Computational efficiency of high-order finite volume methods for magnetohydrodynamics

Kyle Gerard Felker

Princeton University

Arlington, VA

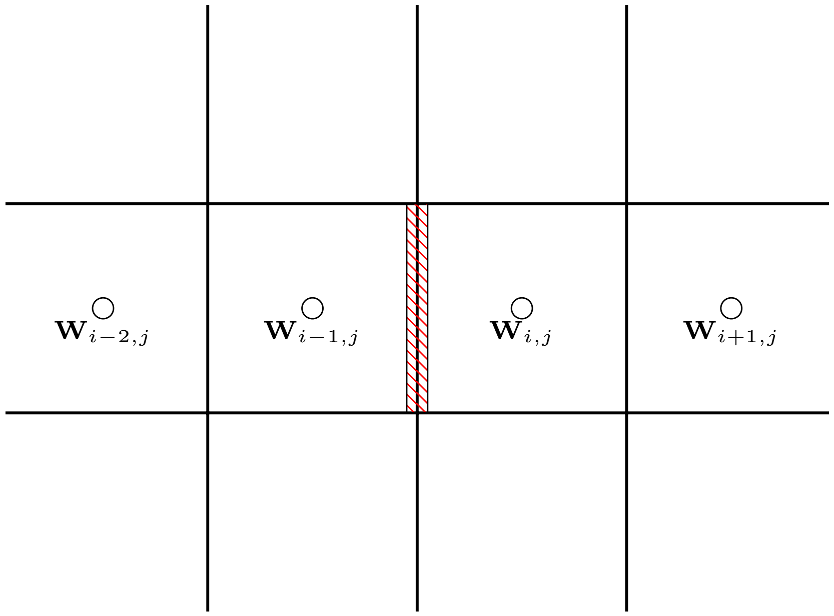
CSGF Annual Program Review: July 16, 2018

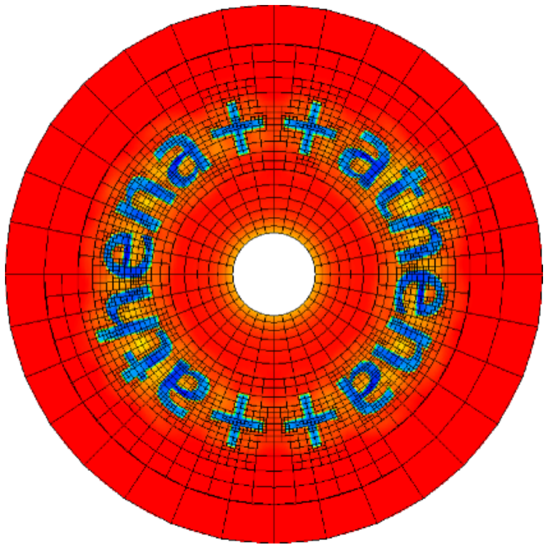
Motivation: computational magnetohydrodynamics (MHD)

- Accretion disk dynamics
  - Need to resolve fine-scale turbulence
  - Magnetic fields are crucial
- Major concerns for numerical methods for astrophysics:
  - Conserve relevant physical quantities to machine precision
  - Highly accurate without introducing artificial oscillations
  - High efficiency
  - Satisfy divergence constraint

$$\nabla \cdot \mathbf{B} = 0$$

Second-order finite volume (FV) method



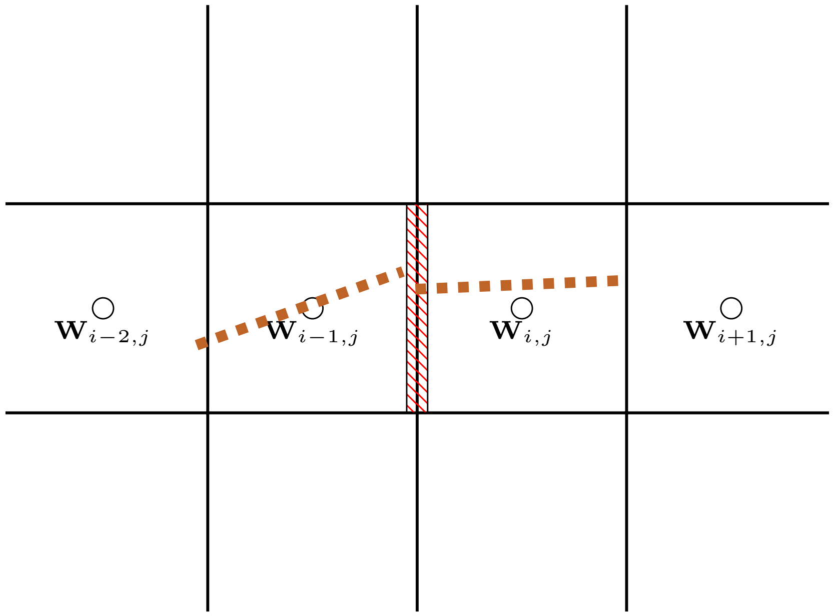


$\mathcal{O}(\Delta x^2)$

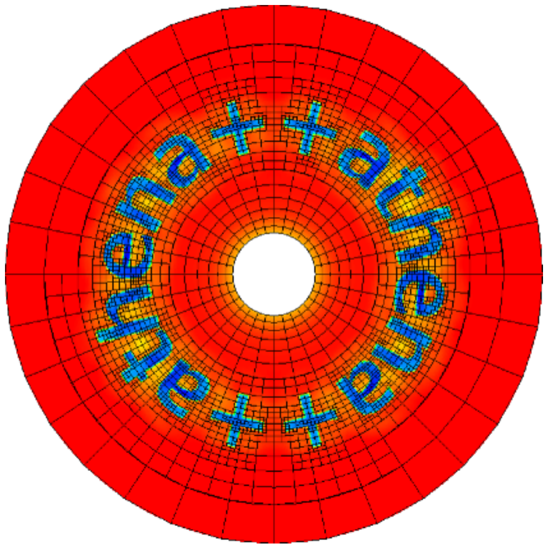
Second-order finite volume (FV) method

1. Reconstruct face-averaged primitive L/R states using Piecewise Linear Method (PLM)

$$\langle \mathbf{W}^{L_1/R_1} \rangle_{i-\frac{1}{2},j} = PLM(\{\langle \mathbf{W} \rangle_{i,j}, \langle \mathbf{W} \rangle_{i\pm1,j}, \langle \mathbf{W} \rangle_{i-2,j}\})$$



$$\mathcal{O}(\Delta x^2)$$



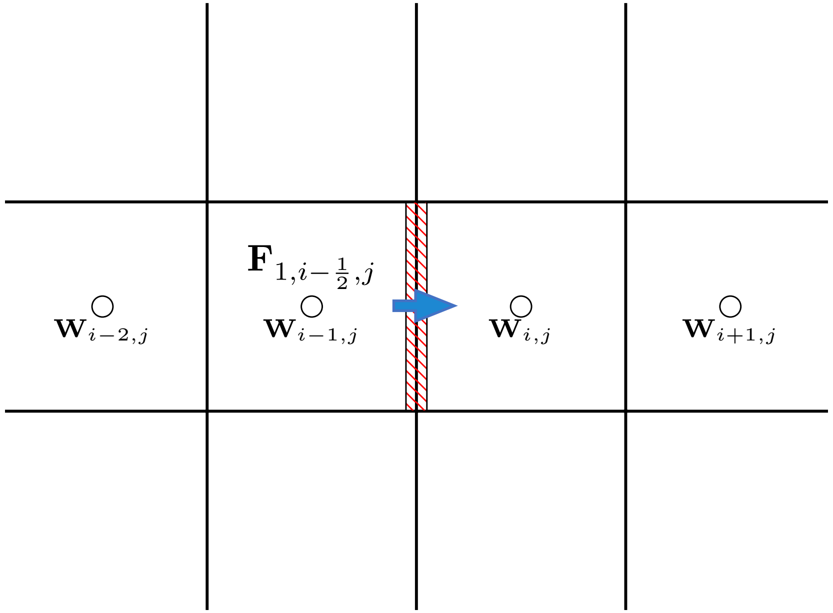
Second-order finite volume (FV) method

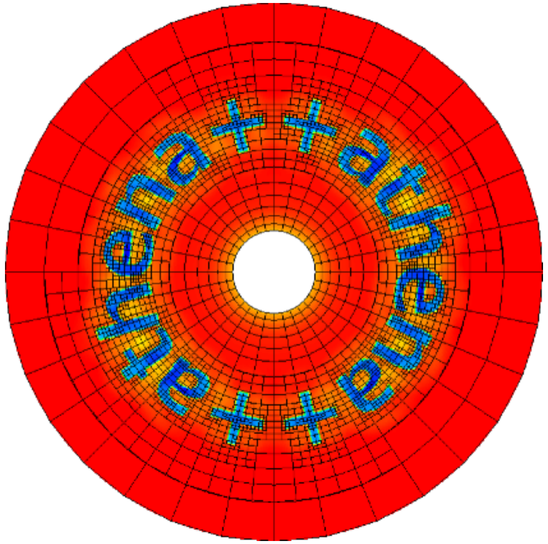
1. Reconstruct face-averaged primitive L/R states using Piecewise Linear Method (PLM)

$$\langle \mathbf{W}^{L_1/R_1} \rangle_{i-\frac{1}{2},j} = PLM(\{\langle \mathbf{W} \rangle_{i,j}, \langle \mathbf{W} \rangle_{i\pm1,j}, \langle \mathbf{W} \rangle_{i-2,j}\})$$

3. Using a Riemann solver $\mathscr{F}()$, compute pointwise face-centered fluxes from face-centered primitive states

$$\mathbf{F}_{1,i-\frac{1}{2},j} = \mathscr{F}(\mathbf{W}^{L_1}_{i-\frac{1}{2},j}, \mathbf{W}^{R_1}_{i-\frac{1}{2},j})$$





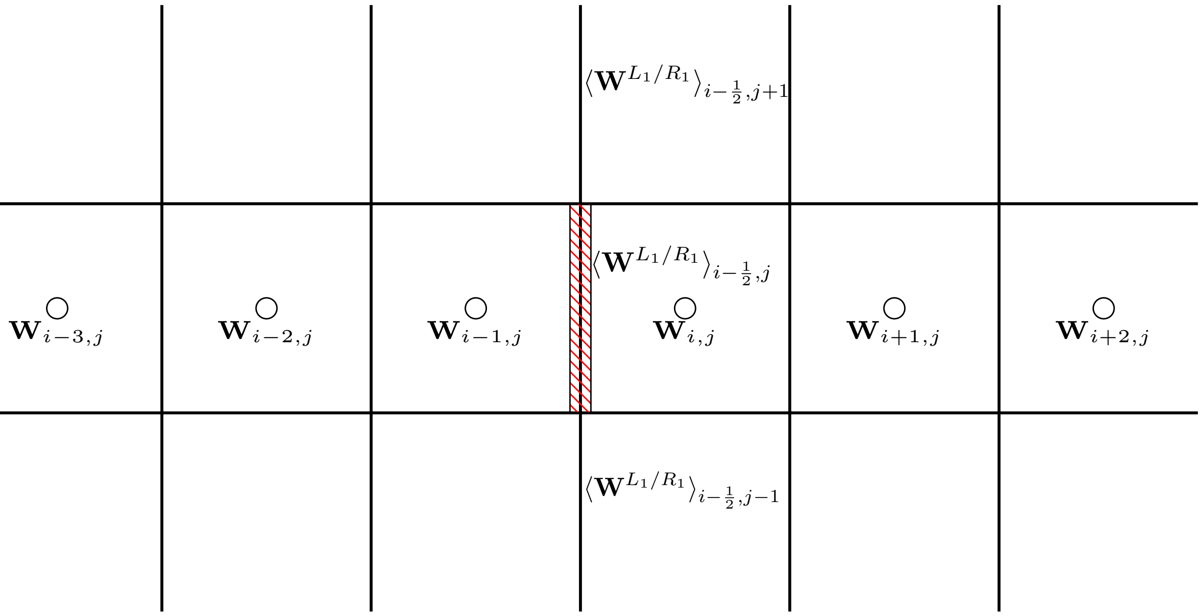
$$\mathcal{O}(\Delta x^2)$$

Fourth-order finite volume method for conservation laws

(McCorquodale & Colella 2011), (Colella+ 2009), (Guzik+ 2015)

1. Reconstruct face-averaged primitive L/R states using Piecewise Parabolic Method (PPM)

$$\langle \mathbf{W}^{L_1/R_1} \rangle_{i-\frac{1}{2},j} = PPM(\{\langle \mathbf{W} \rangle_{i,j}, \langle \mathbf{W} \rangle_{i\pm1,j}, \langle \mathbf{W} \rangle_{i\pm2,j}, \langle \mathbf{W} \rangle_{i-3,j}\})$$



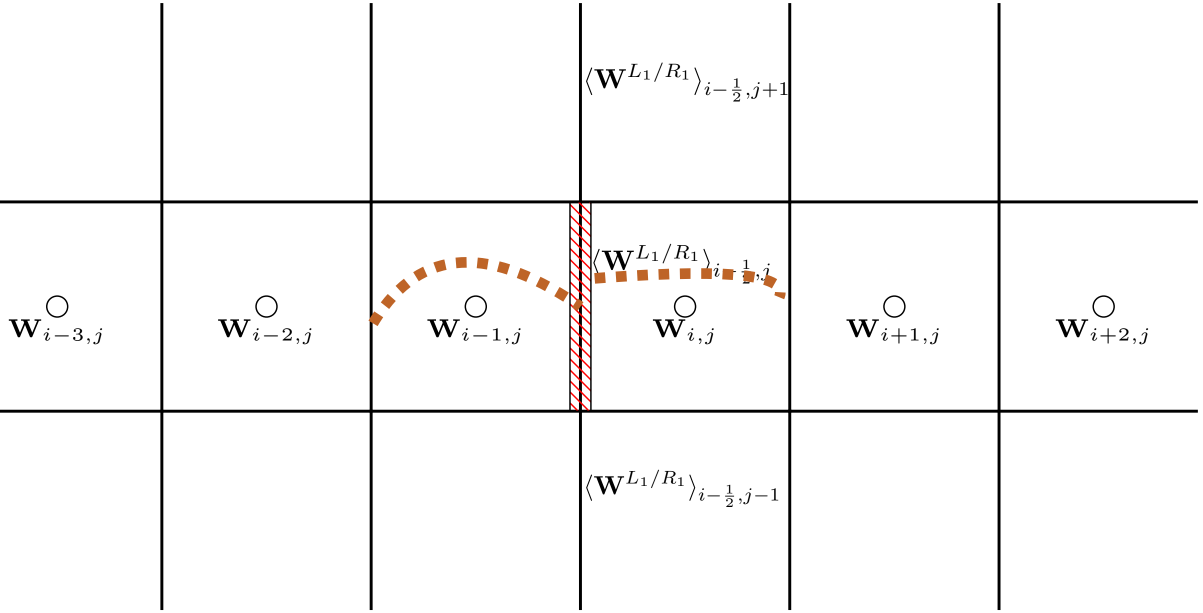
$$\mathcal{O}(\Delta x^4)$$

Fourth-order finite volume method for conservation laws

(McCorquodale & Colella 2011), (Colella+ 2009), (Guzik+ 2015)

1. Reconstruct face-averaged primitive L/R states using Piecewise Parabolic Method (PPM)

$$\langle \mathbf{W}^{L_1/R_1} \rangle_{i-\frac{1}{2},j} = PPM(\{\langle \mathbf{W} \rangle_{i,j}, \langle \mathbf{W} \rangle_{i\pm1,j}, \langle \mathbf{W} \rangle_{i\pm2,j}, \langle \mathbf{W} \rangle_{i-3,j}\})$$



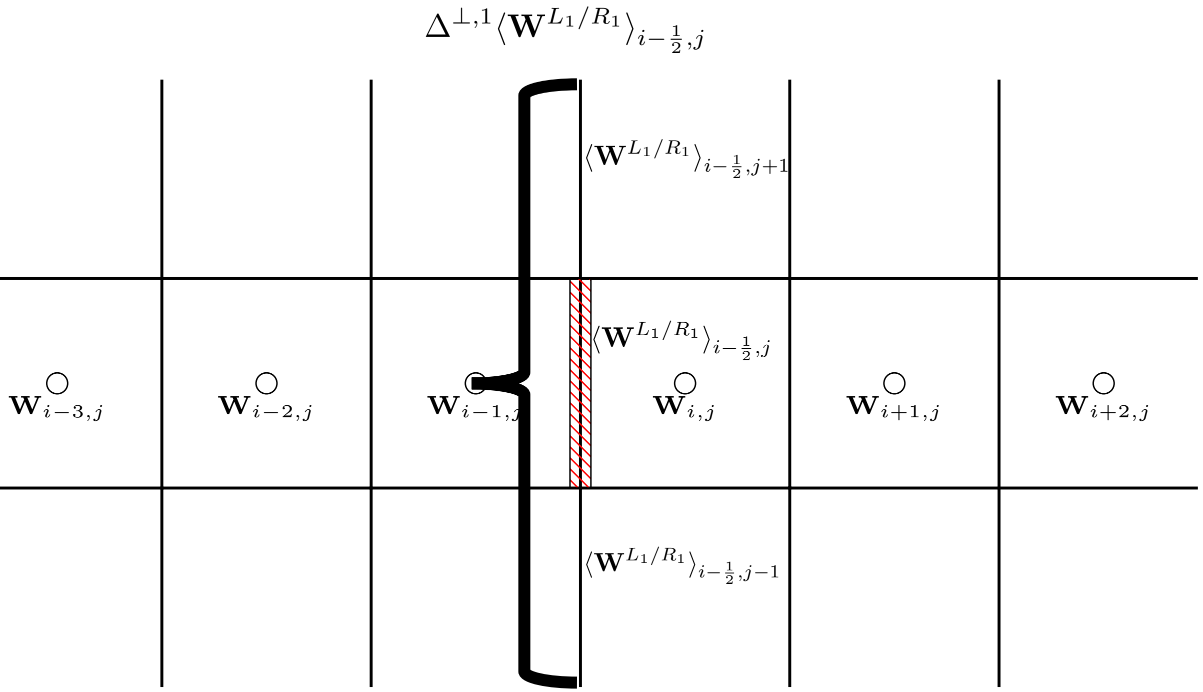
$$\mathcal{O}(\Delta x^4)$$

Fourth-order finite volume method for conservation laws

(McCorquodale & Colella 2011), (Colella+ 2009), (Guzik+ 2015)

1. Reconstruct face-averaged primitive L/R states using Piecewise Parabolic Method (PPM)

$$\langle \mathbf{W}^{L_1/R_1} \rangle_{i-\frac{1}{2},j} = PPM(\{\langle \mathbf{W} \rangle_{i,j}, \langle \mathbf{W} \rangle_{i\pm1,j}, \langle \mathbf{W} \rangle_{i\pm2,j}, \langle \mathbf{W} \rangle_{i-3,j}\})$$



$$\mathcal{O}(\Delta x^4)$$

Fourth-order finite volume method for conservation laws

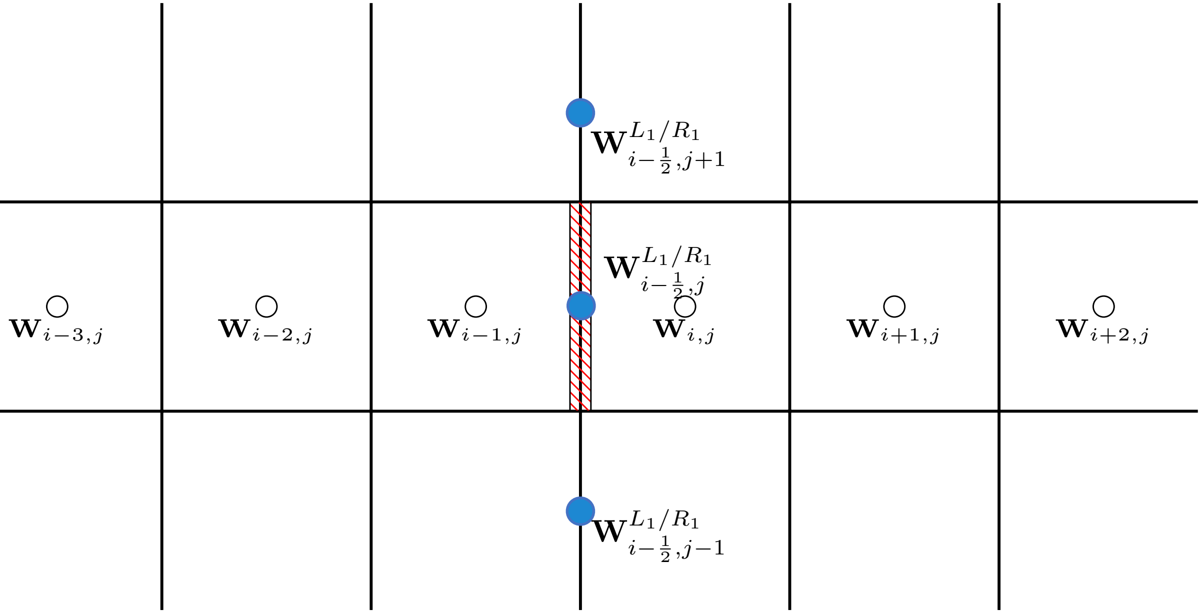
(McCorquodale & Colella 2011), (Colella+ 2009), (Guzik+ 2015)

1. Reconstruct face-averaged primitive L/R states using Piecewise Parabolic Method (PPM)

$$\langle \mathbf{W}^{L_1/R_1} \rangle_{i-\frac{1}{2},j} = PPM(\{\langle \mathbf{W} \rangle_{i,j}, \langle \mathbf{W} \rangle_{i\pm1,j}, \langle \mathbf{W} \rangle_{i\pm2,j}, \langle \mathbf{W} \rangle_{i-3,j}\})$$

2. Compute pointwise face-centered L/R states

$$\mathbf{W}^{L_1/R_1}_{i-\frac{1}{2},j} = \langle \mathbf{W}^{L_1/R_1} \rangle_{i-\frac{1}{2},j} - \frac{h^2}{24} \Delta^{\perp,1} \langle \mathbf{W}^{L_1/R_1} \rangle_{i-\frac{1}{2},j}$$



$$\mathcal{O}(\Delta x^4)$$

Fourth-order finite volume method for conservation laws

(McCorquodale & Colella 2011), (Colella+ 2009), (Guzik+ 2015)

1. Reconstruct face-averaged primitive L/R states using Piecewise Parabolic Method (PPM)

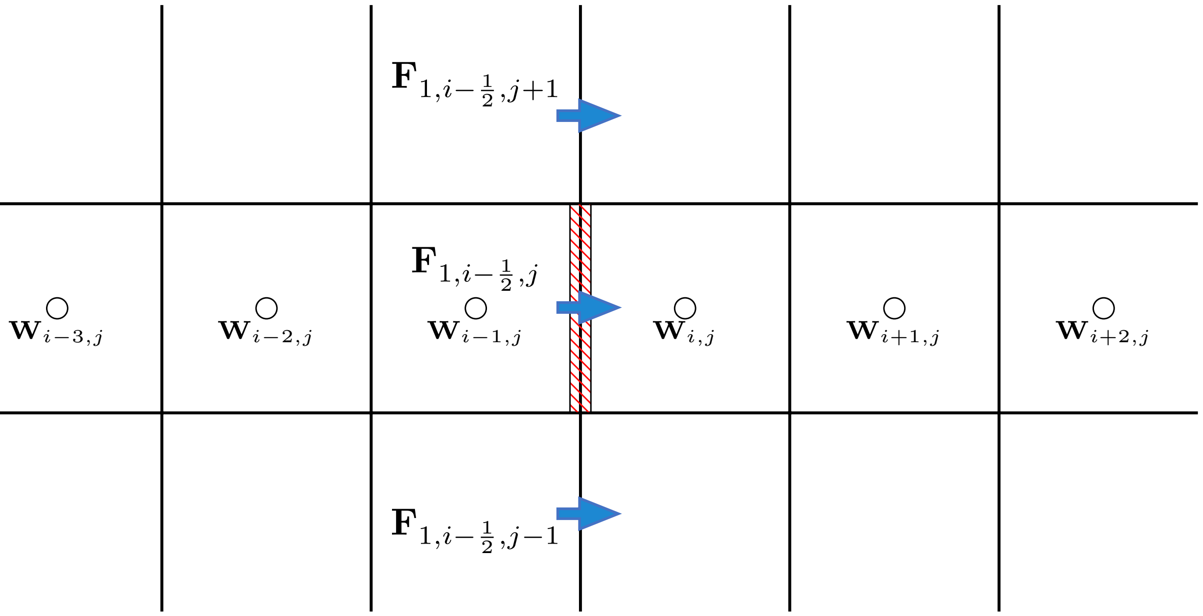
$$\langle \mathbf{W}^{L_1/R_1} \rangle_{i-\frac{1}{2},j} = PPM(\{\langle \mathbf{W} \rangle_{i,j}, \langle \mathbf{W} \rangle_{i\pm1,j}, \langle \mathbf{W} \rangle_{i\pm2,j}, \langle \mathbf{W} \rangle_{i-3,j}\})$$

2. Compute pointwise face-centered L/R states

$$\mathbf{W}^{L_1/R_1}_{i-\frac{1}{2},j} = \langle \mathbf{W}^{L_1/R_1} \rangle_{i-\frac{1}{2},j} - \frac{h^2}{24} \Delta^{\perp,1} \langle \mathbf{W}^{L_1/R_1} \rangle_{i-\frac{1}{2},j}$$

3. Using a Riemann solver $\mathscr{F}()$, compute pointwise face-centered fluxes from face-centered primitive states

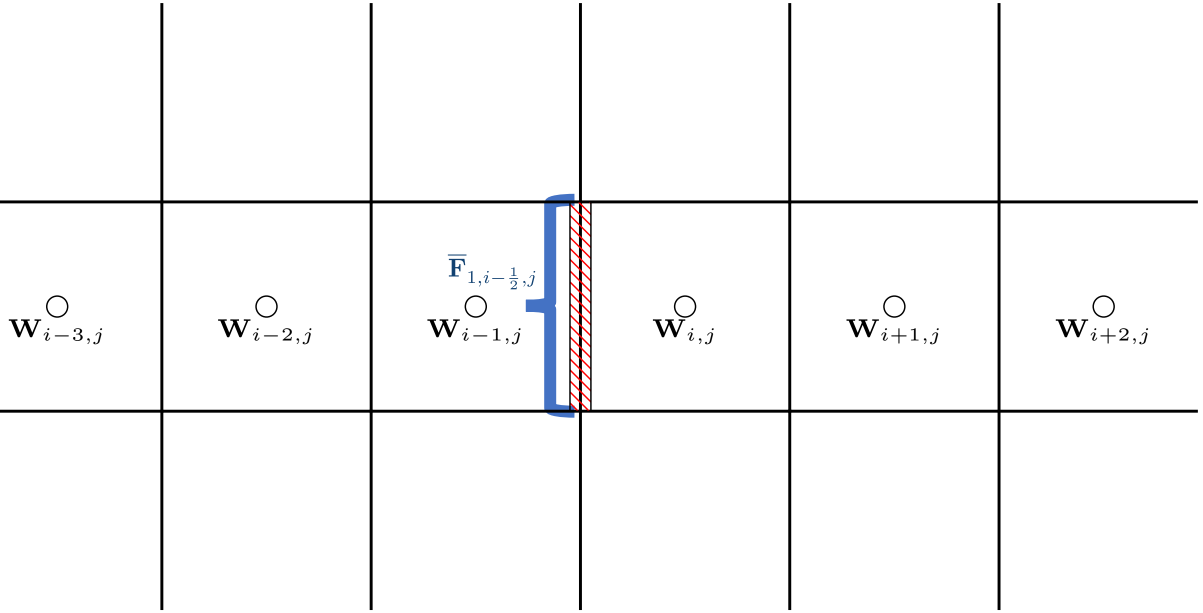
$$\mathbf{F}_{1,i-\frac{1}{2},j} = \mathscr{F}(\mathbf{W}^{L_1}_{i-\frac{1}{2},j}, \mathbf{W}^{R_1}_{i-\frac{1}{2},j})$$



$$\mathcal{O}(\Delta x^4)$$

Fourth-order finite volume method for conservation laws

(McCorquodale & Colella 2011), (Colella+ 2009), (Guzik+ 2015)



$$\mathcal{O}(\Delta x^4)$$

1. Reconstruct face-averaged primitive L/R states using Piecewise Parabolic Method (PPM)

$$\langle \mathbf{W}^{L_1/R_1} \rangle_{i-\frac{1}{2},j} = PPM(\{\langle \mathbf{W} \rangle_{i,j}, \langle \mathbf{W} \rangle_{i\pm1,j}, \langle \mathbf{W} \rangle_{i\pm2,j}, \langle \mathbf{W} \rangle_{i-3,j}\})$$

2. Compute pointwise face-centered L/R states

$$\mathbf{W}^{L_1/R_1}_{i-\frac{1}{2},j} = \langle \mathbf{W}^{L_1/R_1} \rangle_{i-\frac{1}{2},j} - \frac{h^2}{24} \Delta^{\perp,1} \langle \mathbf{W}^{L_1/R_1} \rangle_{i-\frac{1}{2},j}$$

3. Using a Riemann solver $\mathscr{F}()$, compute pointwise face-centered fluxes from face-centered primitive states

$$\mathbf{F}_{1,i-\frac{1}{2},j} = \mathscr{F}(\mathbf{W}^{L_1}_{i-\frac{1}{2},j}, \mathbf{W}^{R_1}_{i-\frac{1}{2},j})$$

4. Compute fluxes from the fourth-order accurate face-averaged primitive states → $\mathcal{O}(\Delta x^2)$ flux approx.

$$\overline{\mathbf{F}}_{1,i-\frac{1}{2},j} = \mathscr{F}(\langle \mathbf{W}^{L_1} \rangle_{i-\frac{1}{2},j}, \langle \mathbf{W}^{R_1} \rangle_{i-\frac{1}{2},j})$$

Fourth-order finite volume method for conservation laws

(McCorquodale & Colella 2011), (Colella+ 2009), (Guzik+ 2015)

1. Reconstruct face-averaged primitive L/R states using Piecewise Parabolic Method (PPM)

$$\langle \mathbf{W}^{L_1/R_1} \rangle_{i-\frac{1}{2},j} = PPM(\{\langle \mathbf{W} \rangle_{i,j}, \langle \mathbf{W} \rangle_{i\pm1,j}, \langle \mathbf{W} \rangle_{i\pm2,j}, \langle \mathbf{W} \rangle_{i-3,j}\})$$

2. Compute pointwise face-centered L/R states

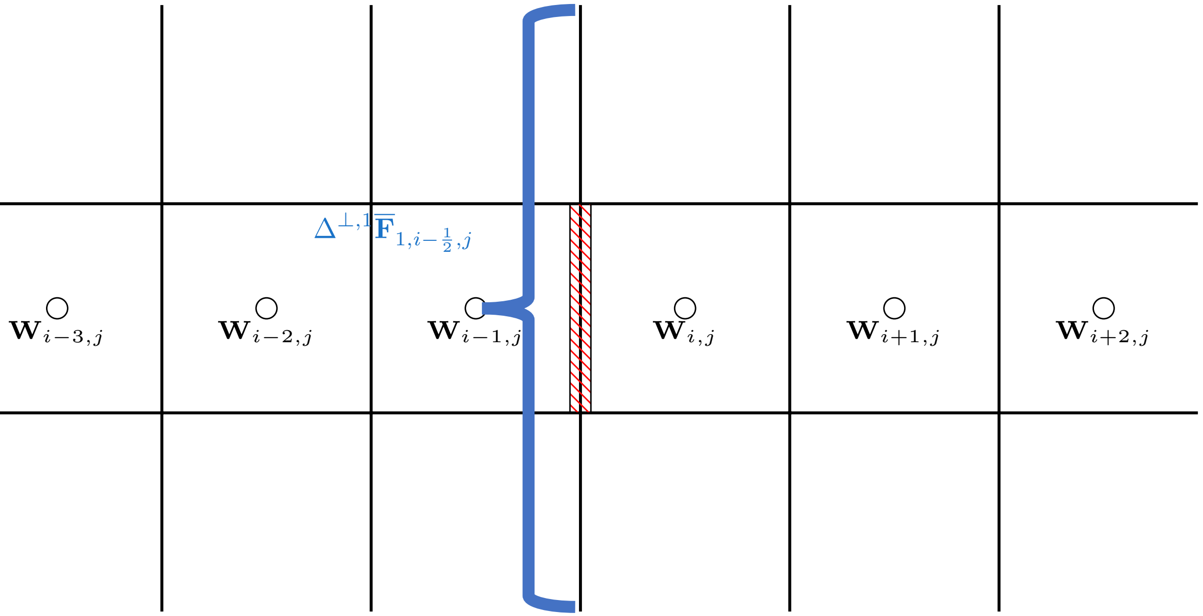
$$\mathbf{W}^{L_1/R_1}_{i-\frac{1}{2},j} = \langle \mathbf{W}^{L_1/R_1} \rangle_{i-\frac{1}{2},j} - \frac{h^2}{24} \Delta^{\perp,1} \langle \mathbf{W}^{L_1/R_1} \rangle_{i-\frac{1}{2},j}$$

3. Using a Riemann solver $\mathscr{F}()$, compute pointwise face-centered fluxes from face-centered primitive states

$$\mathbf{F}_{1,i-\frac{1}{2},j} = \mathscr{F}(\mathbf{W}^{L_1}_{i-\frac{1}{2},j}, \mathbf{W}^{R_1}_{i-\frac{1}{2},j})$$

4. Compute fluxes from the fourth-order accurate face-averaged primitive states → $\mathcal{O}(\Delta x^2)$ flux approx.

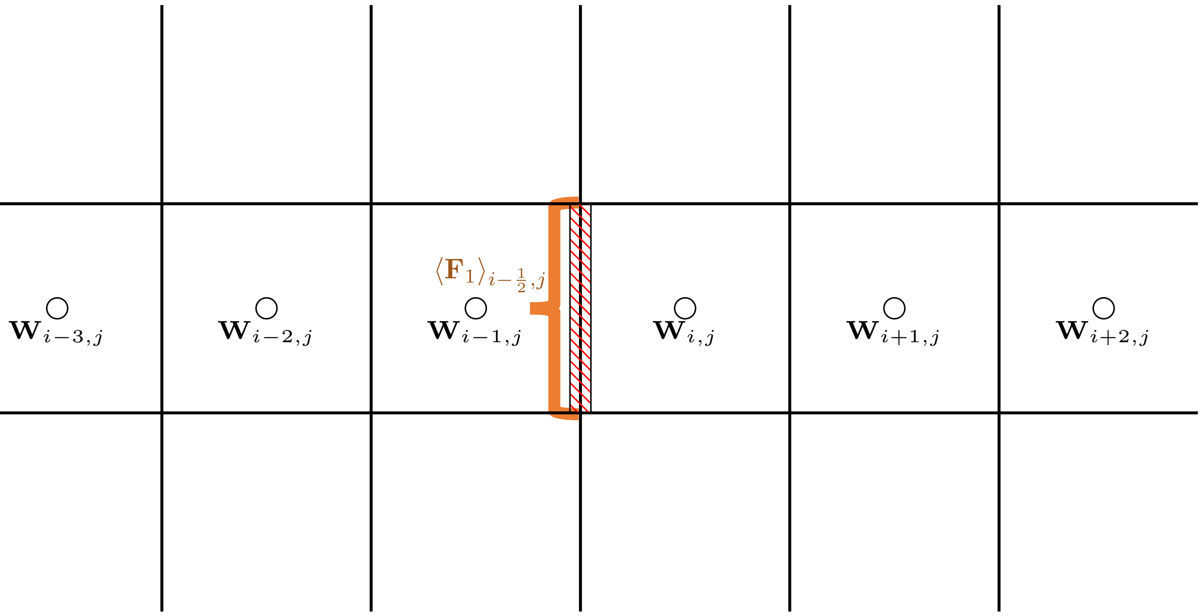
$$\overline{\mathbf{F}}_{1,i-\frac{1}{2},j} = \mathscr{F}(\langle \mathbf{W}^{L_1} \rangle_{i-\frac{1}{2},j}, \langle \mathbf{W}^{R_1} \rangle_{i-\frac{1}{2},j})$$



$$\mathcal{O}(\Delta x^4)$$

Fourth-order finite volume method for conservation laws

(McCorquodale & Colella 2011), (Colella+ 2009), (Guzik+ 2015)



$$\mathcal{O}(\Delta x^4)$$

1. Reconstruct face-averaged primitive L/R states using Piecewise Parabolic Method (PPM)

$$\langle \mathbf{W}^{L_1/R_1} \rangle_{i-\frac{1}{2},j} = PPM(\{\langle \mathbf{W} \rangle_{i,j}, \langle \mathbf{W} \rangle_{i\pm1,j}, \langle \mathbf{W} \rangle_{i\pm2,j}, \langle \mathbf{W} \rangle_{i-3,j}\})$$

2. Compute pointwise face-centered L/R states

$$\mathbf{W}^{L_1/R_1}_{i-\frac{1}{2},j} = \langle \mathbf{W}^{L_1/R_1} \rangle_{i-\frac{1}{2},j} - \frac{h^2}{24} \Delta^{\perp,1} \langle \mathbf{W}^{L_1/R_1} \rangle_{i-\frac{1}{2},j}$$

3. Using a Riemann solver $\mathscr{F}()$, compute pointwise face-centered fluxes from face-centered primitive states

$$\mathbf{F}_{1,i-\frac{1}{2},j} = \mathscr{F}(\mathbf{W}^{L_1}_{i-\frac{1}{2},j}, \mathbf{W}^{R_1}_{i-\frac{1}{2},j})$$

4. Compute fluxes from the fourth-order accurate face-averaged primitive states ➡ $\mathcal{O}(\Delta x^2)$ flux approx.

$$\overline{\mathbf{F}}_{1,i-\frac{1}{2},j} = \mathscr{F}(\langle \mathbf{W}^{L_1} \rangle_{i-\frac{1}{2},j}, \langle \mathbf{W}^{R_1} \rangle_{i-\frac{1}{2},j})$$

5. Transform the face-centered fluxes to fourth-order accurate face-averaged fluxes

$$\langle \mathbf{F}_1 \rangle_{i-\frac{1}{2},j} = \mathbf{F}_{1,i-\frac{1}{2},j} - \frac{h^2}{24} \Delta^{\perp,1} \overline{\mathbf{F}}_{1,i-\frac{1}{2},j}$$

Athena++: management and reproducibility

- Complete redesign & rewrite of Athena (C) in C++
  - ~63,000 lines of C++
  - ~7,000 lines of Python utilities
- >2,000 commits since May 2014
- 10-20 developers

Athena++: management and reproducibility

- Complete redesign & rewrite of Athena (C) in C++
    - ~63,000 lines of C++
    - ~7,000 lines of Python utilities
- >2,000 commits since May 2014
- 10-20 developers

Athena++: management and reproducibility

- Complete redesign & rewrite of Athena (C) in C++
  - ~63,000 lines of C++
  - ~7,000 lines of Python utilities
- >2,000 commits since May 2014
- 10-20 developers
- Modern GitHub workflow
- Style checkers/code linters
- Continuous integration

Jenkins

https://jenkins.io/

Travis CI

https://travis-ci.com/

Athena++: management and reproducibility

- Complete redesign & rewrite of Athena (C) in C++
  - ~63,000 lines of C++
  - ~7,000 lines of Python utilities
- >2,000 commits since May 2014
- 10-20 developers
- Modern GitHub workflow
- Style checkers/code linters
- Continuous integration

Jenkins

https://jenkins.io/

Travis CI

https://travis-ci.com/

v1.1.0 version released on 5/25/18
https://github.com/PrincetonUniversity/athena-public-version

Athena++

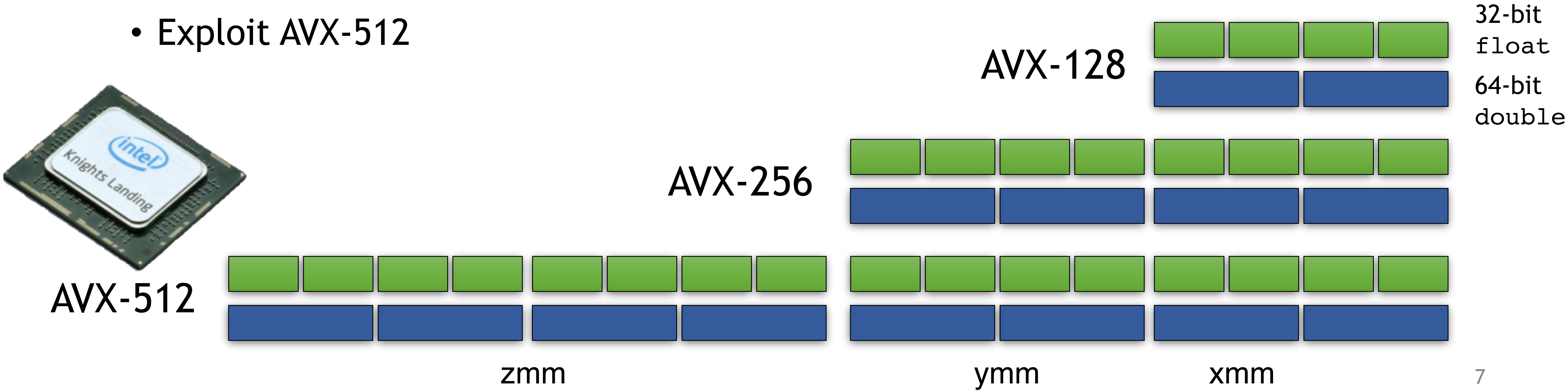
- New core algorithms and physics
  - Adaptive and static mesh refinement
  - General relativity
- Greater source code modularity
- Improved performance
  - Task-based execution model
  - Highly scalable MPI+OpenMP
  - OpenMP 4.5 SIMD explicit vectorization

			MZone-cycles/sec		
			Xeon Phi KNL 7210	Broadwell E5-2680 v4	Skylake-SP Gold 6148
Hydro Sod	PLM	HLLC	1.518	2.660	4.475
		HLLE	1.625	2.773	4.808
		Roe	1.561	2.873	4.697
	PPM	HLLC	0.758	1.328	2.408
		HLLE	0.771	1.331	2.518
		Roe	0.757	1.360	2.414
MHD Brio-Wu	PLM	HLLD	0.708	1.338	2.421
		HLLE	0.806	1.404	2.322
		Roe	0.652	1.104	1.926
	PPM	HLLD	0.395	0.720	1.291
		HLLE	0.424	0.748	1.265
		Roe	0.375	0.643	1.118

Table 1: single-core performance for baseline 3D Newtonian solver configurations. 64^3 uniform Cartesian mesh, VL2+PLM, Intel compiler 18.0.

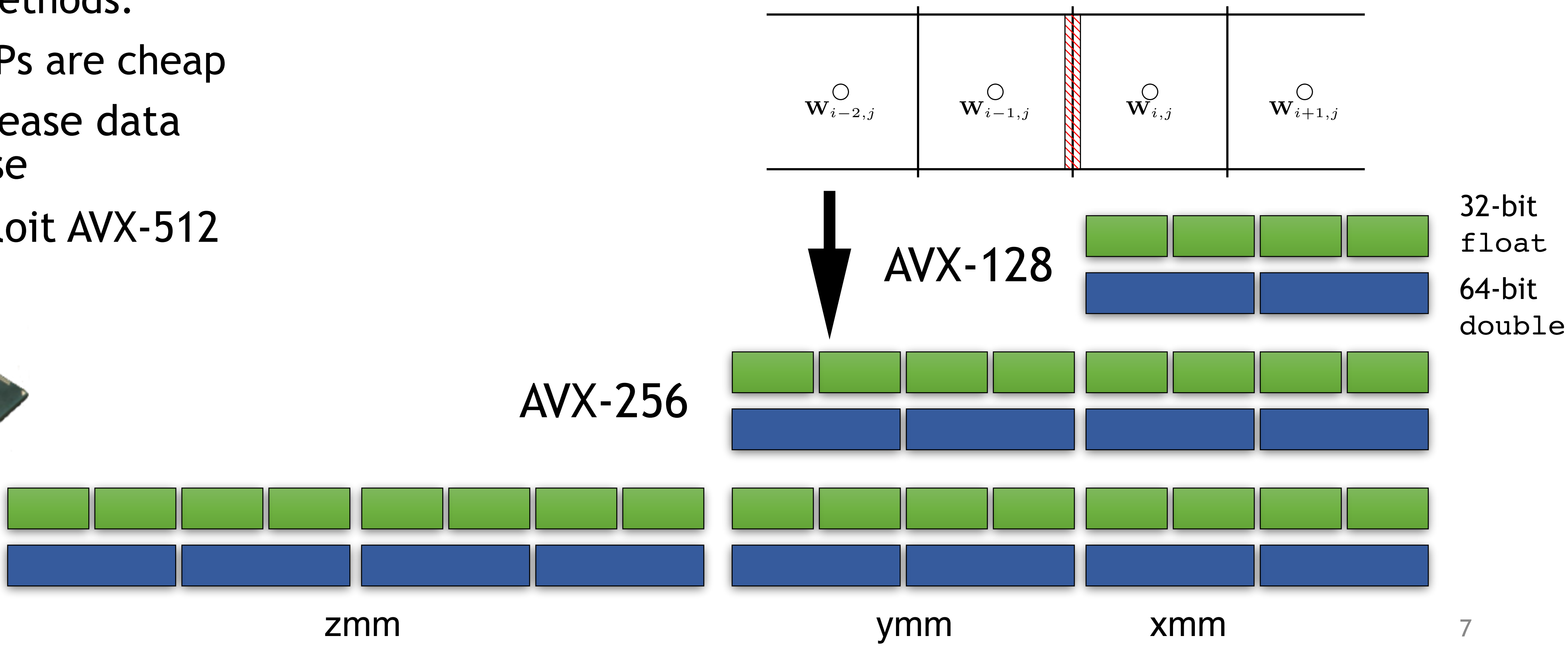
SIMD Vectorization

- GPUs or x86?
- Motivations for high-order methods:
    - FLOPs are cheap
    - Increase data reuse
    - Exploit AVX-512



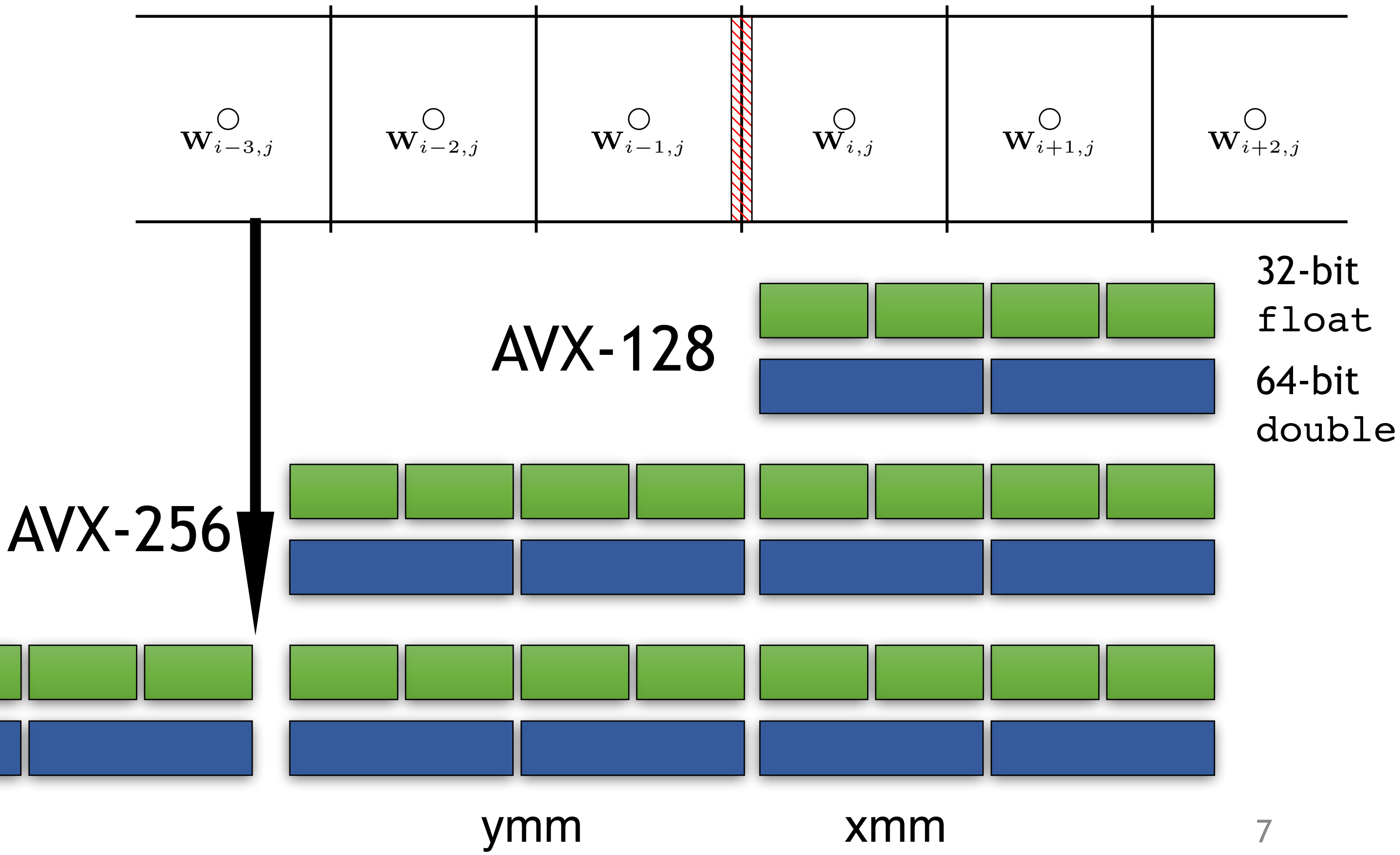
SIMD Vectorization

- GPUs or x86?
- Motivations for high-order methods:
  - FLOPs are cheap
  - Increase data reuse
  - Exploit AVX-512



SIMD Vectorization

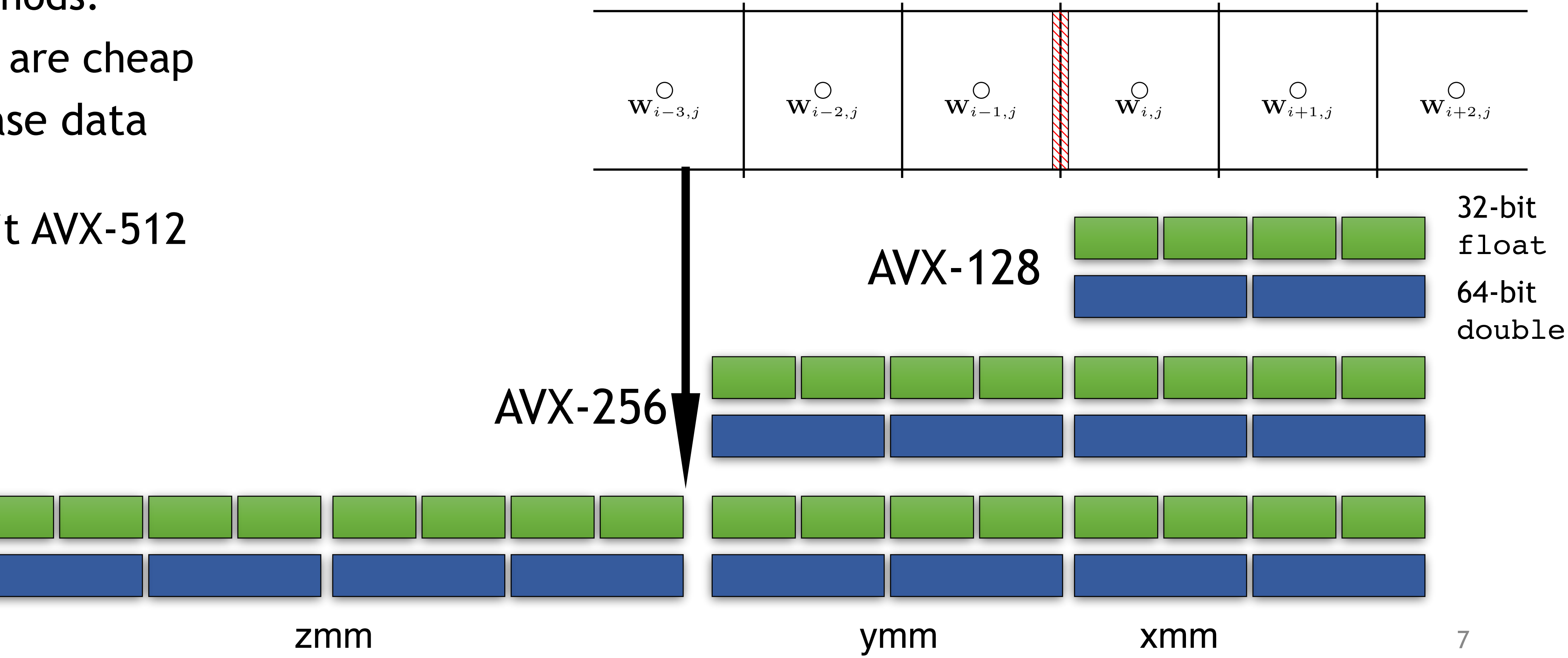
- GPUs or x86?
- Motivations for high-order methods:
  - FLOPs are cheap
  - Increase data reuse
  - Exploit AVX-512



SIMD Vectorization

- Application is memory bandwidth bound
- Intel Turbo Boost causes frequency throttling

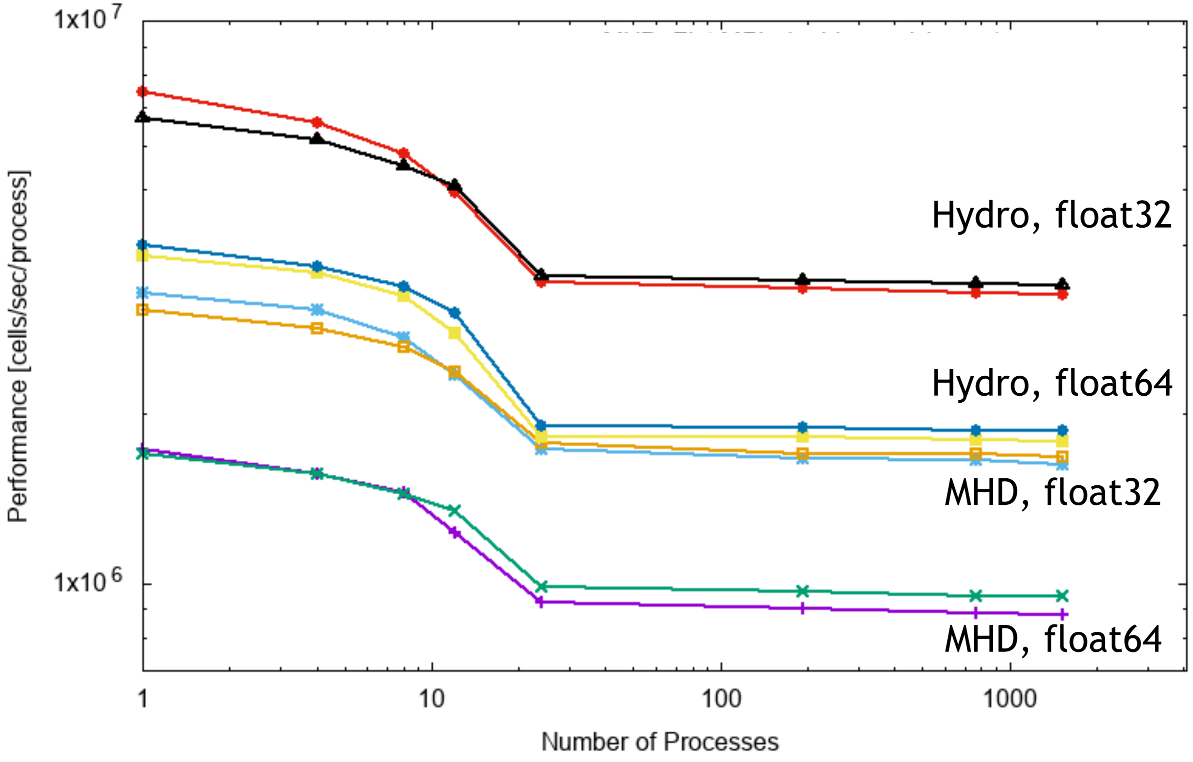
- GPUs or x86?
- Motivations for high-order methods:
  - FLOPs are cheap
  - Increase data reuse
  - Exploit AVX-512



Athena++ scalability

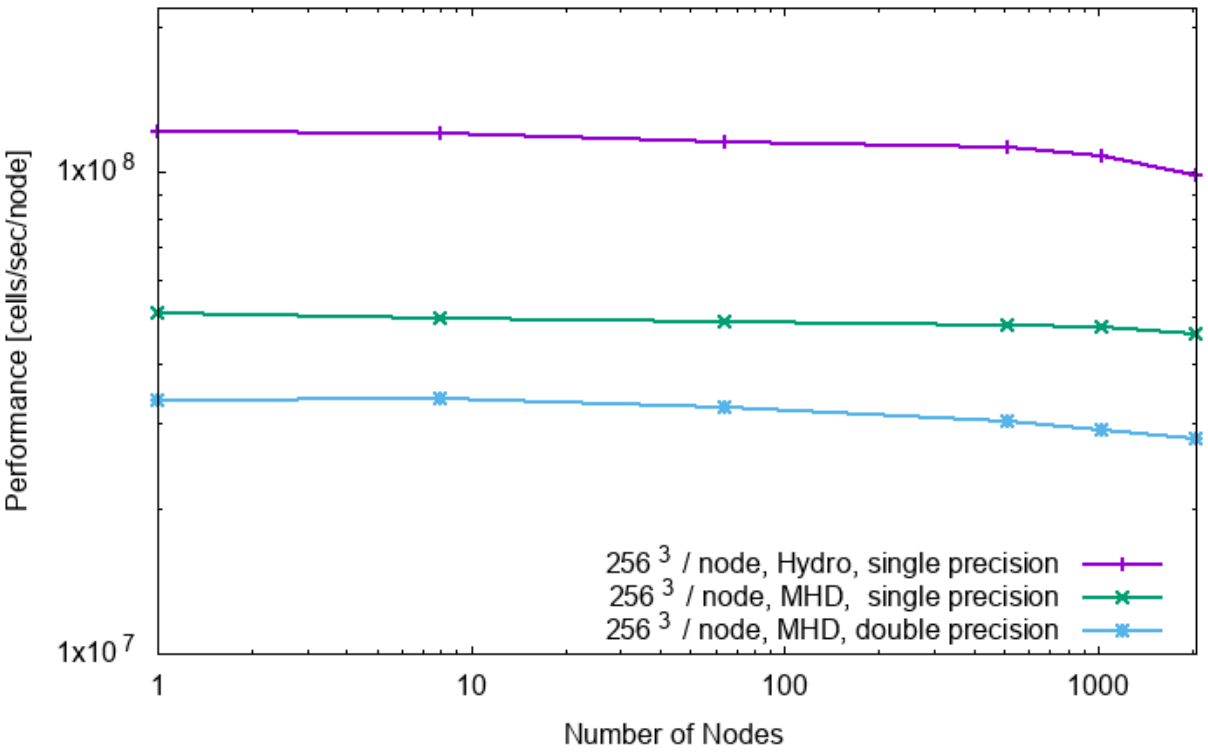
icc: `-O3 -std=c++11 -ipo -xhost -inline-forceinline -qopenmp-simd -qopt-prefetch=4`

`-qopt-zmm-usage=low`

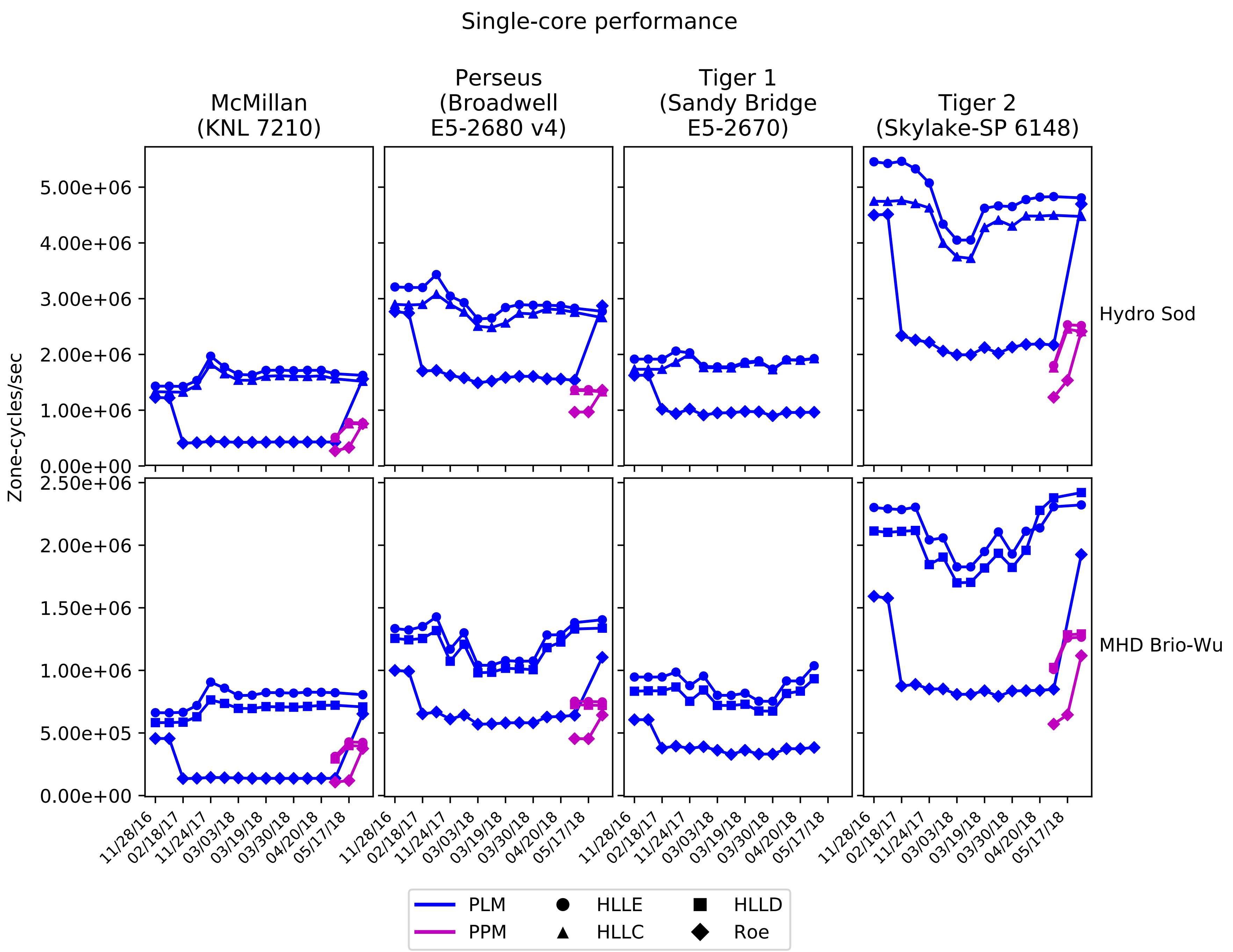


Osaka University OCTOPUS cluster (Skylake-SP 6126):
on-node (2x 12-core) and off-node scaling

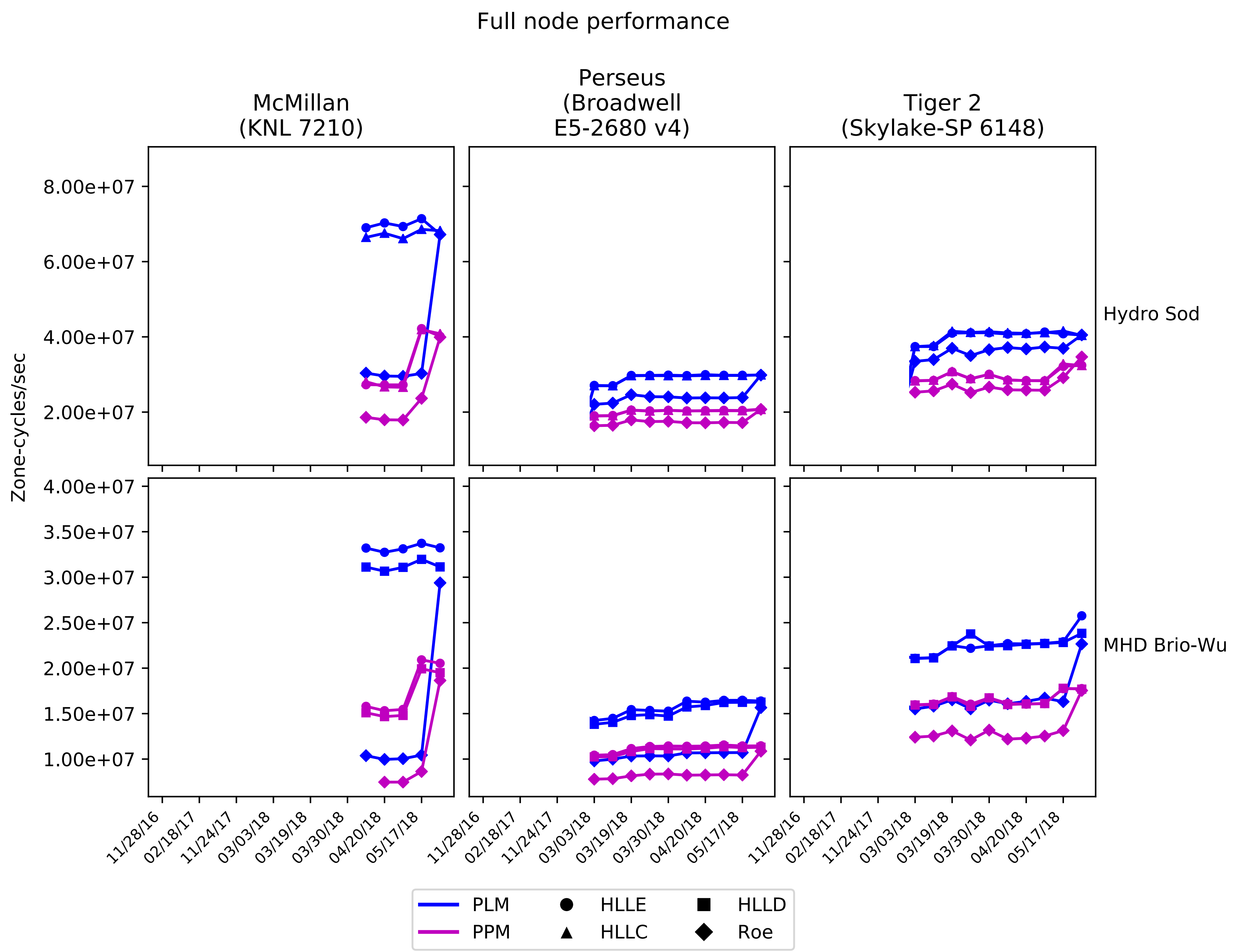
`-qopt-zmm-usage=high`



JCAHPC Oakforest supercomputer (KNL 7250):
off-node scaling (using 64 of 68-core Xeon Phi)

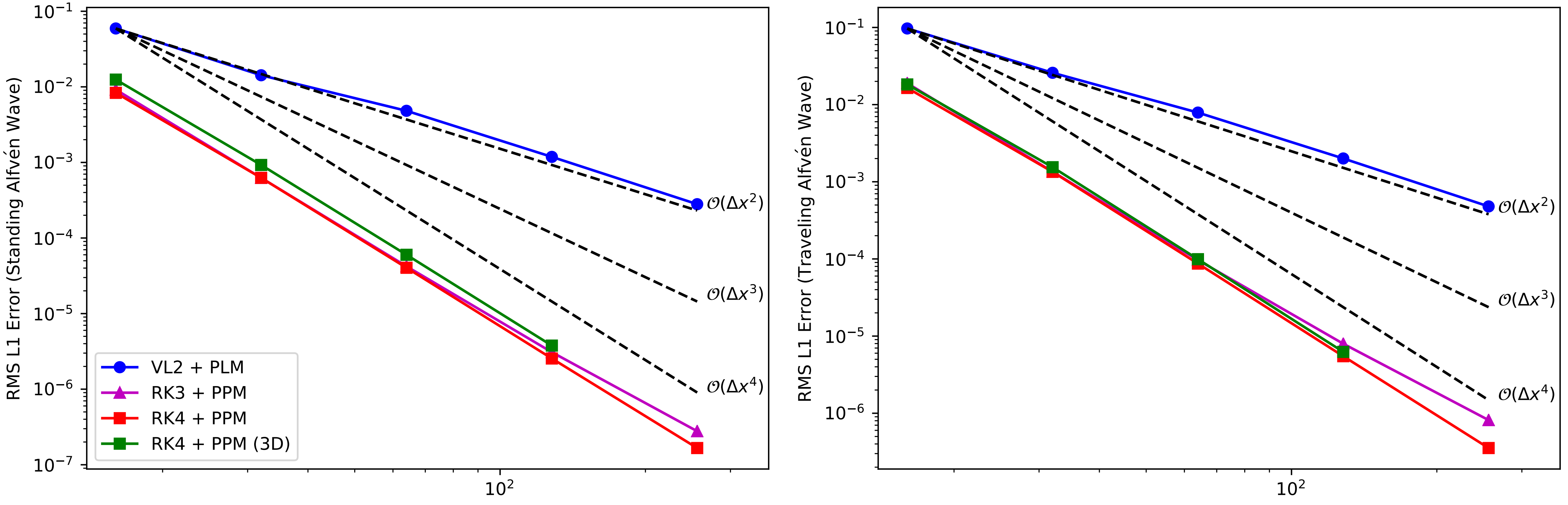


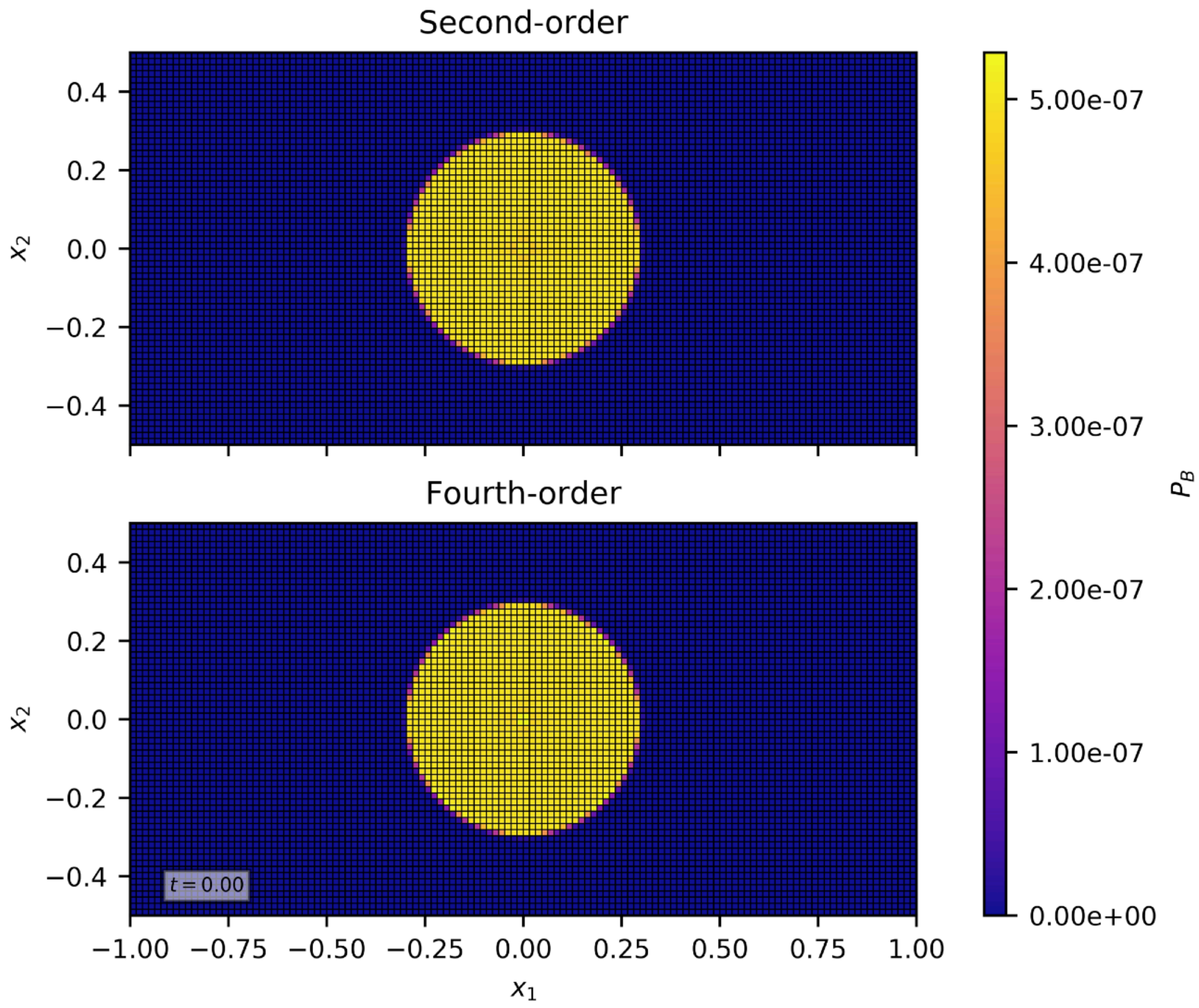
Single-core performance
McMillan (KNL 7210)
Perseus (Broadwell E5-2680 v4)
Tiger 1 (Sandy Bridge E5-2670)
Tiger 2 (Skylake-SP 6148)
Zone-cycles/sec
Hydro Sod
MHD Brio-Wu
5.00e+06
4.00e+06
3.00e+06
2.00e+06
1.00e+06
0.00e+00
2.50e+06
2.00e+06
1.50e+06
1.00e+06
5.00e+05
0.00e+00
11/28/16
02/18/17
11/24/17
03/03/18
03/19/18
03/30/18
04/20/18
05/17/18
PLM
PPM
HLLE
HLLC
HLLD
Roe


Full node performance
McMillan (KNL 7210)
Perseus (Broadwell E5-2680 v4)
Tiger 2 (Skylake-SP 6148)
Zone-cycles/sec
8.00e+07
6.00e+07
4.00e+07
2.00e+07
4.00e+07
3.50e+07
3.00e+07
2.50e+07
2.00e+07
1.50e+07
1.00e+07
Hydro Sod
MHD Brio-Wu
11/28/16
02/18/17
11/24/17
03/03/18
03/19/18
03/30/18
04/20/18
05/17/18
PLM
PPM
HLLE
HLLC
HLLD
Roe

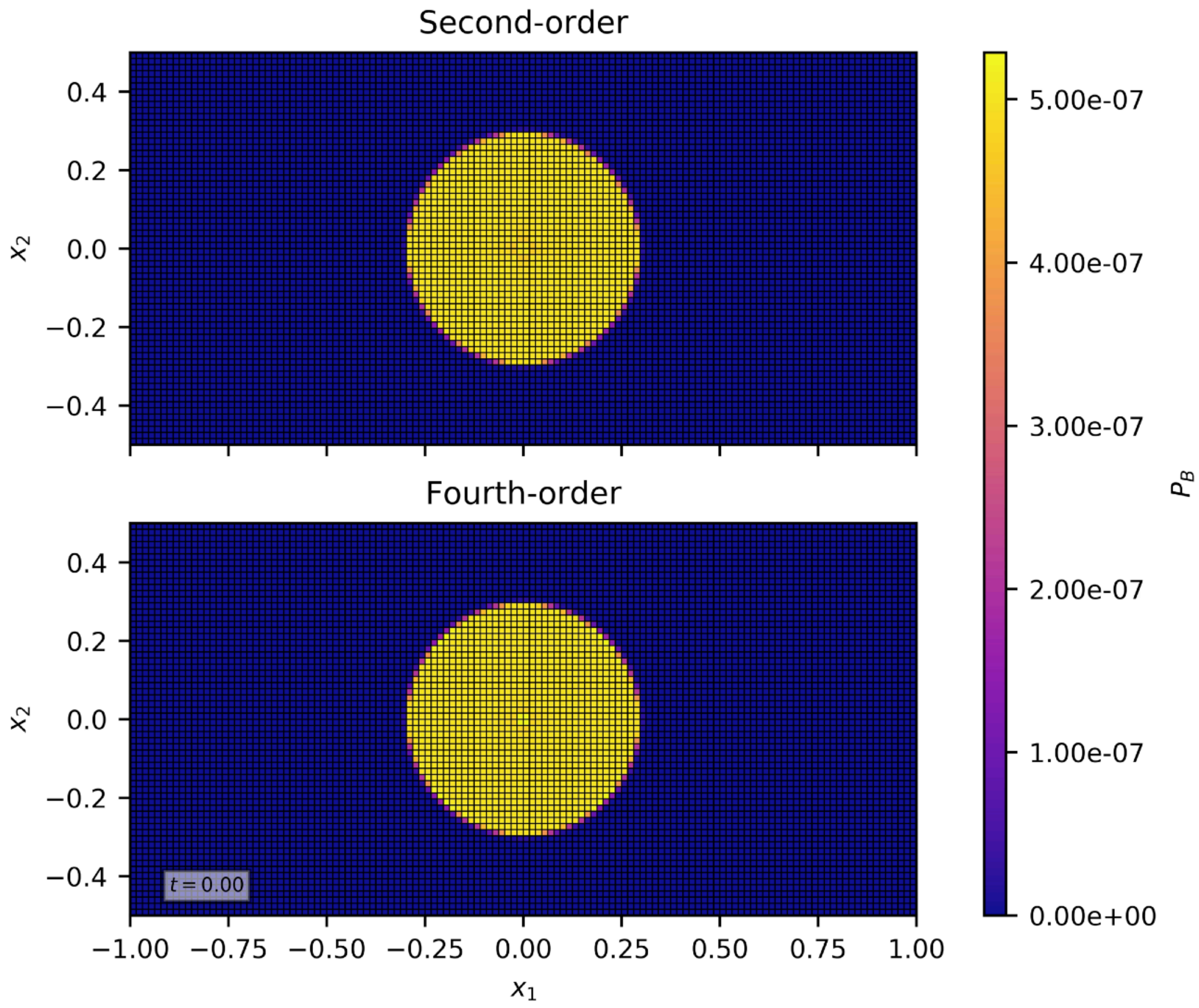
Fourth-order results

Circularly polarized Alfvén wave

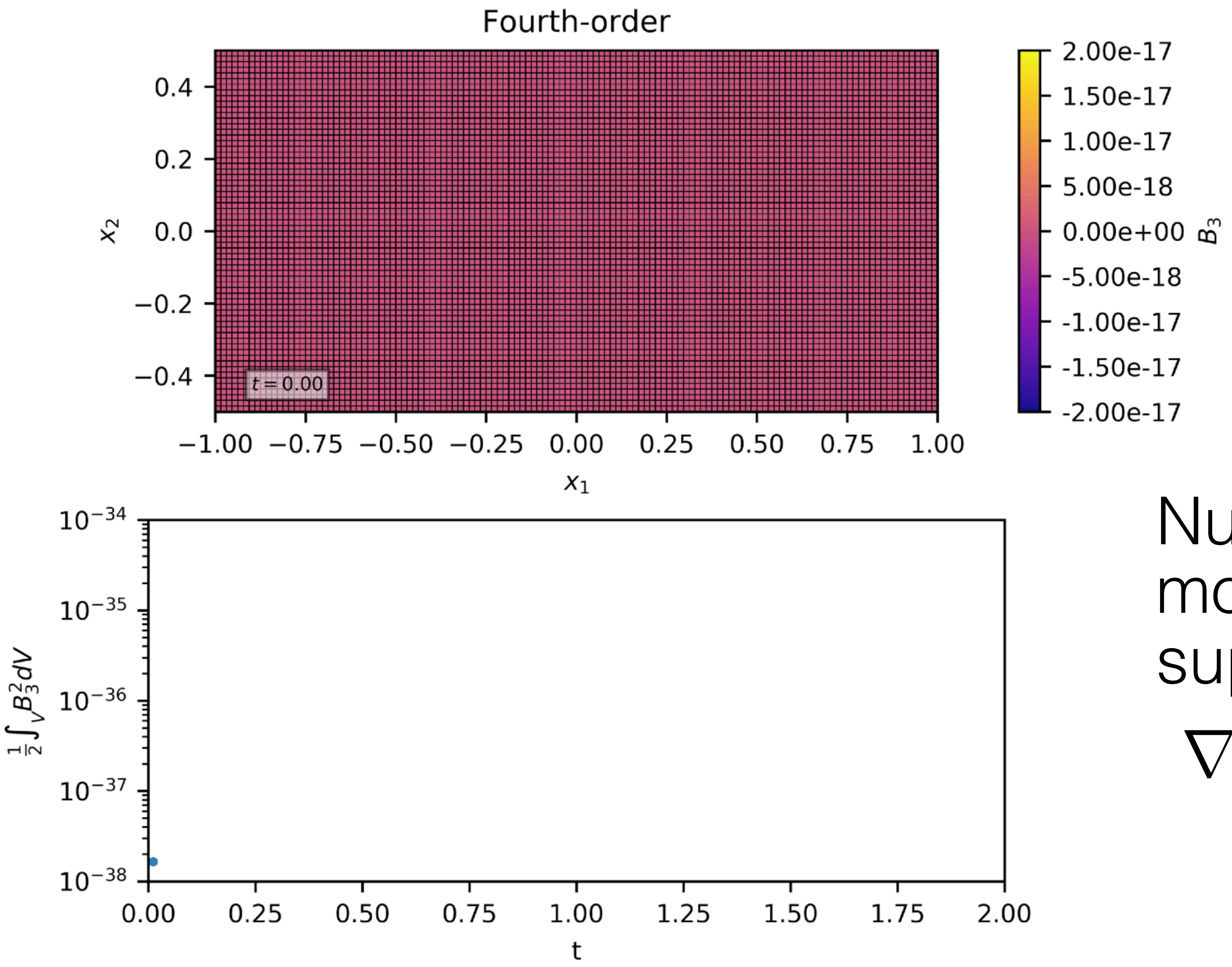




Diagonal advection of field loop

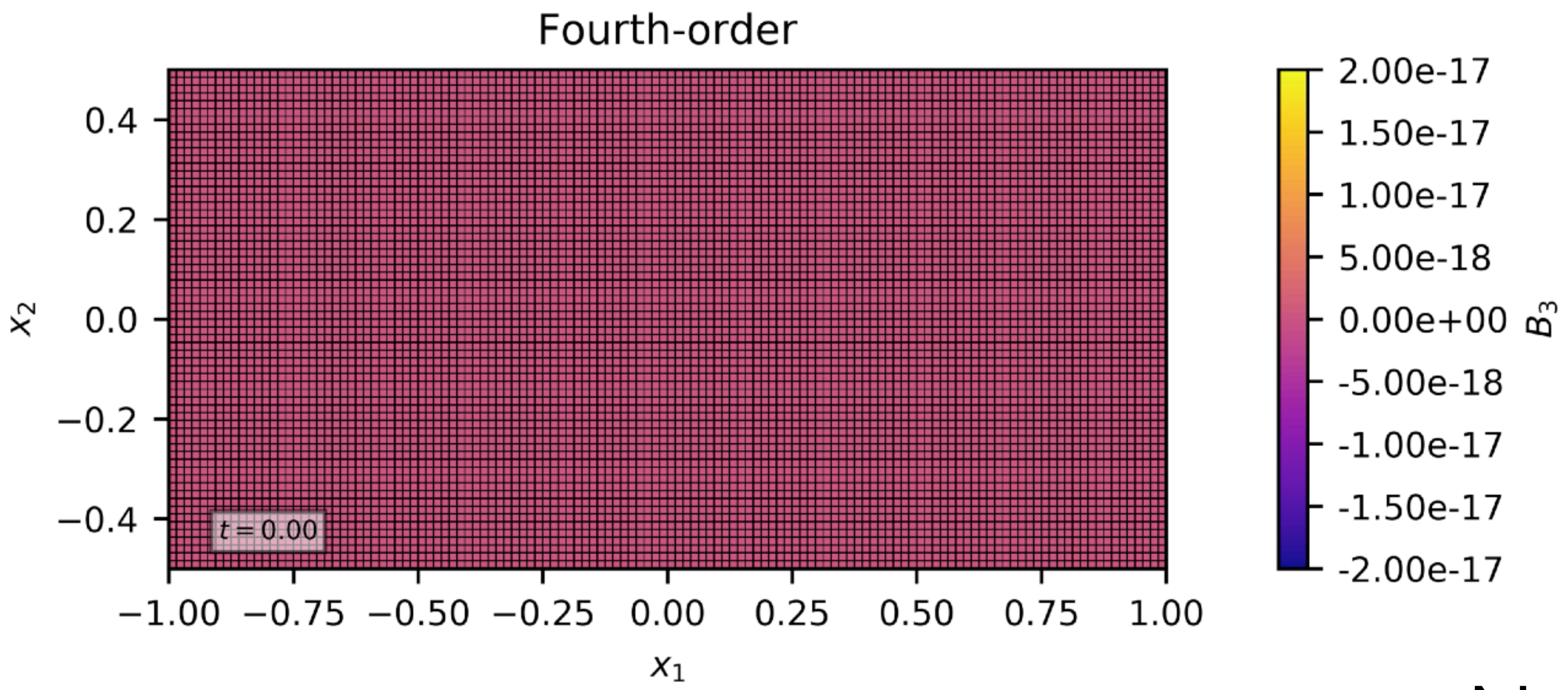


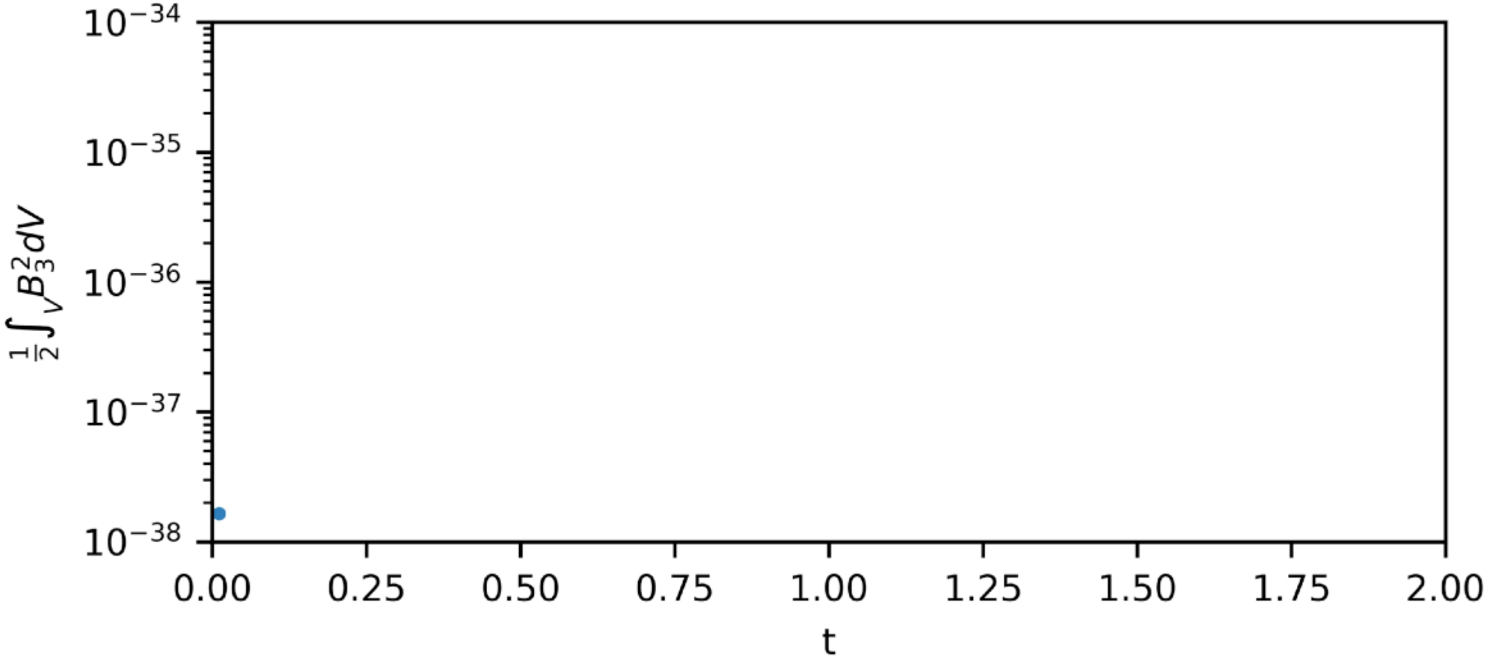
Diagonal advection of field loop



Numerical monopole suppression

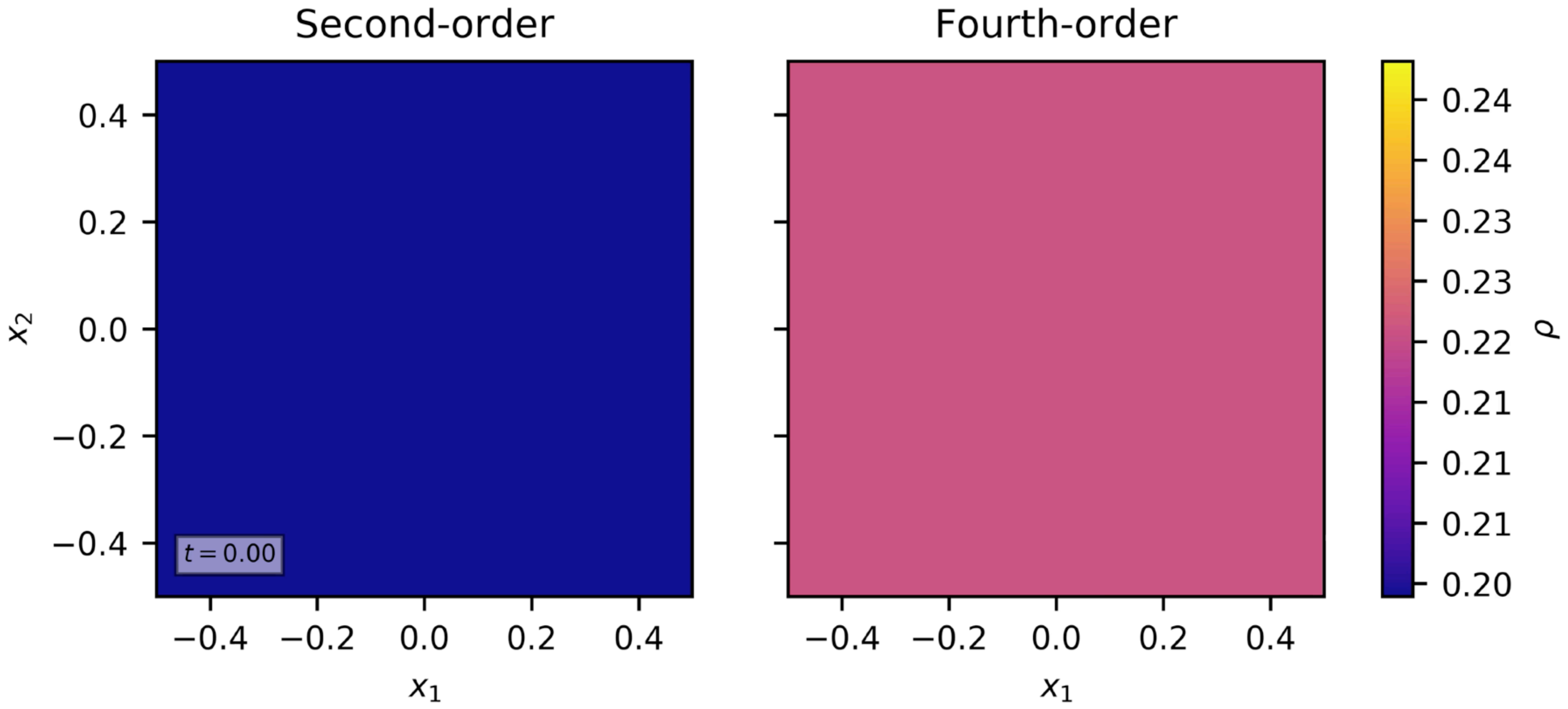
$$\nabla \cdot \mathbf{B} = 0$$



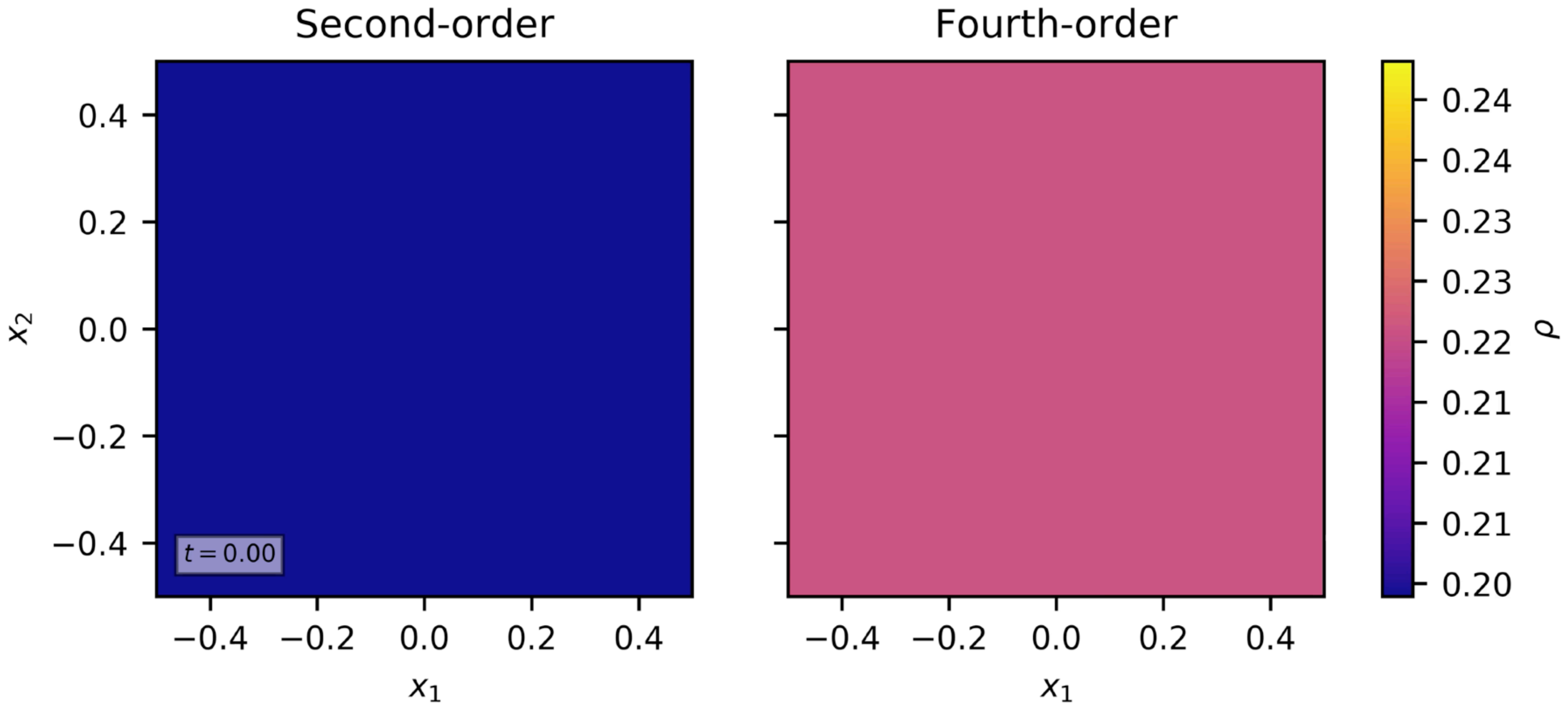


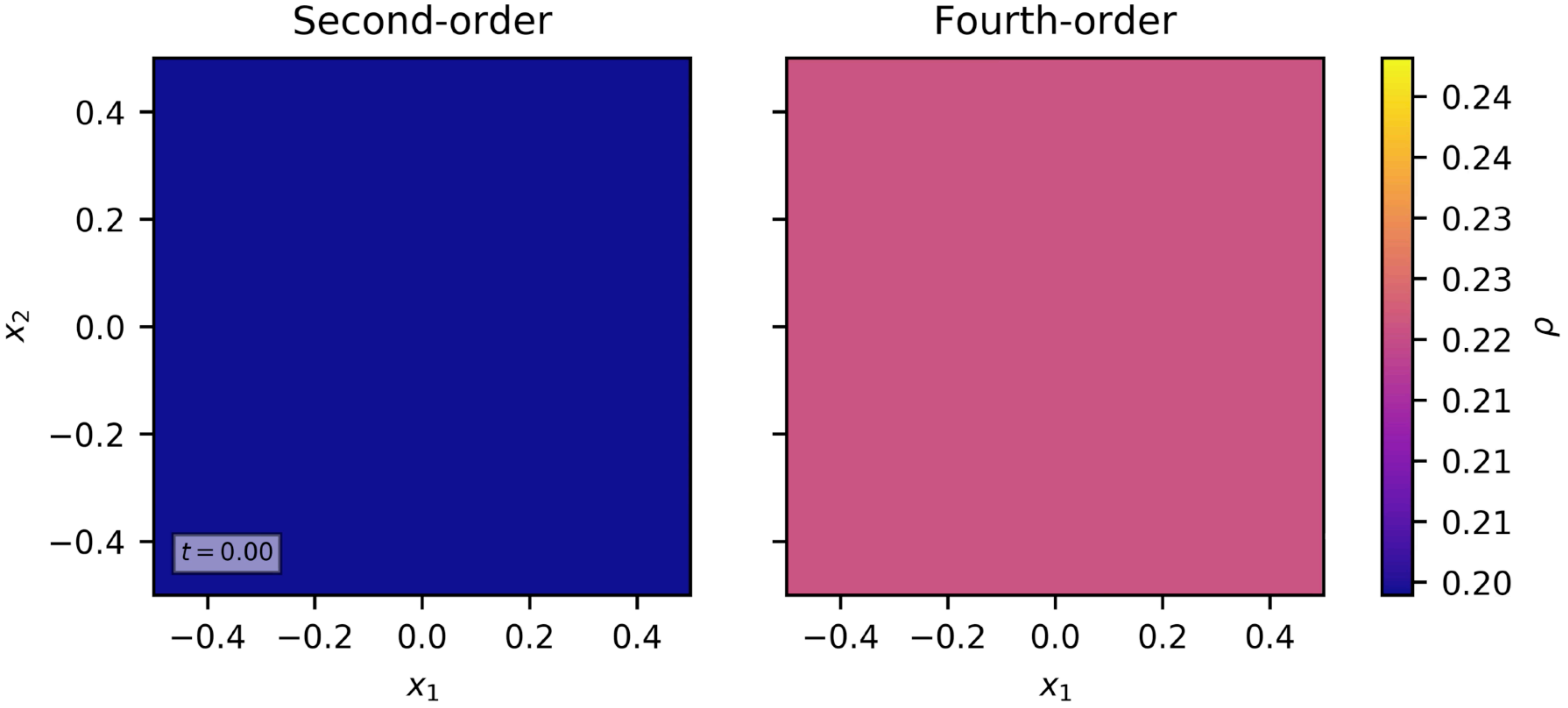
Numerical monopole suppression

$$\nabla \cdot \mathbf{B} = 0$$



Second-order
Fourth-order
0.4
0.2
0.0
−0.2
−0.4
x_2
$t = 0.00$
−0.4
−0.2
0.0
0.2
0.4
x_1
0.24
0.24
0.23
0.23
0.22
0.21
0.21
0.21
0.20
ρ

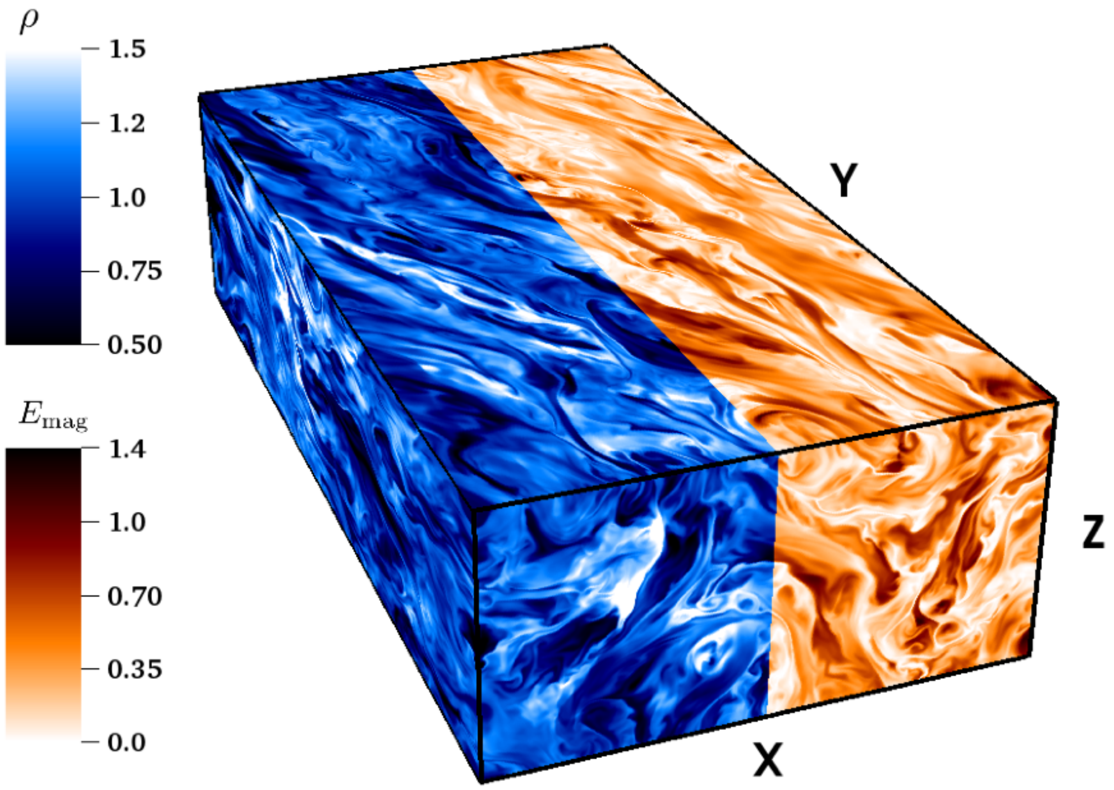

Second-order
Fourth-order
0.4
0.2
0.0
−0.2
−0.4
x_2
$t = 0.00$
−0.4 −0.2 0.0 0.2 0.4
x_1
−0.4 −0.2 0.0 0.2 0.4
x_1
0.24
0.24
0.23
0.23
0.22
0.21
0.21
0.21
0.20
ρ



~ 6-8x more expensive (but higher arithmetic intensity)!

Future work

- Candidate for release in next public version of Athena++, v1.2.0
- Computational cost and efficiency study:
  - Explore the zoo of algorithmic options in Athena++
- Fourth-order shearing box simulations
- Extend to grids with adaptive mesh refinement and general relativity



Thank you!