# The CODEX

Official A.R.C./E.O.N.S. Manual

Author: James (JT) Ternullo

## Our Mission

To develop high-fidelity numerical simulations that advance our understanding of complex physical systems across multiple scientific domains—astrophysics, meteorology, oceanography, and geology—by tailoring mathematical models to the unique challenges of each field while maintaining a unified, adaptable framework.

## Our Goal

To create a versatile and robust computational suite that accurately models the dynamics of continuous media, enabling breakthroughs in scientific research, engineering applications, and data-driven decision making.

By focusing on high-fidelity numerical methods and problem specific adaptations, E.O.N.S. ensures precision, efficiency, and adaptability in simulations. This approach allows us to push the boundaries of scientific inquiry while maintaining a flexible framework that can evolve with new challenges and discoveries.

# Introduction

Explorations with Optimized Numerical Simulations (EONS) is simulation software aimed at solving reactive flows in Earth Science. This is a broad scope and will therefore specialize in astrophysical flows, branching out to other areas once we get stable methods and good results. With this framework we plan on pursuing users of all educational backgrounds. Plans for an interactive middle/high school friendly user interface are being considered. The goal is to make EONS useful to as many people as possible, whether it is for education or research, EONS will be one of the leaders in the evolution of reactive flows in earth science.

At the core of EONS is the numerical suite, Algorithms for Realistic Computation (ARC). This numerical suite contains the fundamental numerical methods that are essential for computational simulations. These methods are categorized as follows:

- Differentiation – Computing derivatives to measure rates of change, essential for modeling dynamic systems.
- Integration – Approximating definite integrals to track cumulative quantities over space or time.
- Iterative Methods – Solving equations numerically when direct analytical solutions are infeasible.
- Interpolation – Constructing continuous functions from discrete data points for smooth representations.
- Linear Algebra – Creating and computing matrix operations.
- Evolution Schemes – Algorithms used to advance solutions of the Euler equations.

While numerical methods handle individual mathematical operations, Evolution Schemes combine these methods into structured, technical computations that allow for the solution of complex physical systems. The Evolution Schemes in A.R.C. employ a variety of strategies to ensure high accuracy and stability when applied to real-world simulations. These include:

- Godunov Methods – Solving hyperbolic conservation laws with shock-capturing capabilities.
- Predictor-Corrector Schemes – Refining solutions iteratively to improve accuracy.
- Riemann Problem Solvers – Handling discontinuities in fluid flow and other physical systems.
- Weighted Essentially Non-Oscillatory (WENO) Schemes – Preserving high-order accuracy while reducing spurious oscillations near sharp gradients.

By carefully selecting and combining numerical methods, we construct solvers that are adaptable to various scientific domains, from astrophysical simulations to meteorological modeling.

Each numerical method has its own underlying strategy, strengths, and limitations. The effectiveness of an algorithm depends on how it manipulates numerical data and interacts with other components of the scheme. Some methods excel in certain conditions but struggle in others. For example:

- Finite difference methods are efficient for smooth problems but struggle near discontinuities.
- Spectral methods offer high accuracy but can be computationally expensive.
- Iterative solvers are useful for large systems but may converge slowly in ill-conditioned problems.

Recognizing these strengths and limitations ensures that users of E.O.N.S. can make informed decisions when selecting methods for their simulations. Moreover, not every algorithm is universally interchangeable—some methods complement specific solvers, while others introduce instability or inefficiency when substituted. Establishing a clear understanding of these relationships enhances the flexibility and usability of the E.O.N.S. framework.

# Algorithms for realistic Computation

## Evolution Schemes

In this section, we provide a comprehensive catalog of the Evolution schemes implemented throughout E.O.N.S., ranging from classical nonconservative finite difference methods to modern conservative finite volume techniques. Each algorithm is accompanied by a step-by-step walkthrough and annotated implementation, designed to serve both as a practical tool and an educational source. We are currently following several sources of which the most important have been Laney's 1998 "Computational Gas Dynamics" and Toro's 1999 "Riemann Solvers and Numerical Methods for Fluid Dynamics." We will use the test models in these resources to compare our employment of the numerical schemes.

In the most basic sense, when we discretize a derivative there are several simple strategies. Consider the following equation

$$\frac{\partial u}{\partial t} + \frac{\partial \rho u}{\partial x} = 0.$$

This partial differential equation has a time derivative and a space derivative. When considering the discretization of these derivatives there are three basic schemes, forward, backward, and central. These schemes take derivatives and express them as algebraic equations, for instance

$$\frac{\partial u}{\partial x} = \frac{u_{i+1} - u_i}{\Delta x}, \quad \text{Forward Difference}$$

$$\frac{\partial u}{\partial x} \approx \frac{u_i - u_{i-1}}{\Delta x}, \quad \text{Backward Difference}$$

$$\frac{\partial u}{\partial x} \approx \frac{u_{i+1} - u_{i-1}}{2\Delta x}, \quad \text{Central Difference.}$$

You can also write the time derivatives in the same way by replacing $x$ with $t$. Doing so gives us access to several discretization schemes which are summarized in the following table.

| Discretization Scheme | Explicit/Implicit | Order of Accuracy (Time, Space) | Stability Condition |
|---|---|---|---|
| Forward Time Forward Space (FTFS) | Explicit | 1st, 1st | Unstable |
| Forward Time Backward Space (FTBS) | Explicit | 1st, 1st | Conditionally Stable (CFL) |
| Forward Time Central Space (FTCS) | Explicit | 1st, 2nd | Unconditionally Unstable (CFL) |
| Backward Time Forward Space (BTFS) | Implicit | 1st, 1st | Unconditionally Stable (CFL) |
| Backward Time Backward Space (BTBS) | Implicit | 1st, 1st | Unconditionally Stable (CFL) |
| Backward Time Central Space (BTCS) | Implicit | 1st, 2nd | Unconditionally Stable (CFL) |
| Central Time Forward Space (CTFS) | Explicit | 2nd, 1st | Unstable |
| Central Time Backward Space (CTBS) | Explicit | 2nd, 1st | Unstable |
| Central Time Central Space (CTCS) | Explicit | 2nd, 2nd | Conditionally Stable (CFL, symmetry) |

Table 1: A table of discretization schemes, their accuracy, and condition for stability. Notice implicit schemes are unconditionally stable, and that the FTFS method is unstable.

---

Example:

Let’s take the above equation and discretize it using the FTCS method.

$$\frac{\partial u}{\partial t}+\frac{\partial \rho u}{\partial x}=0$$

Applying the discretization scheme

$$\frac{u_i^{k+1}-u_i^k}{\Delta t}+\frac{(\rho u)_{\,i+1}-(\rho u)_{i-1}}{2\Delta x}=0.$$

Solving for $u_i^{k+1}$ we get

$$u_i^{k+1}=u_i^k-\frac{\Delta t}{2\Delta x}[(\rho u)_{\,i+1}-(\rho u)_{i-1}]. \qquad \square$$

Problem:

Try this for yourself, do CTCS and notice that the solution is

$$u_i^{k+1}=u_i^{k-1}-\frac{\Delta t}{\Delta x}[(\rho u)_{\,i+1}-(\rho u)_{i-1}]$$

---

When implementing this scheme, users will observe that although FTCS is straightforward and second-order accurate in space, its relatively low numerical dissipation and first-order time accuracy may produce oscillations or weak solutions near steep gradients or discontinuities, reflecting the well-known conditional stability limitations of the method.

## Lax Friedrichs Method

The Lax Friedrichs method builds on the FTCS method by considering the average of the conserved quantity at the current position and the next position. At its core, the Lax-Friedrichs scheme combines two key ideas at each time step:

1. Averaging step
    a. It takes the average of the solution at neighboring points to introduce numerical diffusion.
2. Flux Difference
    a. It subtracts a scaled difference of fluxes at neighboring points to approximate the derivative.

In one dimension, for the above equation, the update formula at grid point $i$ and time level $k$ is:

$$u_i^{k+1} = \frac{(u_{i+1} + u_i)}{2} - \frac{\Delta t}{2\Delta x}[f(u_{i+1}) - f(u_i)].$$

The method is stable under the CFL condition $\frac{\Delta t}{\Delta x}\max|f'(u)| \leq 1$, where $\max|f'(u)|$ is the maximum wave speed. The averaging step introduces significant artificial viscosity, which smooths sharp gradients and prevents oscillations near discontinuities but also smears shock and reduces accuracy. It is only first order accuracy in both space and time, meaning errors decrease linearly with finer grids. The scheme is straightforward to implement and serves as a useful baseline for comparison with more advanced, higher order methods.

## Richtmyer Method

The Richtmyer method is a two-step, explicit, second-order accurate numerical scheme used to solve hyperbolic conservation laws. It is a variant of the Lax-Wendroff method but structured to separate the predictor and corrector steps for clarity and implementation ease. It's particularly well-suited for linear or nonlinear hyperbolic PDEs such as the Euler equations or simple advection equations. The method achieves second-order accuracy in both space and time, and it is conditionally stable under a CFL condition.

The steps to the algorithm are

1. Predictor Step

$$u_{i+\frac{1}{2}}^{k+\frac{1}{2}} = \frac{1}{2}\left(u_{i+1}^k + u_i^k\right) - \frac{\Delta t}{2\Delta x}(f\left(u_{i+1}^k\right) - f(u_i^k)$$

2. Corrector Step

$$u_i^{k+1} = u_i^k - \frac{\Delta t}{\Delta x}\left(f\left(u_{i+\frac{1}{2}}^{k+\frac{1}{2}}\right) - f\left(u_{i-\frac{1}{2}}^{k+\frac{1}{2}}\right)\right).$$

This method is conditionally stable under the CFL condition as well as being second order accurate in both time and space. This method offers significantly less numerical dissipation that Lax-

Friedrichs enabling sharper shock resolution. However, it may develop oscillations in regions of strong gradients.

## Riemann Problem

The exact Riemann solver we use is a numerical method used to solve the initial value problem

$$u(x,0) = \begin{Bmatrix} u_L, & x < 0 \\ u_R, & x > 0 \end{Bmatrix}$$

for a system of hyperbolic conservation laws. The goal of the solver is to determine the evolution of this discontinuity over time. This solver computes the exact solution to the Riemann problem for a given system of equations-typically the Euler equations in gas dynamics.

The solution to the Riemann problem consists of a combination of three elementary waves:

1. Shock waves
2. Rarefaction waves
3. Contact Discontinuities

These waves propagate at different speeds and separate constant states in space-time. The exact solution requires finding the structure and strength of these waves that satisfy both the conservation laws and the Rankine-Hugoniot conditions (for shocks) or Riemann invariants (for rarefactions).

The algorithm we use is as follows

1. Guess an initial estimate for pressure in the star region $p^*$
2. We then use a Newton-Raphson method to find $p^*$ that satisfies the matching conditions across both waves
3. Next, we compute the velocity in the star region $u^*$
4. Then, we use $p^*$ and $u^*$ to compute the states in each region separated by the waves
5. We then construct the piecewise solution across the
    a. Left Shock/Rarefaction
    b. Contact Discontinuity
    c. Right Shock/Rarefaction

This method is physically accurate with no approximations in wave speeds of states. However, it is computationally expensive requiring an iterative solver. The Exact Riemann solution captures all wave types and nonlinear interactions but implementation is complex for full systems (i.e. MHD, relativistic flows).