**README (Plain Text)**

**DV/RIPE SIMULATION FRAMEWORK**

A higher-dimensional, gauge-coupled field simulation for dual-vortex electrons and protons, featuring:

* Dimensionless Compton-based scaling
* Symplectic leapfrog PDE solver (fixed or adaptive)
* Refined potential with scalar couplings, gauge fields, and optional gravitational indentation

**DIRECTORY STRUCTURE**

DeepRIPE/ ┣ src/ ┃ ┣ constants.py (Defines fundamental physical constants & dimensionless conversions) ┃ ┣ fields.py (Field initialization routines for electron, gauge, gravity) ┃ ┣ pde\_solver.py (Leapfrog PDE solver, fixed-step & adaptive-step logic) ┃ ┣ diagnostics.py (Functions to measure spin, charge, gravitational indentation) ┃ ┣ simulation.py (Main simulation driver, calls PDE solver & diagnostics) ┣ tests/ ┃ ┗ test\_dvripe\_simulation.py ┣ drivers/ ┃ ┗ parameter\_search.py (Optional: scikit-optimize or custom parameter scanning) ┗ README.txt (This file)

**INSTALLATION**

1. Ensure Python 3.8+ is installed.
2. Install required libraries: pip install numpy scikit-optimize
3. (Optional) For speed, install numba or any other performance libraries you prefer.

**USAGE EXAMPLES**

1. Minimal Test Run: python -m tests.test\_dvripe\_simulation
   * Runs a short DV/RIPE simulation using placeholders or partial PDE logic
   * Prints final spin, charge, and gravity
2. Full Parameter Scan: python -m drivers.parameter\_search
   * Varies parameters (lambda, delta, etc.) to minimize a combined spin-charge-gravity error
   * You can customize search ranges in parameter\_search.py
3. Adaptive PDE Evolution:
   * In simulation.py, set "adaptive" = True in the params dictionary, e.g.:

params = { "field\_shape": (4,8,8,8), "gauge\_shape": (4,8,8,8), "grav\_shape": (8,8,8), "tau\_end": 2.0, "dt": 0.01, "dx": 1.0, "adaptive": True, "dt\_min": 1e-6, "dt\_max": 0.1, "err\_tolerance": 1e-3, "lambda\_e": 1.0, "v\_e": 1.0, "delta\_e": 0.5, "e\_gauge": 0.1 } spin, charge, grav = run\_dvripe\_sim(params)

**FILES & FUNCTIONS**

1. src/constants.py
   * Holds fundamental constants (HBAR, M\_E, C)
   * Dimensionless conversions (time\_to\_tau, energy\_to\_dimensionless, etc.)
2. src/fields.py
   * init\_electron\_fields, init\_gauge\_field, init\_gravity\_field
   * Creates initial arrays for electron/proton fields, gauge potential, gravitational potential
3. src/pde\_solver.py
   * leapfrog\_step (fixed-step PDE logic)
   * evolve\_fields (loops fixed-step)
   * evolve\_fields\_adaptive (two half-steps vs one full-step for local error, adjusts dt)
   * field\_strength (basic gauge field partial derivatives)
   * potential\_derivative (placeholder for refined potential, to be customized)
4. src/diagnostics.py
   * compute\_spin, compute\_charge, compute\_gravity\_indentation
   * You can replace or expand these for your final topological winding or flux integrals
5. src/simulation.py
   * run\_dvripe\_sim(params): orchestrates the entire run
   * Picks between fixed-step or adaptive PDE solver based on params["adaptive"]
   * Returns final spin, charge, gravity
6. tests/test\_dvripe\_simulation.py
   * A minimal script that calls run\_dvripe\_sim with some default parameters
7. drivers/parameter\_search.py (optional)
   * A scikit-optimize routine that scans for best parameters (lambda, delta, etc.) to minimize error

**POSSIBLE UPGRADES**

1. Proton Fields
   * Add separate phi\_p1, phi\_p2, PDE logic for different mass or coupling
2. Real Maxwell Update
   * Store electric, magnetic fields or A\_momentum for a second-order gauge leapfrog
3. Gravity Poisson Solver
   * If you want dynamic gravitational indentation, add a Poisson or Einstein solver in pde\_solver
4. Visualization
   * Scripts to visualize isosurfaces of phi1, phi2, gauge flux lines, etc.
5. HPC or GPU Acceleration
   * Use numba, cupy, or specialized HPC libraries to handle large Nx, Ny, Nz grids

**LICENSE STATEMENT**

DV/RIPE Simulation Framework Copyright (c) [Year], [Your Name]

You are free to download, use, change, modify, and upload this software, but I (the original author) retain intellectual ownership and must be credited as having used AI assistance to develop it. This code is provided “as is,” without any warranty. If you distribute derivative works, you must keep the attribution and intellectual ownership statements intact.