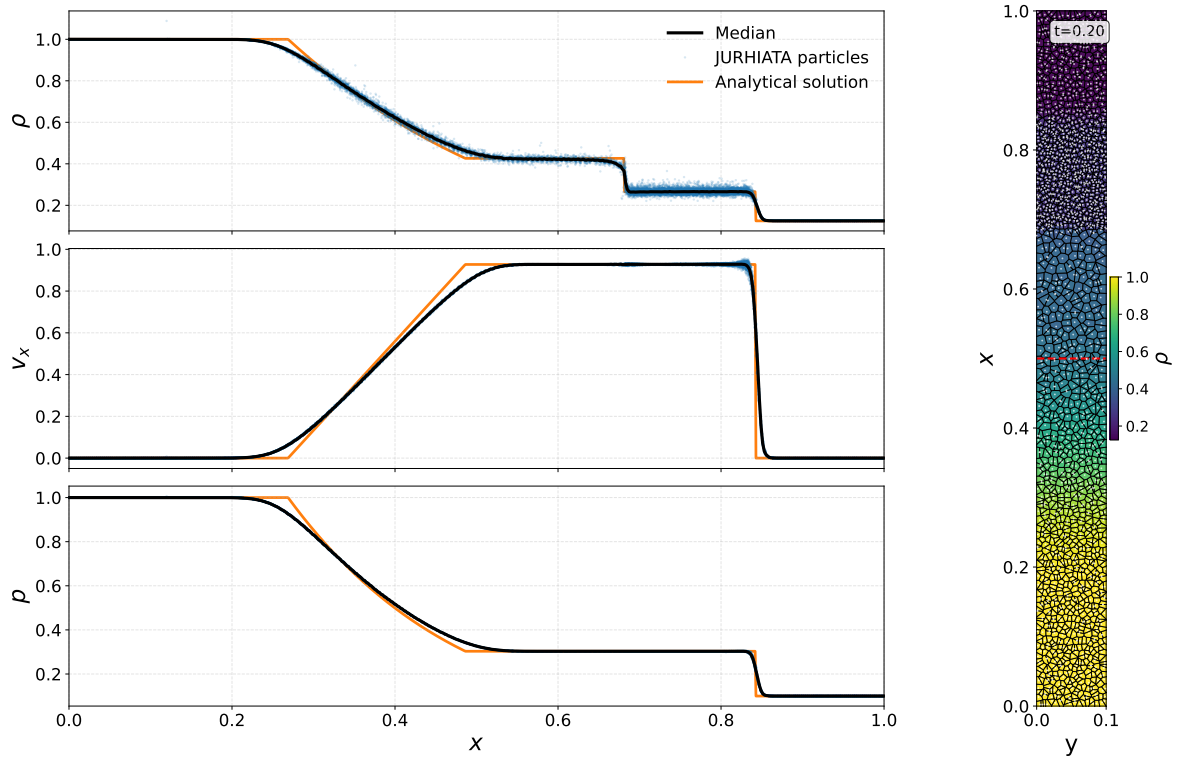

Figure 3: Sod shock-tube validation panel used in this proposal. Left: x-profiles of density, v_x, and pressure, where blue points show cell-by-cell simulation values. Right: density map in the xy plane with mesh-generating points (white markers) and Voronoi-cell structure.

Thermal energy is injected in a central spherical region (effective blast radius $\simeq 0.04$), producing a high-pressure sphere. The IC file contains 100k particles with intentionally enhanced central sampling to better resolve the early shock launch and post-shock gradients.

The run uses CFL $= 0.3$ and $t_{\max} = 0.12$. The solution exhibits the expected expanding spherical blast structure and outward-propagating density shell. The profile behavior shown in Figure 4 is consistent with the expected Sedov morphology at early time ($t = 0.1$). Thus, JURHIATA passes the Sedov validation test (Figure 4).

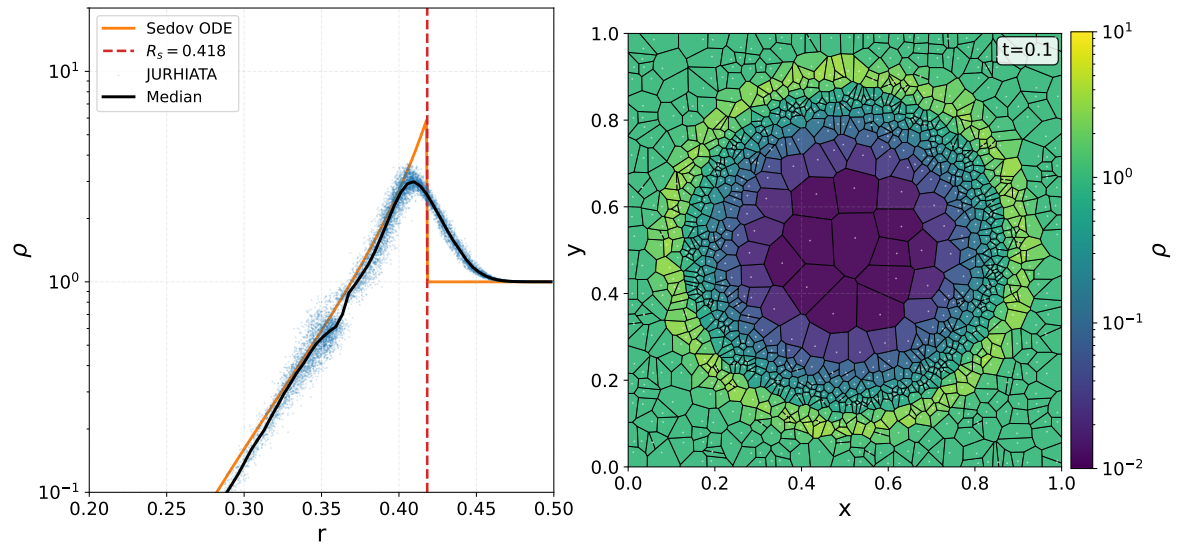

Figure 4: Sedov validation panel (100k particles, refined central sampling, $t = 0.1$). Left: radial density profile with blue points showing cell-by-cell simulation values. Right: density slice in the xy plane with mesh-generating points (white markers) and Voronoi-cell structure.

2.8 Current parallelization status: OpenMP and MPI hybrid implementation

The present codebase of JURHIATA corresponds to a CPU implementation of the moving-mesh hydrodynamic solver. At this stage, the code already includes a complete shared-memory and distributed-memory parallel stack:

- **OpenMP parallelization is complete**, providing threaded execution of the main hydrodynamic update path and supporting kernels in shared-memory environments.
- **MPI parallelization is complete**, including domain decomposition and multi-rank communication integrated into the simulation loop.
- **Hybrid CPU execution is available**, using OpenMP within each computing node and MPI across nodes (in testing phase).

Within the distributed-memory layer, JURHIATA uses the Message Passing Interface (MPI) to decompose the computational domain across processes according to the spatial distribution of Voronoi cells. The decomposition is constructed using a **Peano–Hilbert space-filling curve**, which maps the three-dimensional spatial distribution of mesh-generating points onto a one-dimensional ordering while preserving spatial locality. This strategy enables efficient load balancing and minimizes communication between neighboring domains.

Each MPI rank evolves a subset of the mesh-generating points and their associated Voronoi cells. Communication between neighboring subdomains is required to exchange cell information and to compute fluxes consistently across domain boundaries. As the mesh evolves dynamically, the domain decomposition can be updated to preserve load balance and maintain approximately uniform computational workloads across MPI ranks.

This parallel baseline is scientifically important for two reasons. First, it provides a controlled reference implementation against which future GPU kernels can be validated. Second, it enables numerical reproducibility and regression testing throughout the accelerator transition, ensuring that performance optimization does not alter the physical behavior of the solver.

Figure 5 summarizes the present decomposition and scaling behavior. The top panel illustrates the Peano–Hilbert domain decomposition for four MPI ranks of the computational box (with 10k cells), while the bottom panel shows the measured speedup as a function of the number of MPI ranks. The observed MPI performance is reasonable, the normalized speedup, relative to ideal scaling, remains in the 70–100% range across the tested rank counts.

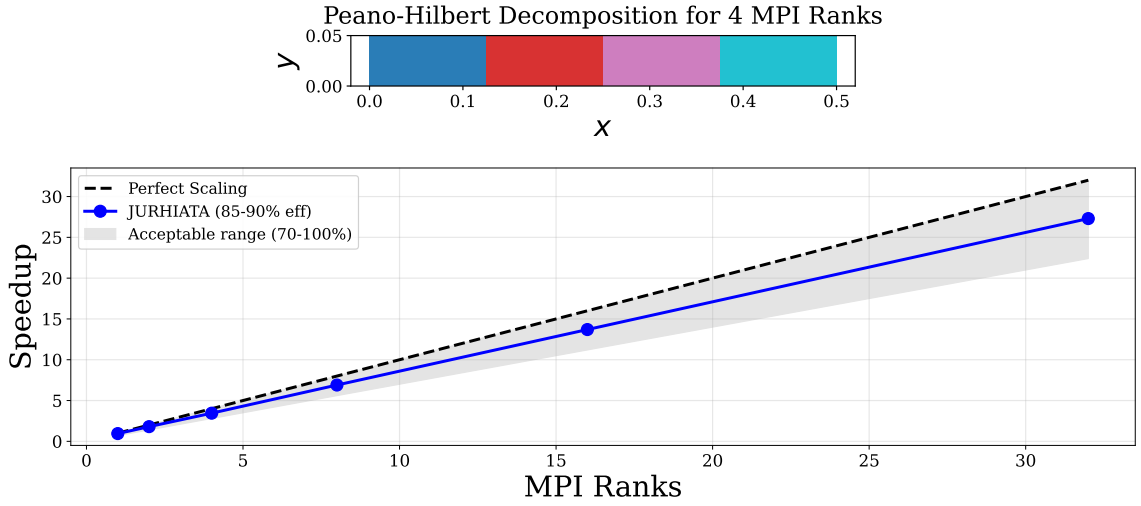

Figure 5: MPI performance summary for JURHIATA. Top panel: Peano–Hilbert domain decomposition of the computational box ($L_x = 0.5$, $L_y = L_z = 0.05$) with 10k cells in four MPI ranks. Bottom panel: speedup as a function of the number of MPI ranks. The measured scaling is reasonable, with normalized speedup/parallel efficiency in the 70–100% range over the tested ranks.

3 Project execution plan

The central objective of this project is to transform JURHIATA from its current validated moving-mesh hydrodynamic baseline into a next-generation computational code for astrophysical fluid magnetohydrodynamics, specifically designed for GPU-accelerated high-performance computing systems. The project is structured around three tightly connected stages: *i)* GPU parallelization and performance optimization, *ii)* implementation of additional physical modules, and *iii)* scientific exploitation through large-scale astrophysical simulations.

This progression is intentional. The current code line already provides a stable numerical and parallel baseline on CPUs, including completed OpenMP and MPI implementations. This existing infrastructure substantially reduces development risk, because the first year can focus on the accelerator transition itself rather than on unresolved issues in the baseline solver. The subsequent stages then build on this validated computational core to incorporate the physical processes required by modern astrophysical codes and to deploy the resulting platform in scientific projects.

The overall project plan is designed so that the first two years are dominated by numerical and physical development, while the third year is primarily devoted to scientific exploitation. This ordering is essential: large-scale science simulations are meaningful only after the code has reached both numerical maturity and physical completeness (see Table 2).

Table 2: Three-year development plan for JURHIATA.

Year		Primary goal	Technical scope	Main outcomes
Year (2026)	**1**	GPU parallelization of the core solver	CUDA port of dominant kernels; profiling; MPI+GPU communication strategy; regression and benchmark validation against the CPU baseline.	GPU-ready core branch, scaling report, validated benchmark suite.
Year (2027)	**2**	Modular physics expansion	Implementation and validation of MHD, self-gravity, cooling/heating, and feedback modules.	Physics-enabled release candidate, expanded verification matrix, production-grade multi-physics test suite.
Year (2028)	**3**	Scientific exploitation	Large-scale GPU scaling tests followed by flagship astrophysical production runs and analysis pipelines.	Science papers, public datasets, reusable analysis products, and a mature scientific release.

Taken together, these three stages will transform JURHIATA from a validated moving-mesh hydrodynamic baseline into a scientifically competitive GPU-native platform for next-generation astrophysical simulations.

3.1 Stage I (2026): GPU parallelization and performance optimization

The first stage of the project focuses on implementing GPU parallelization of the current CPU-parallel baseline of JURHIATA. The goal is to accelerate the most CPU-consuming time parts of the solver while preserving the numerical behavior of the existing implementation.

The GPU transition will follow the methodology successfully adopted in the FARGO3D code (Benítez-Llambay & Masset, 2016). In this approach, the main computational routines are written as mesh operators that act on the grid elements (cells or faces), and these operators are translated into CUDA kernels where each GPU thread processes one element of the mesh. This strategy allows the GPU implementation to remain closely aligned with the CPU solver while exposing the massive parallelism required by modern accelerator architectures.

In practice, the translation from the existing C++ mesh operators to CUDA kernels will be performed using dedicated **Python scripts**, following the philosophy adopted in FARGO3D. These scripts automatically generate the CUDA kernels and launcher functions from the CPU implementation. This approach offers several advantages: it reduces code duplication between CPU and GPU versions, simplifies long-term

maintenance of the codebase, allows rapid development of new numerical operators without manually rewriting CUDA kernels, and keeps debugging anchored to the C++ routines.

3.1.1 Medium-term performance-portable option: Kokkos

A medium-term development path currently under evaluation for JURHIATA is the adoption of a performance-portable programming model based on Kokkos.[8] The main attraction of this approach is that it would allow a single, unified C++ codebase to execute efficiently across NVIDIA GPUs (CUDA), AMD GPUs (HIP), Intel accelerators (SYCL), and multicore CPUs (OpenMP/Threads), thereby reducing long-term maintenance costs and limiting vendor lock-in.

This possibility is particularly relevant for astrophysical HD/MHD solvers, which are expected to remain in use and evolve over many years while HPC architectures continue to diversify. In addition, Kokkos offers explicit control over data layout and memory spaces, which is valuable for optimizing memory-access patterns, one of the dominant performance bottlenecks in GPU-oriented solvers. Its support for hierarchical parallelism would also be advantageous for multi-dimensional stencil operations, reconstruction procedures, and Riemann-solver kernels.

Importantly, Kokkos can be introduced incrementally. This would make it possible to migrate the most computationally intensive kernels in stages, without requiring a complete rewrite of the code. For that reason, although the present Stage-I baseline remains centered on a direct CUDA-oriented path, we consider Kokkos a promising medium-term option for combining high performance with long-term portability and sustainability in JURHIATA.

3.1.2 Implementation strategy

Stage I adopts a **hybrid CPU–GPU architecture**. The Voronoi tessellation and mesh-topology management remain on the CPU,[9] while the computationally dominant hydrodynamic operators are executed on the GPU. This design avoids unnecessary complexity in the early stages of development and concentrates the acceleration effort on the kernels that dominate the runtime of the solver.

Following the FARGO3D methodology, the timestep cycle will be decomposed into a set of mesh-parallel operators. Each operator will be implemented as a CUDA kernel where GPU threads process independent mesh elements. The CUDA kernels will be generated automatically from the corresponding C++ routines through the Python translation scripts described above. This maintains a clear correspondence between the CPU implementation and the GPU implementation, simplifying validation and debugging.

The primary GPU kernels targeted in Stage I correspond to the most computationally expensive parts of the solver: *i)* gradient reconstruction; *ii)* slope limiting and MUSCL–Hancock predictor steps, *iii)* face-centered Riemann flux evaluation; *iv)* conservative update of hydrodynamic variables; *v)* and global timestep (CFL) reduction. Each of these kernels operates naturally over large sets of cells or faces and are therefore well suited for massively parallel execution on GPUs.

3.1.3 Memory layout and data movement

To achieve efficient GPU execution, the internal data structures of the code adopt a **Structure-of-Arrays (SoA)** layout for conserved variables, primitive variables, gradients, and geometric quantities. This memory layout enables coalesced memory access across GPU threads and improves effective memory bandwidth utilization.

Additional optimization efforts will focus on minimizing host–device data transfers and maintaining persistent device-side data structures whenever possible. Operations involving face-based data will be organized to reduce gather/scatter overhead and to improve memory locality.

3.1.4 Hybrid MPI+GPU parallelization

The final execution model will combine distributed-memory MPI parallelization with GPU acceleration inside each MPI rank. In this configuration, each MPI process evolves a spatial subdomain of the simulation, while the dominant numerical kernels are executed on the local GPU.

[8] Kokkos is a C++ performance-portability framework that allows a common source base to target multicore CPUs and multiple accelerator back ends through explicit abstractions for execution and memory spaces. See the Kokkos Core Wiki and the official GitHub repository.

[9] We are also developing a native Voronoi tessellation implementation using the same SoA architecture, and we plan to incorporate it into the code in the medium term. This implementation will be optimized for GPUs, since tessellation is currently one of the most computationally expensive CPU modules.

Communication between subdomains will rely on asynchronous ghost-cell exchanges and non-blocking MPI calls. This allows communication to overlap with GPU computation, which is essential for strong scaling on multi-GPU clusters.

3.1.5 Validation and performance targets

The GPU implementation will be validated against the current CPU baseline using the standard hydrodynamic benchmark suite. Numerical results must remain consistent with the CPU solver within predefined tolerances.

Performance evaluation will include kernel-level profiling and strong/weak scaling tests on multi-GPU systems. Based on the current hotspot distribution and results reported for GPU HD/MHD codes Schive et al. (2018); Caddy et al. (2024); Stone et al. (2024), the expected performance improvements are:

- kernel-level acceleration of approximately 8×–25× relative to a single CPU core,
- overall hybrid-node speedup of approximately 2.5×–5× compared with the current CPU-only baseline,
- and potential speedups approaching 6×–8× for large simulations once communication overlap and memory-transfer optimization are fully implemented.

These developments will position `JURHIATA` as a GPU-capable moving-mesh code compatible with modern accelerator-based supercomputing infrastructures.

3.2 Stage II (2027): Modular physics expansion

Once the GPU-accelerated core reaches a stable and validated operational level, the second stage of the project will focus on extending the physical scope of the code. The present baseline solves hydrodynamics, and the broader scientific potential of the project depends on incorporating the physical processes required by realistic astrophysical applications. This stage is therefore devoted to the implementation of modular physics, with particular emphasis on methods that are robust, well tested in the literature, and compatible with the moving-mesh framework.

3.2.1 Modular design philosophy

A central design principle of `JURHIATA` is modularity. New physical processes are not introduced as ad hoc additions to the solver, but as well-defined modules with stable interfaces to the timestep pipeline and to the shared state containers. This design minimizes coupling errors, simplifies staged validation, and allows physical modules to be enabled or disabled depending on the scientific application.

Within this proposal, Stage II prioritizes the first wave of essential astrophysical modules: *i)* ideal magnetohydrodynamics (in process); *ii)* self-gravity; *iii)* star formation; *iv)* radiative cooling and heating; *v)* and feedback source terms.

3.2.2 Magnetohydrodynamics

Stage II upgrades the solver from pure HD to ideal MHD on a moving Voronoi mesh. The conservative system is written as

$$\frac{\partial \mathbf{U}_{\mathrm{MHD}}}{\partial t} + \nabla \cdot \mathbf{F}_{\mathrm{MHD}}(\mathbf{U}_{\mathrm{MHD}}) = \mathbf{S}, \qquad \mathbf{U}_{\mathrm{MHD}} = \begin{pmatrix} \rho \\ \rho \mathbf{v} \\ \rho e \\ \mathbf{B} \end{pmatrix}, \tag{25}$$

with total energy density

$$e = u + \frac{1}{2}|\mathbf{v}|^2 + \frac{1}{2}\frac{|\mathbf{B}|^2}{\rho}, \qquad p_{\mathrm{tot}} = p + \frac{1}{2}|\mathbf{B}|^2, \tag{26}$$

and the solenoidal constraint $\nabla \cdot \mathbf{B} = 0$. The source vector $\mathbf{S}$ includes gravity and thermochemical source terms when the corresponding modules are active. The corresponding ideal-MHD flux tensor and source vector are

$$\mathbf{F}_{\mathrm{MHD}}(\mathbf{U}_{\mathrm{MHD}}) = \begin{pmatrix} \rho \mathbf{v} \\ \rho \mathbf{v} \otimes \mathbf{v} + p_{\mathrm{tot}} \mathbf{I} - \mathbf{B} \otimes \mathbf{B} \\ (\rho e + p_{\mathrm{tot}}) \mathbf{v} - (\mathbf{v} \cdot \mathbf{B}) \mathbf{B} \\ \mathbf{B} \otimes \mathbf{v} - \mathbf{v} \otimes \mathbf{B} \end{pmatrix}, \tag{27}$$

$$\mathbf{S} = \begin{pmatrix} 0 \\ -\rho \nabla \Phi \\ -\rho\, \mathbf{v} \cdot \nabla \Phi + (\mathcal{H} - \mathcal{L}) + \dot{e}_{\mathrm{fb}} \\ \mathbf{0} \end{pmatrix}, \tag{28}$$

where Φ is the gravitational potential and $\mathcal{H} - \mathcal{L}$ and $\dot{e}_{\mathrm{fb}}$ denote net heating/cooling rates and feedback energy source terms, respectively (see below).

Numerically, the MHD branch follows the same second-order MUSCL–Hancock **ALE** (Arbitrary Lagrangian–Eulerian) pipeline used in HD, but with magnetic reconstruction and MHD fluxes. Interface states are reconstructed with slope limiting, face fluxes are computed with the **HLLD** (Harten–Lax–van Leer Discontinuities) Riemann solver in the moving-face frame (Miyoshi & Kusano, 2005, see also Mocz et al. (2014; 2016)), and conservative updates evolve mass, momentum, total energy, and magnetic components together.

To control divergence errors, Stage II adopts hyperbolic-parabolic Dedner cleaning (Dedner et al., 2002),

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \cdot (\mathbf{B} \otimes \mathbf{v} - \mathbf{v} \otimes \mathbf{B}) = -\nabla \psi, \tag{29}$$

$$\frac{\partial \psi}{\partial t} = -c_h^2\, \nabla \cdot \mathbf{B} - c_r \psi, \tag{30}$$

where ψ is the auxiliary Dedner scalar field that propagates and damps divergence errors, c_h is tied to the local fast magnetosonic speed, and c_r controls damping. In discrete form, the cell-centered divergence diagnostic is evaluated as

$$(\nabla \cdot \mathbf{B})_i \simeq \frac{1}{V_i} \sum_j (\mathbf{B}_{ij} \cdot \mathbf{n}_{ij})\, A_{ij}, \tag{31}$$

and used to evolve ψ consistently with the local MHD update.

Operationally, each MHD timestep applies: (i) divergence estimation from face-centered geometry, (ii) Dedner source update for ψ, (iii) induction correction via $-\nabla\psi$, and (iv) boundary-consistent treatment of ψ.

The MHD implementation will be accepted only after passing a dedicated benchmark suite: Brio–Wu shock tube, circularly polarized Alfvén-wave convergence, Orszag–Tang vortex, MHD rotor, magnetic field-loop advection, and MHD blast-wave test s.

3.2.3 Self-gravity

Many of the astrophysical systems targeted by the project are self-gravitating. The self-gravity module will therefore be built around the Poisson equation,

$$\nabla^2 \Phi = 4\pi G \rho, \qquad \mathbf{g} = -\nabla \Phi. \tag{32}$$

The baseline implementation path is an OctTree short-range solver with opening-angle control and multipole nodes, following the classical Barnes–Hut family of hierarchical gravity solvers (Barnes & Hut, 1986) and later fast-multipole developments (Greengard & Rokhlin, 1987). For large periodic simulations, the production path is a TreePM split,

$$\Phi(\mathbf{x}) = \Phi_{\mathrm{PM}}(\mathbf{x}) + \Phi_{\mathrm{tree,short}}(\mathbf{x}), \tag{33}$$

which combines particle-mesh long-range gravity with tree-based short-range corrections (e.g., Springel, 2005).

A particularly relevant grid-code reference is the `FLASH` OctTree implementation of Wünsch et al. (2018), which shows that a distributed AMR tree can compute both self-gravity and angle-averaged optical depths with good accuracy and competitive cost. This is especially attractive for JURHIATA because, once such a tree infrastructure is available, it offers a comparatively direct path toward tree-based radiative transfer as well (Wünsch et al., 2018; 2021).

3.2.4 Cooling and heating

Astrophysical gas in galaxies cannot be described by adiabatic hydrodynamics alone. Following the `SMUGGLE` framework[10] of Marinacci et al. (2019), Stage II will model the ISM thermodynamics with coupled cooling and heating source terms in the internal-energy equation,

[10] `SMUGGLE` is a subgrid ISM and stellar-feedback model introduced for moving-mesh galaxy simulations, designed to couple multiphase thermochemistry, star formation, and feedback in a numerically robust way.

$$\left.\frac{d(\rho u)}{dt}\right|_{\rm src} = -\mathcal{L}(\rho, T, Z) + \mathcal{H}(\rho, T, J_\nu) + \dot{e}_{\rm fb}, \tag{34}$$

where $\mathcal{L}$ represents radiative cooling losses, $\mathcal{H}$ accounts for heating processes (e.g. UV background, cosmic rays, and photoelectric heating), and $\dot{e}_{\rm fb}$ represents the local energy injection rate due to stellar feedback from the newborn stars (see below).

The cooling and heating module will combine primordial H/He cooling, metal-line cooling, and low-temperature atomic, fine-structure, and molecular cooling, allowing the gas to span the full thermal range required for a multiphase ISM, from hot diffuse material down to cold star-forming gas (Vogelsberger et al., 2013; Hopkins et al., 2018; Marinacci et al., 2019). To avoid overestimating UV heating in dense regions, the model will also include density-dependent self-shielding corrections that reduce ionization and heating rates where the gas becomes optically thick (Rahmati et al., 2013; Vogelsberger et al., 2013).

On the heating side, the baseline implementation will include photoheating from the metagalactic UV background, cosmic-ray heating, and dust/PAH photoelectric heating, following established galaxy-formation and ISM models (Guo & Oh, 2008; Wolfire et al., 2003; Hopkins et al., 2018; Marinacci et al., 2019). These source terms will be integrated with a stiffness-aware operator-split scheme so that rapid cooling in dense gas and strong heating near young stars remain numerically stable, enabling the coexistence of cold, warm, and hot gas phases within the same simulation.

3.2.5 Star formation and stellar population evolution

Star formation will follow a stochastic cell-conversion prescription as in Marinacci et al. (2019): collisionless stellar particles (simple stellar populations with a standard initial mass function, IMF) are spawned only in cold, dense, gravitationally bound gas. The baseline thresholds are a high-density criterion (typically, $n_{\rm H} \gtrsim 100\,{\rm cm}^{-3}$) plus a local boundedness/virial condition, so star formation is restricted to collapsing ISM regions.

The local star formation rate follows the standard *efficiency per free-fall time* ($\epsilon_{\rm ff}$) prescription,

$$\dot{\rho}_\star = \epsilon_{\rm ff} \frac{\rho}{t_{\rm ff}}, \tag{35}$$

where $\dot{\rho}_\star$ is the star formation rate density, ρ is the gas density, and $t_{\rm ff} = \sqrt{3\pi/(32G\rho)}$ is the local free-fall time. Fiducially, $\epsilon_{\rm ff} \sim 0.01$. This formulation is widely used in galaxy simulations (e.g., Springel & Hernquist, 2003; Hopkins et al., 2014; Marinacci et al., 2019) and is consistent with observations of star-forming molecular clouds (e.g., Kennicutt, 1998).

Each stellar particle represents a single-age stellar population characterized by an IMF. The stellar particle evolves in time according to stellar evolution models that track the lifetimes and mass-loss histories of stars of different masses. This time-dependent evolution determines the rate at which mass, metals, energy, and momentum are returned to the surrounding gas.

In practice, the feedback produced by a stellar particle depends on its age and initial mass and is typically computed using tabulated stellar evolution yields. These tables determine the timing of supernova explosions, the strength of stellar winds, and the radiative output of young massive stars. This approach ensures that the feedback energy injection follows the physical timescales of stellar evolution rather than being applied instantaneously.

For galaxy-scale Stage II runs, this stochastic star-formation prescription is robust and the most natural baseline.However, for calculations that explicitly target star formation inside resolved molecular clouds, a sink-particle strategy becomes more appropriate. In that regime, a particularly relevant alternative is the adaptive-mesh sink-particle algorithm of Federrath et al. (2010), which was designed to form sinks only after a set of local checks has been satisfied, including density-threshold, gravitational boundedness, converging-flow, Jeans-instability, and local-potential-minimum criteria. This makes the Federrath et al. (2010) approach especially attractive for future `JURHIATA` studies focused on clustered star formation, core growth, and local accretion histories rather than on effective galaxy-scale star-formation laws.

3.2.6 Stellar feedback

Stellar feedback will be implemented as a local, multi-channel coupling model in which mass, metals, energy, and momentum from each stellar particle are distributed to neighboring gas cells with conservative geometric weights, closely following the philosophy of Marinacci et al. (2019). A key design choice is that the affected gas remains fully hydrodynamically active after injection: outflows, bubbles, and turbulence

should emerge from the resolved gas response rather than from decoupled wind particles or globally disabled cooling.

Radiative feedback

For young massive stars, the first radiative channel will be photoionization. The ionizing luminosity of each stellar particle will be estimated from its age and mass, converted into an ionizing photon rate,

$$\dot{N}_\gamma = \frac{L_\star}{\langle h\nu \rangle}, \tag{36}$$

where $L_\star$ is the ionizing luminosity of the stellar particle and $\langle h\nu \rangle$ is the mean energy of ionizing photons. Following Marinacci et al. (2019), photoionization will be applied probabilistically because the unresolved Strömgren mass is usually smaller than the full coupling region: eligible gas cells are heated to a characteristic H II-region temperature of order 10^4 K and their radiative cooling is temporarily disabled over the stellar time-step. In parallel, direct radiation pressure will inject momentum as

$$\Delta p_{\rm rad} = \frac{L_\star}{c}\,(1+\tau_{\rm IR})\,\Delta t, \qquad \tau_{\rm IR} = \kappa_{\rm IR}\,\Sigma_{\rm gas}, \tag{37}$$

where $\Delta p_{\rm rad}$ is the momentum injected by radiation pressure during the timestep Δt, c is the speed of light, $\tau_{\rm IR}$ is the infrared optical depth, $\kappa_{\rm IR}$ is the infrared opacity (which scales with metallicity), and $\Sigma_{\rm gas}$ is the local gas surface density surrounding the source. This UV/radiative channel is especially important before the first supernovae, because it pre-processes dense birth clouds and lowers the ambient density into which later explosions occur.

An attractive medium-term alternative, especially if the self-gravity branch adopts a `FLASH`-like OctTree infrastructure, is to promote the UV channel to an explicit tree-based radiative-transfer solver. The `TreeRay` method of Wünsch et al. (2021) reuses the same OctTree to perform reverse ray tracing with tree-accelerated integration, has computational cost essentially independent of the number of radiation sources, and is therefore well suited to clustered star formation and diffuse-emission problems (Wünsch et al., 2018; 2021).

Stellar winds

The second channel is the continuous return from OB and AGB stellar winds. In the implementation path motivated by Hopkins et al. (2018); Marinacci et al. (2019), each stellar particle carries age- and metallicity-dependent mass-loss rates; the returned mass and metals are injected continuously, while the associated wind energy and momentum are computed from the wind mass loss ($M_{\rm loss}$) and characteristic wind energetics,

$$E_{\rm wind} = M_{\rm loss}\,\psi\,10^{12}\,{\rm erg\,g^{-1}}, \qquad p_{\rm wind} = \sqrt{2\,M_{\rm loss}\,E_{\rm wind}}. \tag{38}$$

where $E_{\rm wind}$ is the total wind energy injected, ψ is a dimensionless factor that sets the effective specific wind energetics, and $p_{\rm wind}$ is the corresponding wind momentum. OB winds act during the first few Myr and complement radiative feedback by dispersing dense gas around young stars, whereas AGB winds provide later-time mass recycling and additional momentum/energy input from older stellar populations.

A useful alternative algorithm for this channel is the wind (and supernova) feedback implementation in the `TORCH` code (Wall et al., 2020): stellar winds are injected directly into the gas with a momentum-conserving scheme, and unresolved losses in the shocked wind are represented through an effective mass-loading term. As in `SMUGGLE`, the wind channel will couple to the same local neighborhood as the other feedback terms, ensuring consistent conservation and avoiding special hydrodynamic decoupling.

Supernova feedback

The third and dominant momentum channel will be discrete Type II and Type Ia supernova feedback. Rather than smearing explosions continuously in time, the baseline Stage II implementation will sample individual events from each stellar particle using an IMF-based Type II rate and a delay-time-distribution model for Type Ia explosions, with a time-step limiter that keeps the expected number of events per step of order unity (Marinacci et al., 2019). The injected energy budget is

$$E_{\rm SN,tot} = f_{\rm SN}\,E_{51}\,(N_{\rm SNII} + N_{\rm SNIa}), \tag{39}$$

where $E_{\rm SN,tot}$ is the total energy budget assigned to supernova feedback, $f_{\rm SN}$ is the fraction of the nominal supernova energy effectively coupled to the gas, E_{51} is the energy per supernova in units of 10^{51} erg, and $N_{\rm SNII}$ and $N_{\rm SNIa}$ are the numbers of core-collapse and Type Ia supernova events occurring in the timestep. As in the case of wind feedback, the momentum carried by the blast wave is estimated, for

each type, as $p_{\rm SN} = \sqrt{2\,E_{\rm SN}\,M_{\rm SN}}$, where $E_{\rm SN}$ is the supernova energy effectively coupled to the gas and $M_{\rm SN}$ is the ejecta mass associated with the explosion.

When the cooling radius is unresolved the injected momentum is boosted to the terminal-momentum value expected after the unresolved Sedov–Taylor phase (Hopkins et al., 2018; Kim & Ostriker, 2015; Marinacci et al., 2019).

3.2.7 Algorithm choices

A key principle of Stage II is to adopt algorithms with strong precedent in the literature rather than unnecessarily speculative formulations.

For *resolved* star formation inside molecular clouds, the most suitable combination of algorithms are: sink-particle creation and accretion following Federrath et al. (2010); OctTree self-gravity and tree-based UV/radiative feedback in the spirit of Wünsch et al. (2018; 2021); and stellar-wind and supernova feedback following Wall et al. (2020), with the resolution-dependent supernova injection strategy of Simpson et al. (2015). This combination is better matched to local collapse, protostellar clustering, feedback-driven cloud dispersal, and the need to resolve individual star-forming sites, whereas the Marinacci et al. (2019) prescription remains the more appropriate baseline for global galactic-disk simulations.

This strategy is intended to avoid common numerical failures such as spurious stellar particle creation, overcooling, and unstable source-term integration.

3.3 Stage III (2028): Scientific exploitation

Stage III is the scientific exploitation of the code. At this point, the code is expected to include a GPU-accelerated core and a first-generation modular physics stack (MHD, self-gravity, stellar particles, cooling/heating, and feedback), enabling production-grade simulations.

The scientific importance of Stage III lies in the fact that moving-mesh methods and GPU acceleration are not ends in themselves. Their value resides in enabling simulations that are simultaneously more accurate, more scalable, and more ambitious than those accessible with conventional CPU-only methods or with numerically less appropriate discretizations. The combination of reduced advection errors, robust shock capturing, adaptive moving geometry, and accelerator-enabled throughput opens a particularly promising window for the study of supersonic, multiphase, self-gravitating, and magnetized astrophysical flows.

Before the main scientific production phase begins, Stage III includes a mandatory large-scale scaling simulation on the largest GPU allocation available at the time. This test is a hard gate for production science. Its purpose is to verify communication patterns, memory pressure, stability, and end-to-end throughput near maximum practical concurrency. This is especially important for a moving-mesh code, where irregular communication and dynamic topology can create scalability bottlenecks that are not visible in smaller validation runs.

All Stage III validation tests and production simulations will be executed on the acquired project cluster and, when available, on larger national allocations (e.g., Miztli/Coatlicue, see Section 3.7).

3.3.1 Simulations strategy

The scientific exploitation plan is organized around a flagship Milky-Way-like disk-galaxy simulation using the full validated physics stack. Around this flagship objective, the proposal defines a single high-resolution fiducial isolated Milky-Way-like disk simulation for detailed diagnostics of interstellar structure and star-formation regulation. The fiducial Stage III run will start from Milky-Way-like initial conditions, considering the dark matter halo, stellar disk, bulge, and gas components, with $N_{\rm halo} = 10^8$, $N_{\rm stars,disk} = 10^8$, $N_{\rm bulge} = 10^7$, and $N_{\rm gas} = 10^8$, i.e. $\sim 3.1 \times 10^8$ initial particles/cells. During the simulation, adaptive gas refinement is expected to increase the total resolution to as many as 10^9 **particles/cells**. The scientific questions that motivate this simulation include: *i)* how star formation is regulated in Milky-Way-like disks (e.g., Ostriker et al., 2010; Sun et al., 2023; Ellison et al., 2024); *ii)* how metallicity-dependent cooling reshapes the multiphase interstellar medium; and *iii)* how magnetic fields and feedback influence molecular cloud evolution and galactic outflows (e.g., Kim & Ostriker, 2015; Hopkins et al., 2018; Kim et al., 2021).

Milky-Way-like initial conditions

The Stage III fiducial initial conditions will follow the standard multi-component galaxy construction used in equilibrium galaxy generators such as `GalIC` (Yurin & Springel, 2014). The collisionless system

is described by a Hernquist dark-matter halo, an exponential stellar disk with a sech^2 vertical profile, and a Hernquist bulge, while the gaseous disk is added consistently to build a full moving-mesh initial configuration. This choice provides a stable, observationally anchored Milky-Way-like galaxy model with analytic control over halo concentration (c), spin (λ), stellar disk mass (M_{disk}), bulge mass (M_{bulge}), and structural scale parameters (disk height, bulge size, etc.; see Table 3).

For the fiducial Milky-Way-like case, we adopt the global structural parameters of the isolated galaxy models used by Marinacci et al. (2019), mapped into the `GalIC`-style parameterization summarized in Table 3.

Table 3: Representative Milky-Way-like initial-condition parameters for the Stage III fiducial run. The parameterization follows the equilibrium galaxy-construction framework of `GalIC` (Yurin & Springel, 2014). Here M_{disk} and M_{bulge} are mass fractions relative to the total halo mass.

Galaxy	M ($10^{12}\,M_\odot$)	c	V_{200} (km s^{-1})	λ	M_{disk}	M_{bulge}	DiskHeight	BulgeSize
MW	1.0	12	169	0.034	0.035	0.0093	0.10	0.05

3.4 Quarterly stage schedule

We track each stage by quarter (Q1–Q4) with a uniform traffic-light status: green = completed, yellow = in progress, red = pending.

Table 4: Quarterly schedule by stage (blank cells indicate no planned work in that quarter).

Status Color Legend					
Done		In progress		Pending	

Activity	Q1	Q2	Q3	Q4	Global Status
Stage 1 (2026): Core GPU parallelization					
CPU baseline stabilization (OpenMP+MPI), benchmark closure, and reproducibility controls	Done	Done			Done
CUDA port of core kernels (reconstruction, Riemann, update, reductions)	To do	To do	To do		Pending
GPU profiling and scaling optimization (roofline, overlap, transfer minimization)		To do	To do	To do	Pending
Stage-1 release candidate package (performance report + reproducible scripts)				To do	Pending
Stage 2 (2027): Modular physics expansion					
MHD core completion (HLLD + Dedner + CT option) and benchmark closure	In prog.	To do	To do		In progress
Self-gravity module (Poisson + TreePM + Jeans refinement) and verification closure	To do	To do	To do		Pending
Subgrid physics module (stellar particles + cooling/heating + feedback) and integration		To do	To do	To do	Pending
Coupled multi-physics validation matrix and acceptance-threshold closure			To do	To do	Pending
Stage-2 physics-enabled release candidate and expanded test suite				To do	Pending
Stage 3 (2028): Scientific exploitation					
Pre-science massive scaling simulation (Miztli (UNAM) / Coatlicue)	To do	To do			Pending
Final scientific setup (IC suite and execution protocol)	To do	To do			Pending
Full-physics production simulations (high resolution isolated MW-type galaxy)		To do	To do	To do	Pending
Final deliverables (papers, open datasets, and public release package)			To do	To do	Pending

3.5 Scientific objectives

General objective: develop, validate, and scientifically exploit JURHIATA as a GPU-oriented moving-mesh astrophysical HD/MHD code that combines robust physics, scalable performance, and reproducible open-science delivery.

Some specific objectives are:

1. **SO1: Validate quantitative HD accuracy on moving Voronoi meshes.**
   Build and execute a benchmark matrix that measures shock capturing, contact resolution, and conservation behavior across canonical HD tests and multiple resolutions.

2. **SO2: Ensure robust MHD evolution with controlled divergence errors.**
   Implement and stress-test the MHD branch under shock-dominated and wave-propagation regimes, with explicit monitoring of magnetic-field divergence and solver robustness in long integrations.

3. **SO3: Execute MPI+GPU performance closure for production-scale runs.**
   Port and optimize dominant kernels for GPU execution, quantify scaling limits, and validate that performance gains are achieved without degrading numerical reproducibility.

4. **SO4: Integrate Stage-II physics modules for star-formation studies.**
   Couple self-gravity, stellar particles, cooling/heating, and feedback in a modular pipeline with staged integration gates to preserve stability and physical consistency.

5. **SO5: Deliver a reproducible and reusable software/data release.**
   Package code, benchmark assets, and post-processing workflows in versioned artifacts that external users can rerun to reproduce key validation figures and metrics.

3.6 Expected products

The project-level publication commitment is five articles:

- Three Review-style implementation papers at the end of Stage II in *Revista Mexicana de Astronomía y Astrofísica* (RevMex), focused on self-gravity, stellar particles, and feedback/cooling-heating implementations.
- Two papers at the end of Stage III in *MNRAS*: one full-code description paper (numerics + GPU implementation + validation) and one flagship science paper based on the Milky-Way-like simulation.
- Public release version of the JURHIATA source code, benchmark suites, and reproducible analysis products.

3.7 Computing resource justification

The compute demand is driven by three concrete requirements: GPU-kernel verification during CUDA migration, MPI+GPU multi-node validation for added physics modules, and long production campaigns in Stage 3. The corresponding infrastructure objective is staged acquisition of a dedicated **3-node GPU cluster**, purchasing one node per stage. Each node will be based on a modern GPU-enabled HPC server equipped with **NVIDIA RTX PRO 6000 Blackwell GPUs with 96 GB GDDR7 memory**, providing large on-device memory capacity and full CUDA compatibility for astrophysical simulations.

Each node configuration is based on dual-socket AMD EPYC processors (96 cores total), DDR5 ECC memory, NVMe storage, and one high-memory GPU accelerator, ensuring a balanced CPU–GPU architecture suitable for hybrid MPI+CUDA workloads.

Table 5: Compact compute-demand justification.

Stage	Workload Focus	Demand	Needed Capacity
2026	Core CUDA migration + regression suite (HD/MHD).	∼500 node-h/year	1 node
2027	MPI+GPU multi-physics validation (gravity/sinks/feedback/cooling).	∼1,500 node-h/year	2 nodes
2028	Pre-science massive scaling campaign (Miztli/Coatlicue) + full-physics production campaign and reproducibility reruns.	∼5,000 node-h/year	3 nodes

3.8 Budget and staged acquisition plan

The Stage 1 and Stage 2 costs correspond to the acquisition of full GPU-enabled HPC nodes (including VAT), each priced at approximately MXN 750,000 based on vendor quotation. Stage 3 allocates the remaining budget for the final node, with flexibility to adjust configuration (e.g., reduced CPU or storage footprint) while maintaining GPU capability (see Table 6).

For administrative consistency with call rules, durable computing hardware and storage assets will be mapped to the corresponding *capital expenditure* category where required, with explicit stage-level traceability between technical milestones and financial line items. Detailed cost quotations are available at https://mixtli.inaoep.mx/mixtli/docs/cost-quotations-846.pdf.

Table 6: Compact budget (MXN) and cluster-growth plan.

Stage	Budget (MXN)	Procurement Action	Cluster Size
Stage 1 (2026)	750,000	Acquire Node 1 (AMD EPYC + RTX PRO 6000 96GB) and enable core GPU validation workflows.	1 node
Stage 2 (2027)	750,000	Acquire Node 2 for multi-node MPI+GPU physics validation.	2 nodes
Stage 3 (2028)	470,000	Acquire Node 3 and finalize production cluster configuration.	3 nodes

Indicative per-stage spending envelope: 70–75% compute node hardware, 10–15% storage, 5–10% interconnect/integration, and 8–12% power/cooling/warranty reserve.

This cluster supports three direct products: benchmark validation, student training in HPC/GPU workflows, and production simulation campaigns (including the Stage-3 Milky-Way-like scientific case). It also provides a local pre-production platform before national-scale execution on infrastructures such as Coatlicue.

3.9 Project management and team roles

The ownership matrix (see Table 3.9) summarizes the technical responsibilities and stage-by-stage execution roles used for governance and accountability.

Execution follows a weekly technical meeting, a monthly milestone review, and a formal end-of-stage acceptance check anchored to the validation matrix in Table 3.9. No new execution mode (e.g., MPI+GPU production) or physics module is promoted to science runs until benchmark scripts, figure regeneration, conservation diagnostics, and the relevant performance thresholds are closed on the reference hardware of that stage.

Table. Ownership matrix by participant.

ID	Participant	Institution	Primary Technical Ownership	Stage Responsibilities
PI	Dr. Manuel Zamora-Avilés	SECIHTI/INAOE	Project lead, architecture governance, and final V&V sign-off.	Leads Stage 1 (GPU core), Stage 2 (physics integration), and Stage 3 (science exploitation).
coPI	Dr. Javier Ballesteros-Paredes	IRyA (UNAM)	Co-PI; physics strategy (MHD/ISM/star formation) and science interpretation.	Stage 1 physics requirements; Stage 2 module priorities; Stage 3 science diagnostics/publications.
P1	Dr. Jose Franco	IA (UNAM)	Theoretical-astrophysics and feedback specialist.	Stage 2 feedback calibration design; Stage 3 interpretation of feedback-regulated disk evolution.
P2	Dr. Gilberto C. Gómez-Reyes	IRyA (UNAM)	Numerical methods lead (HD/MHD discretization and robustness).	Involved in all stages; co-leads numerical consistency and validation gates.
P3	Dr. Enrique Vázquez-Semadeni	IRyA (UNAM)	ISM and star-formation physics leadership.	Strong Stage 2–3 involvement in physical-model prioritization and calibration criteria.
P4	Dra. Adriana Gazol	IA (UNAM)	(Magneto)hydrodynamic modeling of the interstellar medium.	Strong Stage 2–3 involvement in MHD validation and science-analysis strategy.
P5	Dra. Griselda Arroyo Chavez	University of Arizona	MHD simulations, star formation, visualization, and SPH expertise.	Involved in all stages; leads project-wide scientific visualization, QA, and Stage 3 science visual products.
P6	Dr. Raúl Naranjo-Romero	INAOE	Cluster administration, deployment, HPC operations, and outreach coordination.	Stage 1 cluster setup; Stage 2 operations support; Stage 3 production reliability and intensive outreach in indigenous communities (secondary and preparatory levels).
P7	Dr. Abraham Luna Castellanos	INAOE	Millimeter-band astronomy, observational polarimetry, and astronomical instrumentation.	Stage 2 observational-diagnostics guidance; Stage 3 interpretation support for synthetic-observation and polarization-oriented products.
P8	Dr. Ruben Guerrero Gamboa	IRyA (UNAM)	MHD and adaptive-mesh-method specialist.	Stage 2 MHD/AMR cross-method validation; Stage 3 comparative diagnostics for robust interpretation.
P9	Dra. Vianey Edaly Camacho Pérez	National Taiwan Normal University	Numerical simulations and star formation.	Stage 2 cross-code validation (moving-mesh vs AMR); Stage 3 comparative diagnostics.
P10	M.C. Josué Gerardo López Castillo	INAOE	CUDA implementation, profiling, and performance engineering.	Stage 1 GPU kernels; Stage 2 acceleration support; Stage 3 performance support and outreach activities.
P11	M.C. Fabián Alberto Quesada Zúñiga	INAOE	Numerical-analysis pipelines and MHD diagnostics.	Stage 1 benchmark analytics; Stage 2 diagnostics; Stage 3 reproducible figures and outreach activities.
P12	M.C. Luis Andrés Hernández Cruz	INAOE	MHD analysis and synthetic-observable workflows.	Stage 1 validation support; Stage 2 synthetic-observable integration; Stage 3 science-ready mock products and outreach activities.

Continued on next page

ID	Participant	Institution	Primary Technical Ownership	Stage Responsibilities
P13	M.C. Alejandro Edmundo Aguilar Torrez	INAOE	Numerical simulations, data analysis, and stellar clusters.	Stage 1 CI automation; Stage 2 cross-physics regression; Stage 3 long-run validation and outreach activities.
P14	M.C. Karla Alejandra Gutierrez Davila	IRyA (UNAM)	Simulation-analysis specialist and galactic-evolution diagnostics.	Stage 2/3 support for science diagnostics, visual analysis, and dissemination products.

3.10 Risk and mitigation framework

To strengthen execution realism, Table 8 summarizes the main project risks, their operational impact, the mitigation gate used to control them, and the primary owner responsible for follow-up.

Table 8: Formal risk and mitigation summary.

Risk	Project impact	Mitigation and control gate	Owner
MPI+GPU scaling stalls at high concurrency	Delay in Stage-I closure and reduced Stage-III throughput.	Prioritize overlap of communication/computation, profile-driven kernel tuning, and a mandatory pre-production scaling run before any flagship science campaign.	PI + P10
MHD divergence-control instability in long runs	Reduced physical reliability of the Stage-II MHD branch.	Use HLLD→HLLE fallback, Dedner-parameter sweeps, CT fallback for strict cases, and benchmark-gated acceptance thresholds before release-candidate promotion.	PI + P2
Multiphysics coupling stiffness (cooling/feedback/gravity)	Integration delays and unstable coupled runs in Stage II.	Enforce module-by-module integration with regression gates (Evrard/Jeans, one-zone cooling, SN momentum tests) before enabling coupled production workflows.	PI + coPI
Cluster procurement or large-allocation scheduling delays	Reduced local test capacity or postponed massive-scaling campaign.	Stage the project around a local 3-node cluster, preserve CPU/MPI fallback validation paths, and request national allocations only after local pre-production closure.	PI + P6

4 Open science and impact

4.1 Reproducibility, open science, and release strategy

Following established scientific software reproducibility guidance, this project adopts:

- **Versioning:** semantic version tags for code and benchmark packs.
- **Data products:** curated IC files, parameter files, and reference outputs for all published tests.
- **Automation:** scripted run/plot pipelines and CI-based regression checks.
- **Release guides:** user-level and developer-level guides linked to each release.
- **Licensing policy:** explicit open-source license declaration for all public release packages.

Thus, the public `JURHIATA` repository will use the **Apache License 2.0**. This is a permissive license with explicit patent-language clauses, which is advantageous for broad scientific reuse and multi-institution collaboration in long-lived HPC codebases.

Importantly, the open-science commitment covers the full JURHIATA stack: not only the hydro/MHD core, but also all physics modules integrated during the project (including self-gravity, stellar particles, cooling/heating, and feedback) will be released in the public code line.

4.2 Scientific and technical impact

JURHIATA targets a high-value numerical gap: moving-mesh MHD with a hardware-forward architecture for future GPU production. The expected impact is twofold:

- **Scientific:** reduced method-driven biases in shock-dominated and magnetized flows.
- **Technical:** reusable moving-mesh software components, GPU-optimized high-performance workflows, and benchmark pipelines for the community.

4.3 Alignment with Mexican Government priority projects

The project has direct institutional and policy alignment with Mexico's federal science and technology agenda. The PI's appointment context (SECIHTI affiliation with commission at INAOE) places the effort within the public R&D ecosystem tasked with strengthening national computational sovereignty and high-impact scientific infrastructure. In this framework, JURHIATA is explicitly positioned as an enabling scientific-software component aligned with federal "priority projects," including the GPU-centered Coatlicue supercomputing initiative announced by President Claudia Sheinbaum Pardo.[11]

4.4 Outreach

Outreach targets are measurable: at least 3 public talks/year (Puebla, Morelia, Mexico City), 2 undergraduate HPC workshops/year, 1 high-school outreach activity/year, and 4 public communication outputs/year linked to project milestones. Dr. Raúl Naranjo-Romero will coordinate an intensive outreach track focused on indigenous communities, including activities for secondary and preparatory students. Student researchers will be actively involved in the design and execution of these activities.

5 References

References

Barnes J., Hut P., 1986, Nature, 324, 446

Benítez-Llambay P., Masset F. S., 2016, The Astrophysical Journal Supplement Series, 223, 11

Caddy M., Mebane R., Fielding D., Quataert E., Emerick A., Bryan G., O'Shea B., 2024, The Astrophysical Journal, 969, 100

Chang P., Wadsley J., Quinn T. R., 2017, Monthly Notices of the Royal Astronomical Society, 471, 3577

Dedner A., Kemm F., Kroener D., Munz C.-D., Schnitzer T., Wesenberg M., 2002, Journal of Computational Physics, 175, 645

Duffell P. C., MacFadyen A. I., 2011, The Astrophysical Journal Supplement Series, 197, 15

Ellison S. L., Pan H., Bluck A. F. L., et al., 2024, Monthly Notices of the Royal Astronomical Society, 527, 10201

Federrath C., Banerjee R., Clark P. C., Klessen R. S., 2010, The Astrophysical Journal, 713, 269

Fryxell B., Olson K., Ricker P., et al., 2000, The Astrophysical Journal Supplement Series, 131, 273

Greengard L., Rokhlin V., 1987, Journal of Computational Physics, 73, 325

Guo F., Oh S. P., 2008, Monthly Notices of the Royal Astronomical Society, 384, 251

Hopkins P. F., 2015, Monthly Notices of the Royal Astronomical Society, 450, 53

Hopkins P. F., Kereš D., Oñorbe J., et al., 2014, Monthly Notices of the Royal Astronomical Society, 445, 581

Hopkins P. F., et al., 2018, Monthly Notices of the Royal Astronomical Society, 480, 800

Kennicutt R. C., 1998, The Astrophysical Journal, 498, 541

Kim C.-G., Ostriker E. C., 2015, The Astrophysical Journal, 802, 99

Kim J., Ostriker E. C., Filippova N., 2021, The Astrophysical Journal, 911, 128

[11] Official announcement web page: https://www.gob.mx/

Marinacci F., et al., 2019, Monthly Notices of the Royal Astronomical Society, 489, 4233

Miyoshi T., Kusano K., 2005, Journal of Computational Physics, 208, 315

Mocz P., Vogelsberger M., Sijacki D., Pakmor R., Hernquist L., 2014, Monthly Notices of the Royal Astronomical Society, 442, 43

Mocz P., Pakmor R., Springel V., et al., 2016, Monthly Notices of the Royal Astronomical Society, 463, 477

Monaghan J. J., 1992, Annual Review of Astronomy and Astrophysics, 30, 543

Ostriker E. C., McKee C. F., Leroy A. K., 2010, The Astrophysical Journal, 721, 975

Pillepich A., Nelson D., Hernquist L., et al., 2018, Monthly Notices of the Royal Astronomical Society, 475, 648

Price D. J., 2012, Journal of Computational Physics, 231, 759

Rahmati A., Pawlik A. H., Raičević M., Schaye J., 2013, Monthly Notices of the Royal Astronomical Society, 430, 2427

Rycroft C. H., 2009, Chaos, 19, 041111

Schive H.-Y., Zhang U.-L., Chiueh T., Tsai Y.-C., 2018, Monthly Notices of the Royal Astronomical Society, 481, 4815

Simpson C. M., Bryan G. L., Hummels C., Ostriker J. P., 2015, The Astrophysical Journal, 809, 69

Springel V., 2005, Monthly Notices of the Royal Astronomical Society, 364, 1105

Springel V., 2010, Monthly Notices of the Royal Astronomical Society, 401, 791

Springel V., Hernquist L., 2003, Monthly Notices of the Royal Astronomical Society, 339, 289

Steinberg E., Yalinewich A., Sari R., 2016, Monthly Notices of the Royal Astronomical Society, 459, 1596

Stone J. M., Gardiner T. A., Teuben P., Hawley J. F., Simon J. B., 2008, The Astrophysical Journal Supplement Series, 178, 137

Stone J. M., Kannan R., Tomida K., et al., 2024, AthenaK: A Performance-Portable Magnetohydrodynamics Code for Exascale Astrophysical Simulations (arXiv:2409.16053)

Sun J., Leroy A. K., Ostriker E. C., et al., 2023, The Astrophysical Journal Letters, 945, L19

Teyssier R., 2002, Astronomy and Astrophysics, 385, 337

Toro E. F., 2009, Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction, 3 edn. Springer, Berlin, doi:10.1007/b79761, https://doi.org/10.1007/b79761

Toro E. F., Spruce M., Speares W., 1994, Shock Waves, 4, 25

Vandenbroucke B., De Rijcke S., 2016, Astronomy and Computing, 16, 109

Vogelsberger M., Genel S., Sijacki D., Torrey P., Springel V., Hernquist L., 2013, Monthly Notices of the Royal Astronomical Society, 436, 3031

Vogelsberger M., Genel S., Springel V., et al., 2014, Monthly Notices of the Royal Astronomical Society, 444, 1518

Wall J. E., Mac Low M.-M., McMillan S. L. W., Klessen R. S., Portegies Zwart S., Pellegrino A., 2020, The Astrophysical Journal, 904, 192

Weinberger R., Springel V., Pakmor R., 2020, The Astrophysical Journal Supplement Series, 248, 32

Wolfire M. G., McKee C. F., Hollenbach D., Tielens A. G. G. M., 2003, The Astrophysical Journal, 587, 278

Wünsch R., Walch S., Dinnbier F., Whitworth A. P., Palouš J., 2018, Monthly Notices of the Royal Astronomical Society, 475, 3393

Wünsch R., Walch S., Dinnbier F., Seifried D., Haid S., Klepitko A., Whitworth A. P., Palouš J., 2021, Monthly Notices of the Royal Astronomical Society, 505, 3730

Yalinewich A., Sari R., Steinberg E., 2015, The Astrophysical Journal Supplement Series, 216, 35

Yurin D., Springel V., 2014, Monthly Notices of the Royal Astronomical Society, 444, 62