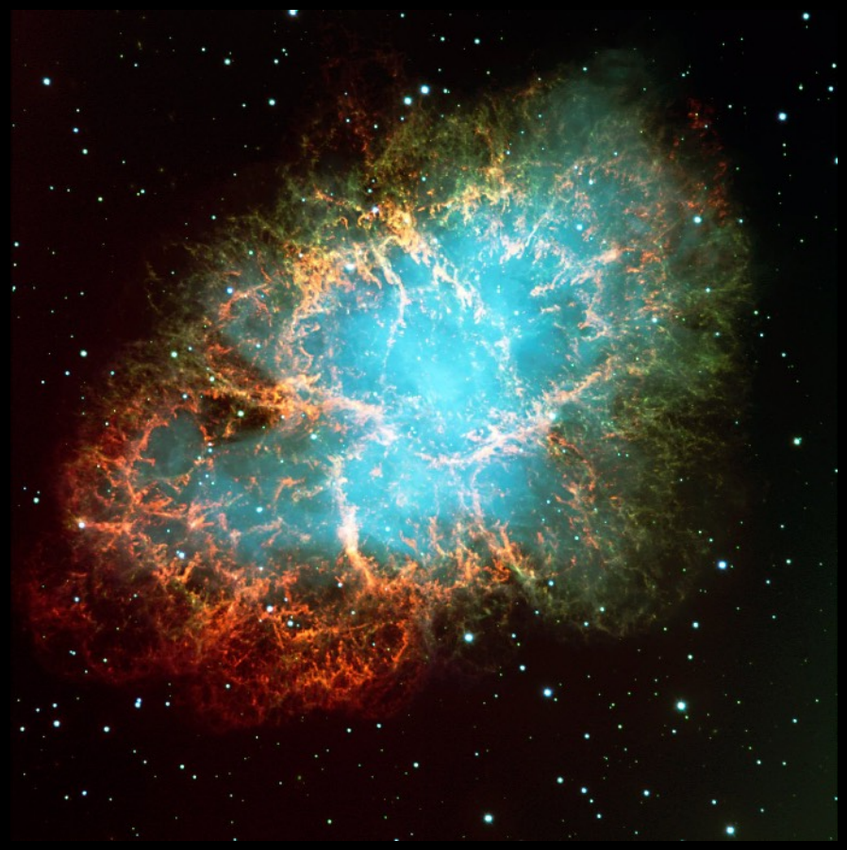


Super-Time-Stepping

P. D. Mullen
UIUC
Email: pmullen2@illinois.edu
GitHub: pdmullen

Explicit Integration:

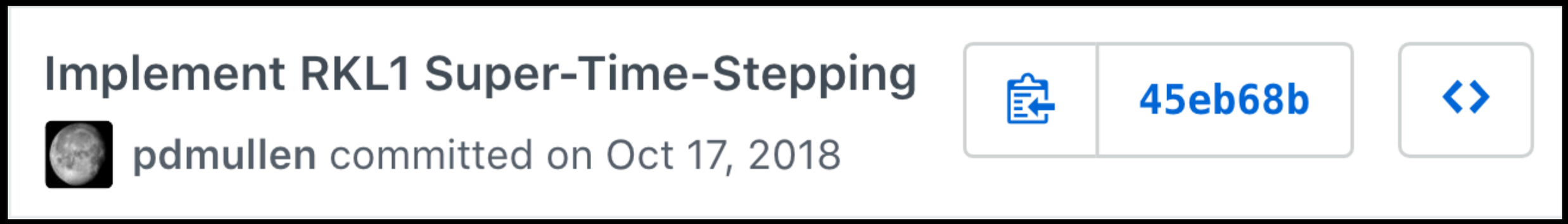
$$\Delta t_{\text{parabolic}} \propto \frac{\Delta x^2}{D} \propto \frac{1}{N^2 D}$$

For a 3-D problem, doubling the resolution yields:

-2^3 more zones (factor of 8 increase in cost)
-time-step reduced by 1/4 (factor of 4 increase in cost)

Total: ***32x more expensive*** *(!!!)*

Runge-Kutta Legendre (RKL) Super-Time-Stepping:




```
python configure.py –sts
```

(1) The scheme is formulated using Legendre polynomials as stability polynomials, coupled with their recurrence relations. For more details, see Meyer et al. (2014).
(2) Operator split diffusive physics.
(3) In operator split, execute an RKL super-time-step. The super-time-step is taken in *s*-stages.
(4) The scheme is explicit, i.e., the *j*-th stage only uses solution vectors from previous stages.

Meyer, C. D., Balsara, D. S., & Aslam, T. 2014, **JCP**, 257, 594

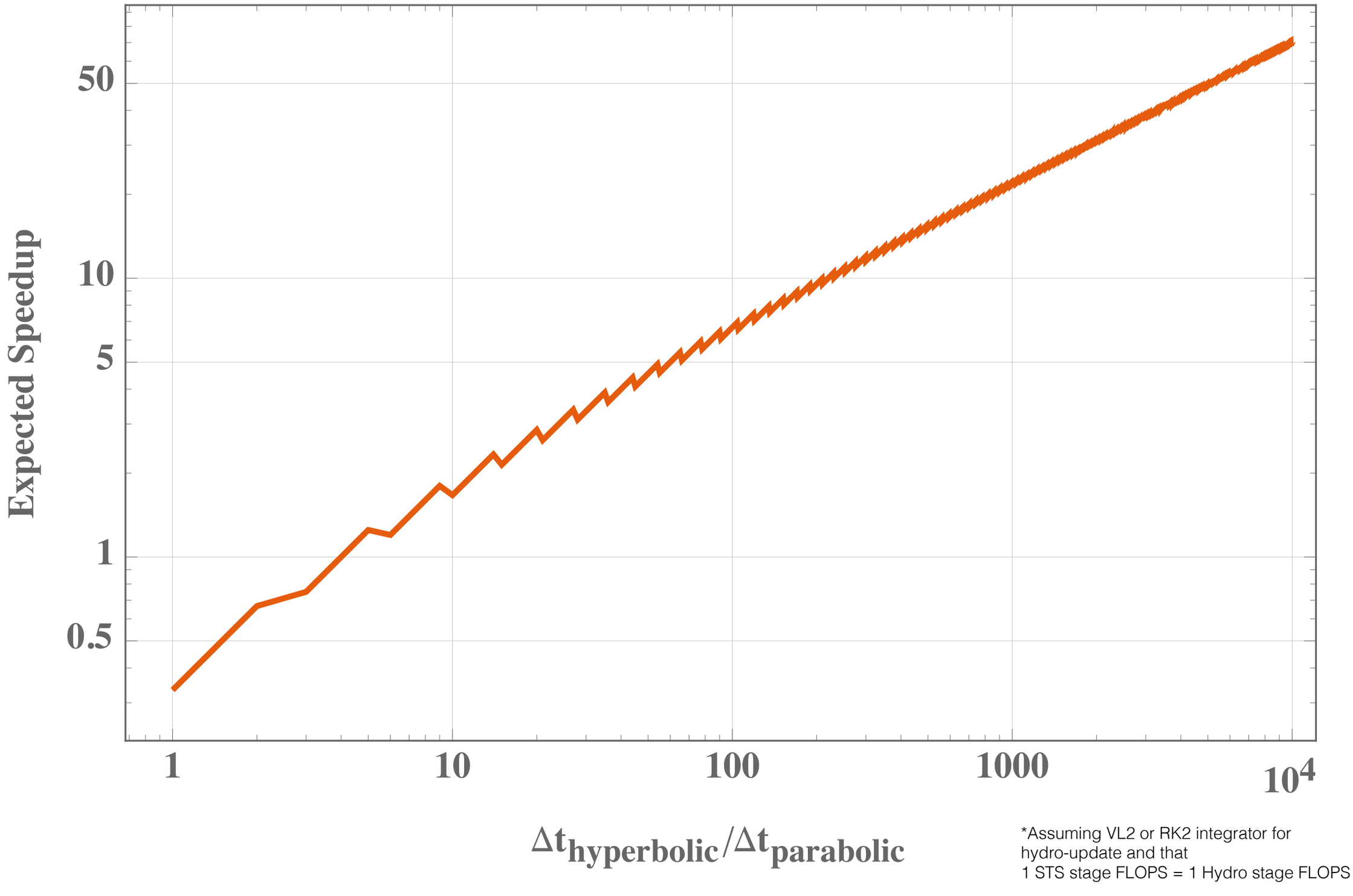
Why do all of this?

The RKL scheme allows us to take a time-step that is ~s^2 times larger than the parabolic time-step!

$$\Delta t_{\mathrm{STS}} = \Delta t_{\mathrm{parabolic}} \frac{s^2 + s}{2}$$

The desired super-time-step is the hyperbolic time-step, therefore, the ratio of the hyperbolic time-step to the parabolic time-step gives the number of stages.

$$s = \left[\frac{1}{2}\left(\sqrt{1 + 8\frac{\Delta t_{\text{hyperbolic}}}{\Delta t_{\text{parabolic}}}} - 1\right)\right] \implies \text{Speedup}^{*} \sim \frac{2\Delta t_{\text{hyperbolic}}/\Delta t_{\text{parabolic}}}{s+2}$$



*Assuming VL2 or RK2 integrator for hydro-update and that
1 STS stage FLOPS = 1 Hydro stage FLOPS

RKL1 Algorithm:

(1) Evaluate $\Delta t_{\rm hyperbolic}$ and $\Delta t_{\rm parabolic}$.

(2) Operator Split Diffusive Physics with RKL1 STS

(a) Compute number of stages, *s*.

$$s = \left[\frac{1}{2}\left(\sqrt{1 + 8\frac{\Delta t_{\rm hyperbolic}}{\Delta t_{\rm parabolic}}} - 1\right)\right]$$

(b) Cycle through the s-stages, where the solution vector for the *j*-th stage is (Meyer et al. 2014):

$$Y_j = \mu_j Y_{j-1} + \nu_j Y_{j-2} + \tilde{\mu}_j \Delta t_{\rm sts} \mathbf{M} Y_{j-1}$$

(3) Regular Hydro Update.

Implementation: Operator Split SuperTimeStepTaskList

```
425     if (STS_ENABLED) {
426       // compute nstages for this STS
427       Real my_dt = pmesh->dt;
428       Real dt_diff = pmesh->dt_diff;
429       pststlist->nstages = static_cast<int>(0.5*(-1.+std::sqrt(1.+8.*my_dt/dt_diff))) + 1;
430
431       // super-time-step
432       for (int stage=1; stage<=pststlist->nstages; ++stage)
433         pststlist->DoTaskListOneStage(pmesh,stage);
434     }
435
436     if (pmesh->turb_flag > 1) pmesh->ptrbd->Driving(); // driven turbulence
437
438     for (int stage=1; stage<=ptlist->nstages; ++stage) {
439       if (SELF_GRAVITY_ENABLED == 1) // fft (flag 0 for discrete kernel, 1 for continuous)
440         pmesh->pfgrd->Solve(stage, 0);
441       else if (SELF_GRAVITY_ENABLED == 2) // multigrid
442         pmesh->pmgrd->Solve(stage);
443       ptlist->DoTaskListOneStage(pmesh, stage);
444     }
445
446     pmesh->ncycle++;
447     pmesh->time += pmesh->dt;
```

- `pststlist` uses existing `u,u1,u2,b,b1,b2` registers, no need for more storage!

- Added Functions: `phydro->CalculateFluxes_STS()`, and `pfield->ComputeCornerE_STS()`

- `StartupTaskList()` computes μ_j, ν_j, and $\tilde{\mu}_j$. Applied with existing `WeightedAve*()` functions.

Testing:

The `diffusion` test suite has been extended to include `*_sts.py` tests.

Alarming (!!!):

Even though RKL1 is only a first-order scheme, `linear_wave3d` and `thermal_attenuation` tests still passed when run with super-time-stepping.

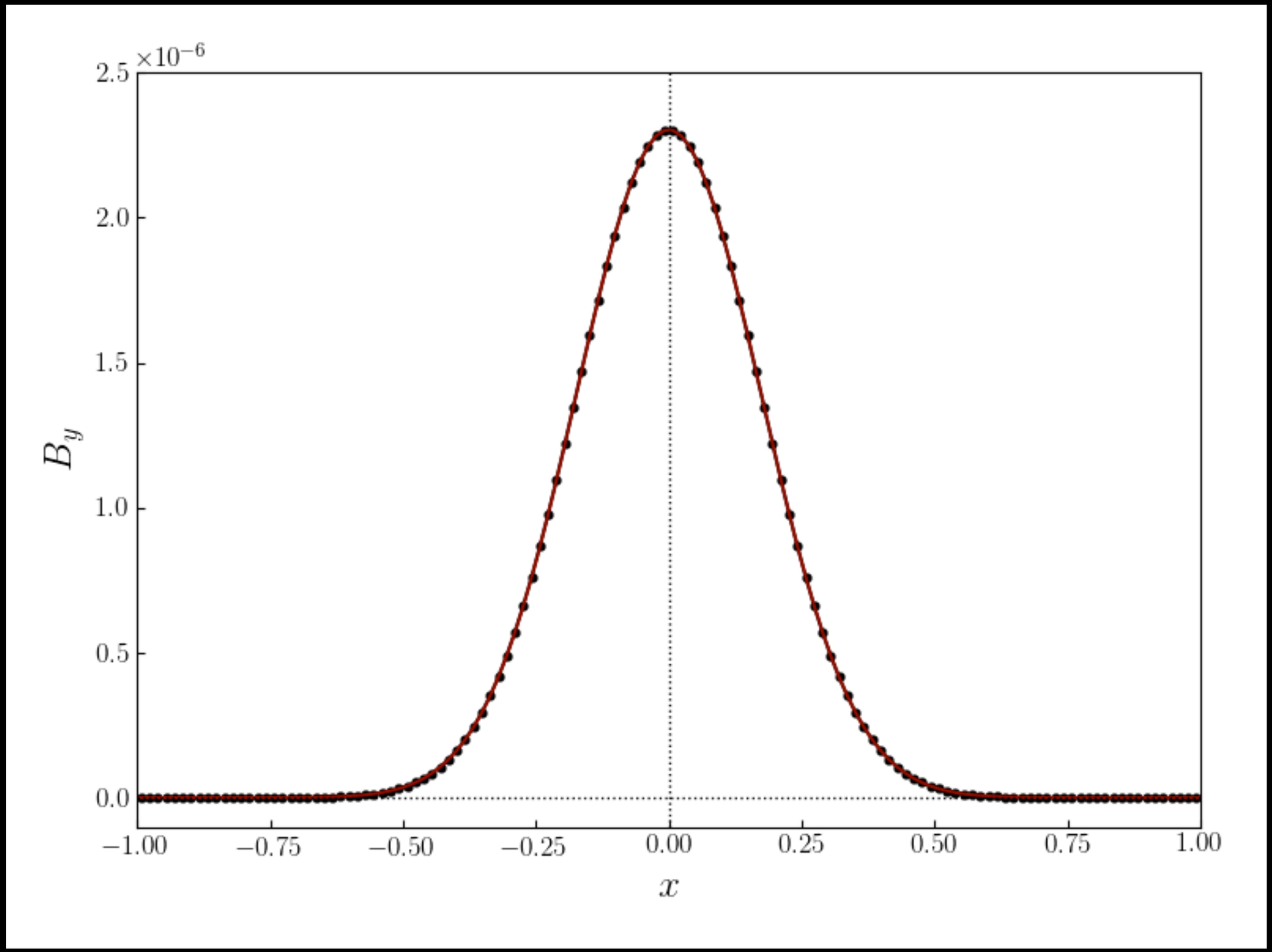
First Efforts Towards A Solution:

Added two convergence tests (for both explicit and STS!!!)

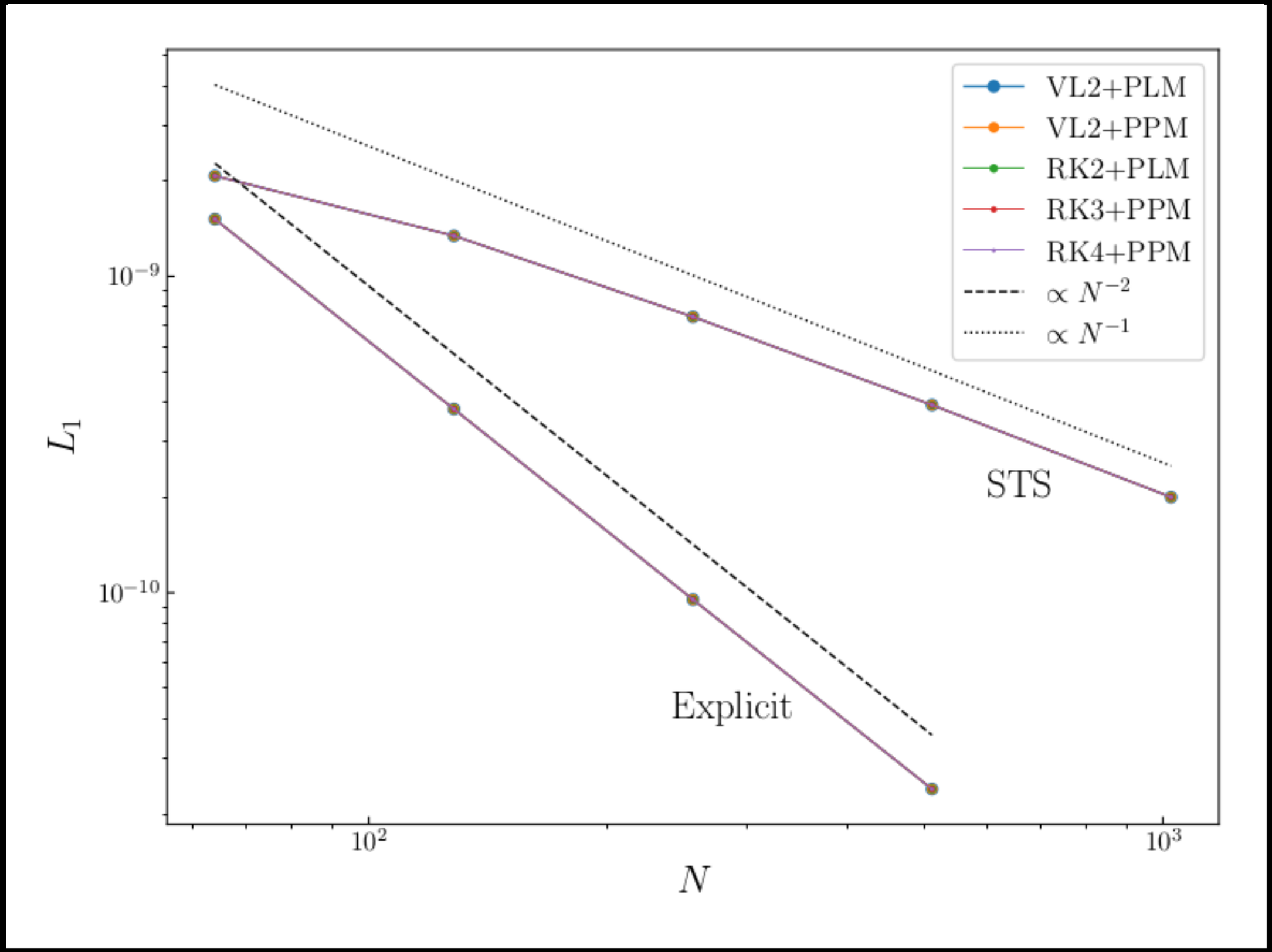
(1) Ohmic diffusion of a Gaussian B-field
   — `resistive_diffusion*.py`

(2) Viscous diffusion of a Gaussian velocity-field
   — `viscous_diffusion*.py`

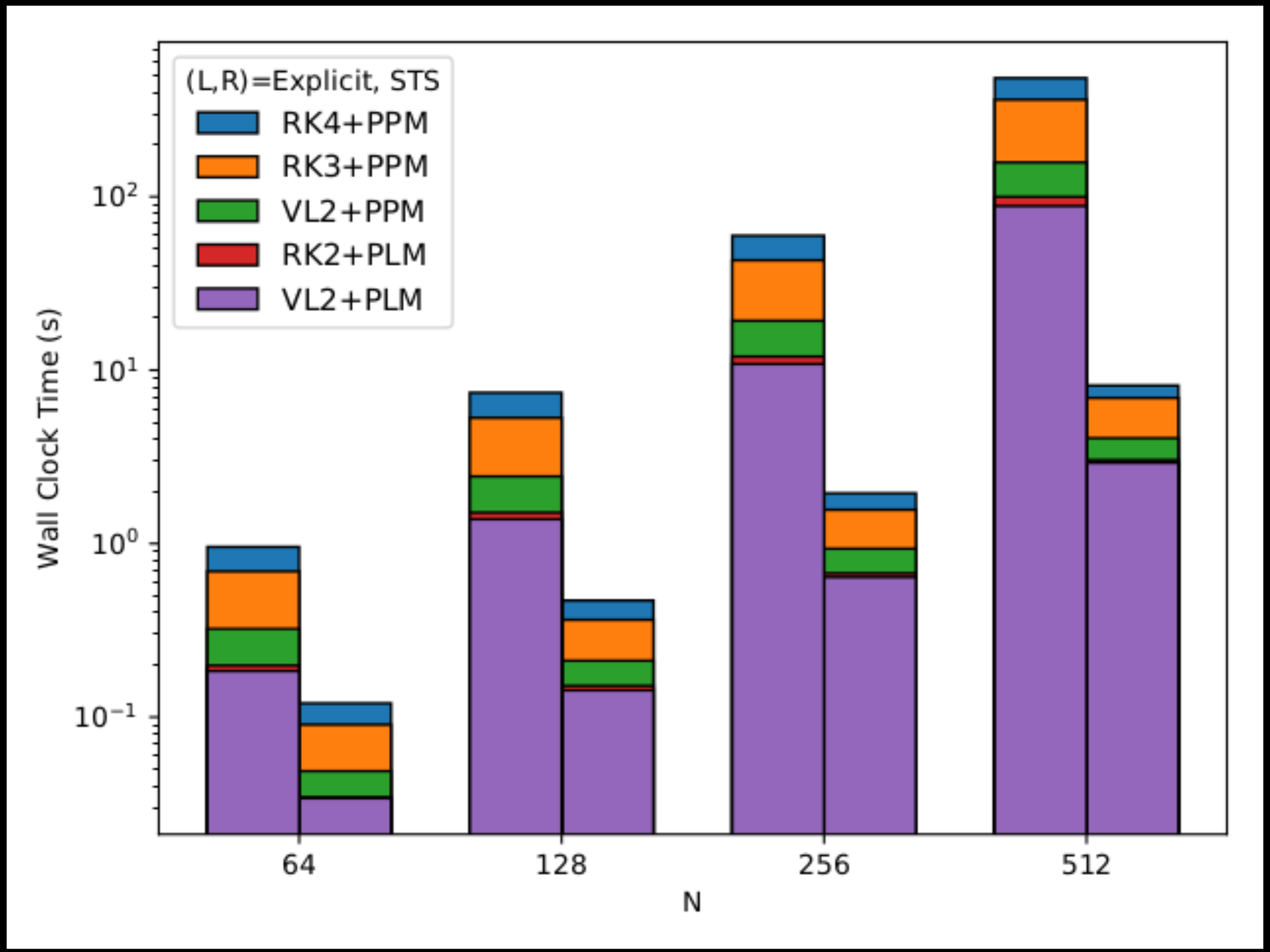
resistive_diffusion*.py



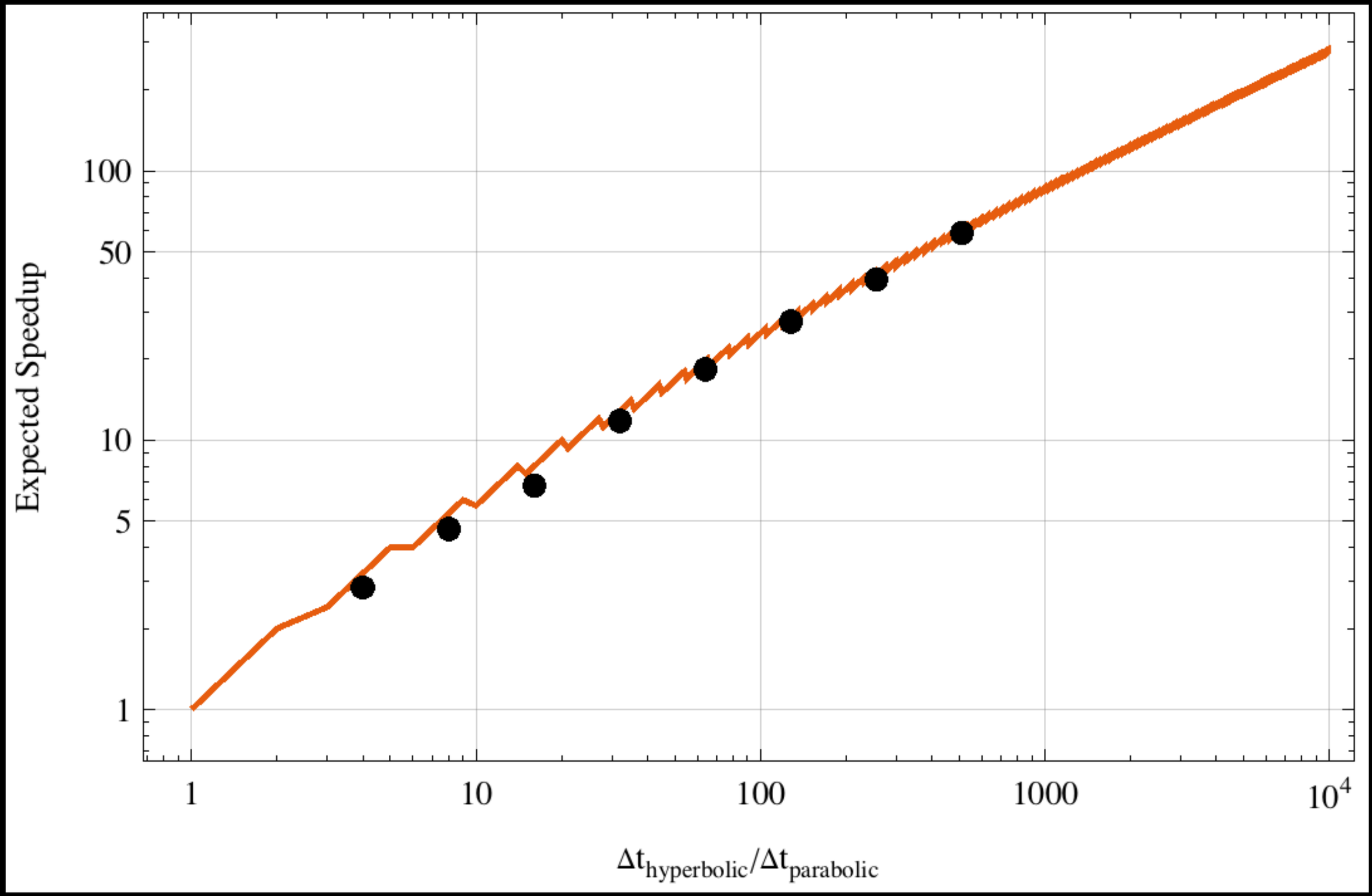
resistive_diffusion*.py



Performance:

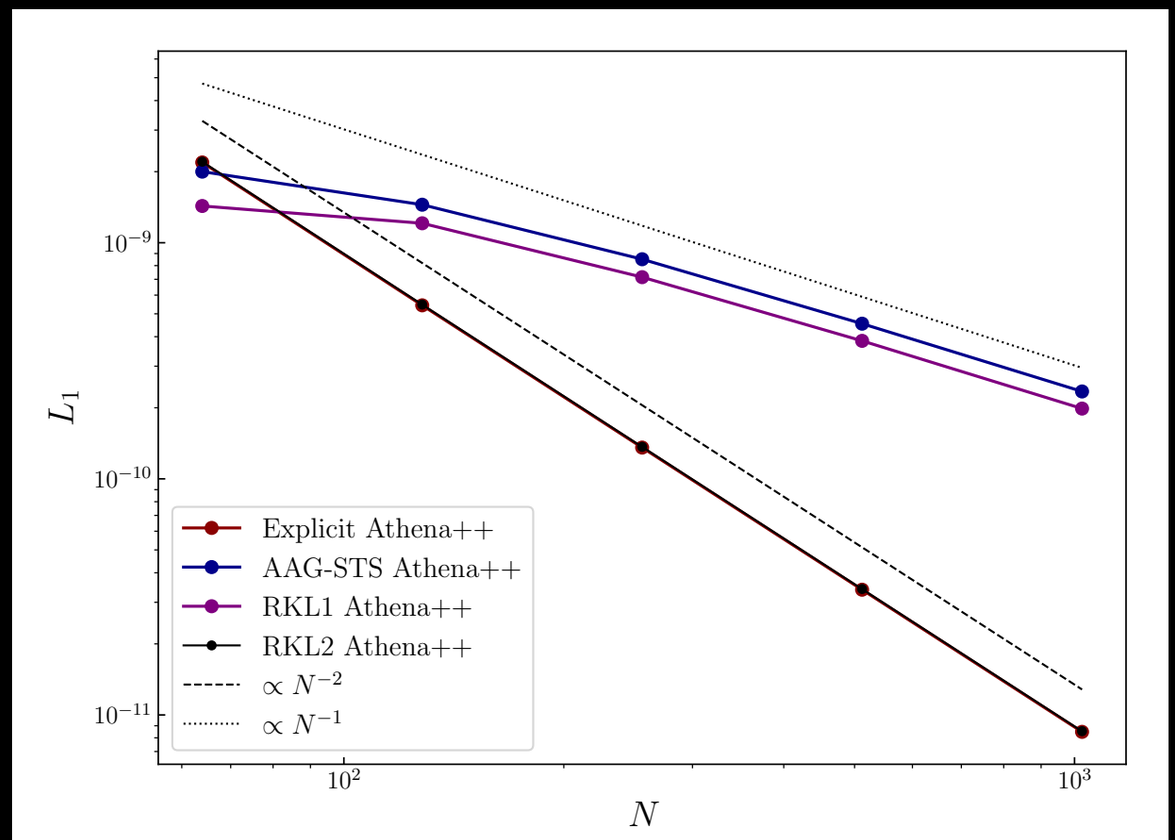


Performance:



Future Directions:

```
41    // STS Incompatiblities
42    if (MAGNETIC_FIELDS_ENABLED &&
43        !(pm->pblock->pfield->pfdif->field_diffusion_defined) &&
44        !(pm->pblock->phydro->phdif->hydro_diffusion_defined)) {
45      std::stringstream msg;
46      msg << "### FATAL ERROR in SuperTimeStepTaskList" << std::endl
47          << "Super-time-stepping requires setting parameters for "
48          << "diffusive processes in input file." << std::endl;
49      ATHENA_ERROR(msg);
50    }
51    // TODO(pdmullen): time-dep BC's require knowing the time within
52    //                 an RKL1 operator-split STS, what is the time?
53    if (SHEARING_BOX) {
54      std::stringstream msg;
55      msg << "### FATAL ERROR in SuperTimeStepTaskList" << std::endl
56          << "Super-time-stepping is not yet compatible "
57          << "with shearing box BC's." << std::endl;
58      ATHENA_ERROR(msg);
59    }
60    // TODO(pdmullen): how should source terms be handled inside
61    //                 operator-split RKL1 STS?
62    if (pm->pblock->phydro->psrc->hydro_sourceterms_defined==true) {
63      std::stringstream msg;
64      msg << "### FATAL ERROR in SuperTimeStepTaskList" << std::endl
65          << "Super-time-stepping is not yet compatible "
66          << "with source terms." << std::endl;
67      ATHENA_ERROR(msg);
68    }
69    // TODO(pdmullen): fix non-Cartesian compatibility; this requires the
70    //                 handling of coordinate source terms.
71    if (std::strcmp(COORDINATE_SYSTEM, "cartesian") != 0) {
72      std::stringstream msg;
73      msg << "### FATAL ERROR in SuperTimeStepTaskList" << std::endl
74          << "Super-time-stepping is not yet compatibile "
75          << "with non-Cartesian coordinates." << std::endl;
76      ATHENA_ERROR(msg);
77    }
78    // TODO(pdmullen): add mesh-refinement functionality
79    if (pm->multilevel==true) {
80      std::stringstream msg;
81      msg << "### FATAL ERROR in SuperTimeStepTaskList" << std::endl
82          << "Super-time-stepping is not yet compatibile "
83          << "with mesh refinement." << std::endl;
84      ATHENA_ERROR(msg);
85    }
```



(1) Source Terms
(2) Curvilinear Coordinates*
(3) Shearing Box BCs
(4) AMR
(5) RKL2 & AAG-STS (?)
(6) Stage outputs?

*would yield huge increase in code coverage