PASC17
Lugano Switzerland
26-28 June 2017

The Particle-In-Fourier (PIF) Approach Applied to Gyrokinetic Simulations and its Implementation on GPUs

PASC Conference, Lugano

June 26, 2017

Noé Ohana [1], Andreas Jocksch [2], Emmanuel Lanti [1], Aaron Scheinberg [1], Stephan Brunner [1], Claudio Gheller [2], Laurent Villard [1]

[1] SPC, Lausanne

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
SWISS PLASMA CENTER

[2] CSCS, Lugano

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Outline

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
SWISS PLASMA CENTER

1. Introduction

2. Model
   - Physical framework
   - Numerical implementation

3. Precision of physical results
   - Linear simulations
   - Non-linear simulations

4. Numerical performance
   - PIC
   - PIF
   - Parallel scalability

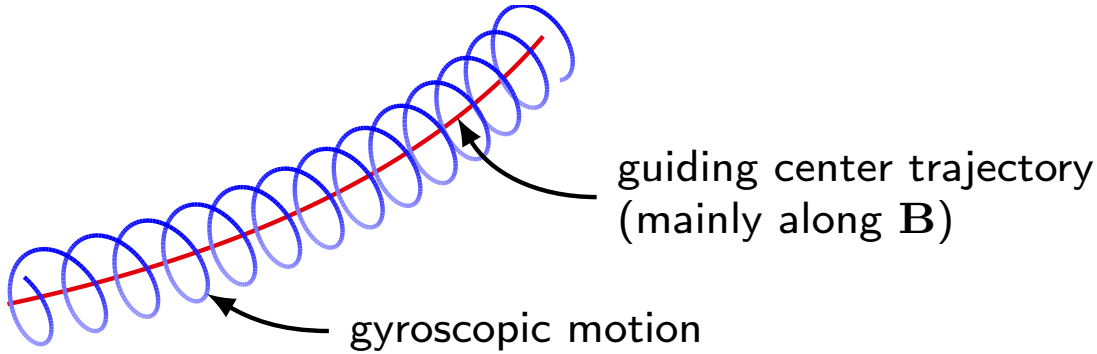
5. Conclusion

Motivation of our PASC project

- Adapt legacy code ORB5 [Tran, 1999, Jolliet, 2009] to new architectures such as GPUs
- Keep portability $\Rightarrow$ OpenACC directive-based approach
- Investigate alternative field representation (PIF instead of PIC)
- Modularize data structures for easier maintenance

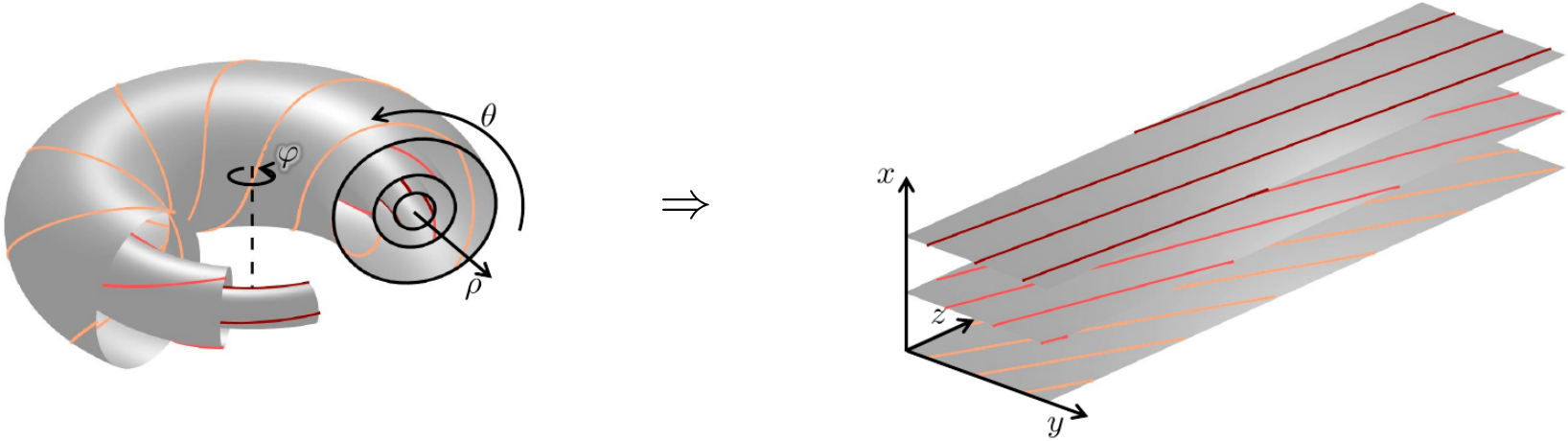
GK-engine features

- Physics:
  - Gyrokinetic equations of particle motion



  - Slab geometry
  - Sheared magnetic field
  - Electrostatic, collisionless instabilities
  - Adiabatic electrons
- Numerics:
  - Runge-Kutta of fourth order time integrator
  - Intra-node multi-threading with OpenMP or OpenACC
  - Inter-node MPI communication with domain decomposition and cloning

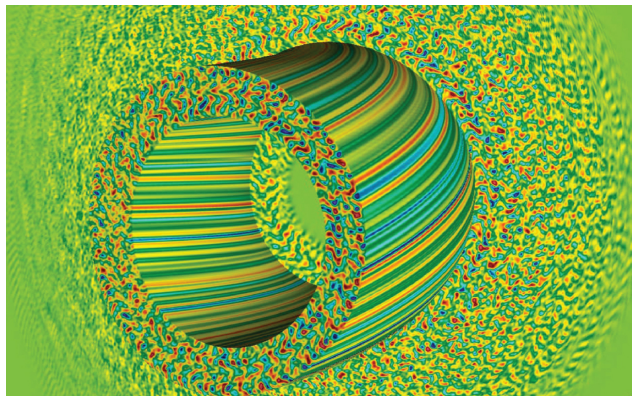
Magnetic field and anisotropy



- Toroidal (z) and poloidal (y) magnetic field with radially-dependent (x) shear:

$$\mathbf{B}(x) = B_z \mathbf{e}_z + B_y(x)\mathbf{e}_y \qquad q(x) = \frac{L_y}{L_z}\frac{B_z}{B_y}$$

- Strong anisotropy of turbulence along/across field lines:

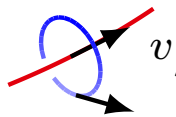


[Fasoli, 2016]

Vlasov-Maxwell system of equations

- Distribution function:

$$f = f_0(\mathbf{R}, v_{/\!/}, \mu) + \delta f(\mathbf{R}, v_{/\!/}, \mu, t)$$



$$v_\perp \rightarrow \mu = \frac{m v_\perp^2}{2B}$$

- Vlasov:

$$\frac{df}{dt} = 0 \quad \Rightarrow \quad \frac{d\,\delta f}{dt} = -\frac{d\mathbf{R}}{dt} \cdot \frac{\partial f_0}{\partial \mathbf{R}} - \frac{dv_{/\!/}}{dt} \frac{\partial f_0}{\partial v_{/\!/}}$$

- Equations of motion:

$$\frac{d\mathbf{R}}{dt} = \underbrace{v_{/\!/}\mathbf{b}}_{\text{parallel motion}} + \underbrace{\frac{\mu}{eB^\star_{/\!/}}\mathbf{b} \wedge \boldsymbol{\nabla} B}_{\boldsymbol{\nabla} B \text{ drift}} + \underbrace{\frac{1}{B^\star_{/\!/}}\mathbf{b} \wedge \boldsymbol{\nabla}\langle\phi\rangle}_{\mathbf{E}\wedge\mathbf{B} \text{ drift}}$$

$$\frac{dv_{/\!/}}{dt} = -\frac{e}{m}\mathbf{b} \cdot \boldsymbol{\nabla}\langle\phi\rangle$$

$$\frac{d\mu}{dt} = 0$$

- Quasi-neutrality equation:

$$\int \frac{n_0 e}{T_e}(\phi - \bar{\phi})\eta \; d^3\mathbf{x} + \int \frac{m}{eB^2} n_0 \boldsymbol{\nabla}_\perp \phi \cdot \boldsymbol{\nabla}_\perp \eta \; d^3\mathbf{x} = \int \delta f \langle\eta\rangle \; d^3\mathbf{x} \; d^3\mathbf{v}$$

Particle and field representations

- Marker discretization:

$$\delta f(\mathbf{R}, v_{/\!/}, \mu, t) = \sum_{p=1}^{N_p} \frac{m}{2\pi B^{\star}_{/\!/}} w_p(t) \delta\big(\mathbf{R} - \mathbf{R}_p(t)\big) \delta\big(v_{/\!/} - v_{/\!/p}(t)\big) \delta\big(\mu - \mu_p\big)$$

- B-spline field representation (PIC method):

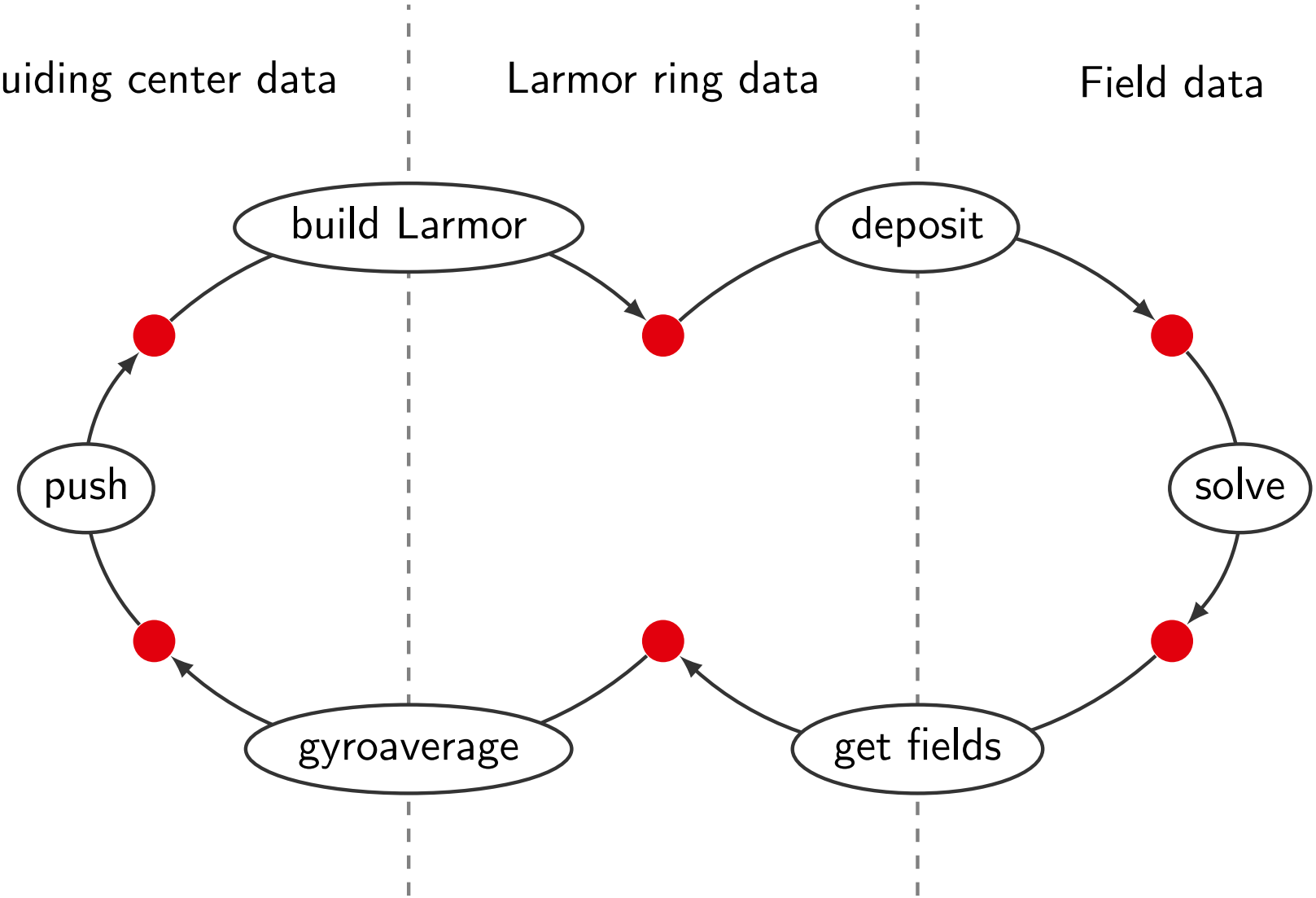
$$\phi(x, y, z) = \sum_{i=1}^{N_x+p} \sum_{j=1}^{N_y} \sum_{k=1}^{N_z} \phi_{ijk} \Lambda_i(x) \Lambda_j(y) \Lambda_k(z)$$

- Fourier mode field representation (PIF method):

$$\phi(x, y, z) = \sum_{i=1}^{N_x+p} \sum_{n=n_{\min}}^{n_{\max}} \sum_{m=-nq_i-\Delta m}^{-nq_i+\Delta m} \tilde{\phi}_{imn} \Lambda_i(x) e^{2\pi \mathrm{i}(my/L_y + nz/L_z)}$$

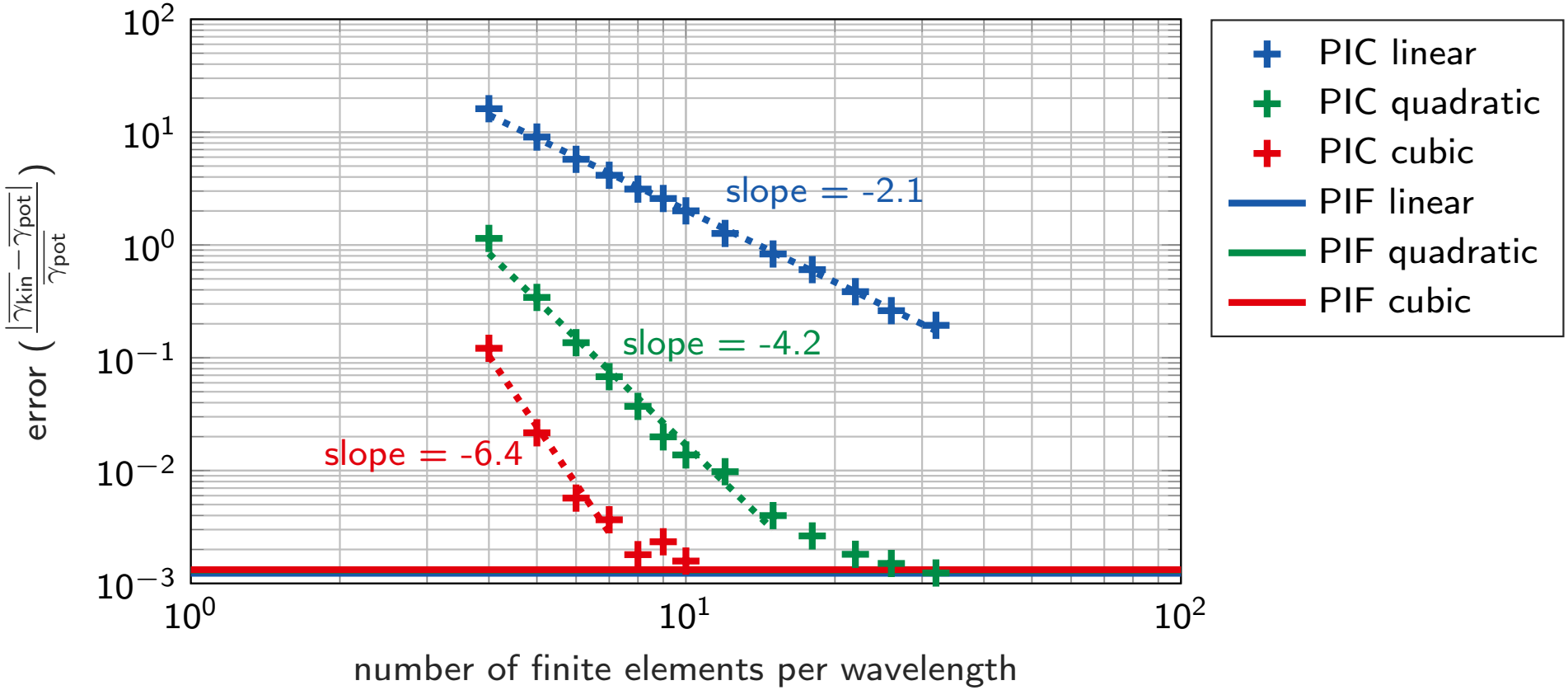
- Field aligned $\Leftrightarrow$ $k_{/\!/} \sim 0$ $\Leftrightarrow$ $m + nq(x) \sim 0$

Stages of a time step



Energy conservation

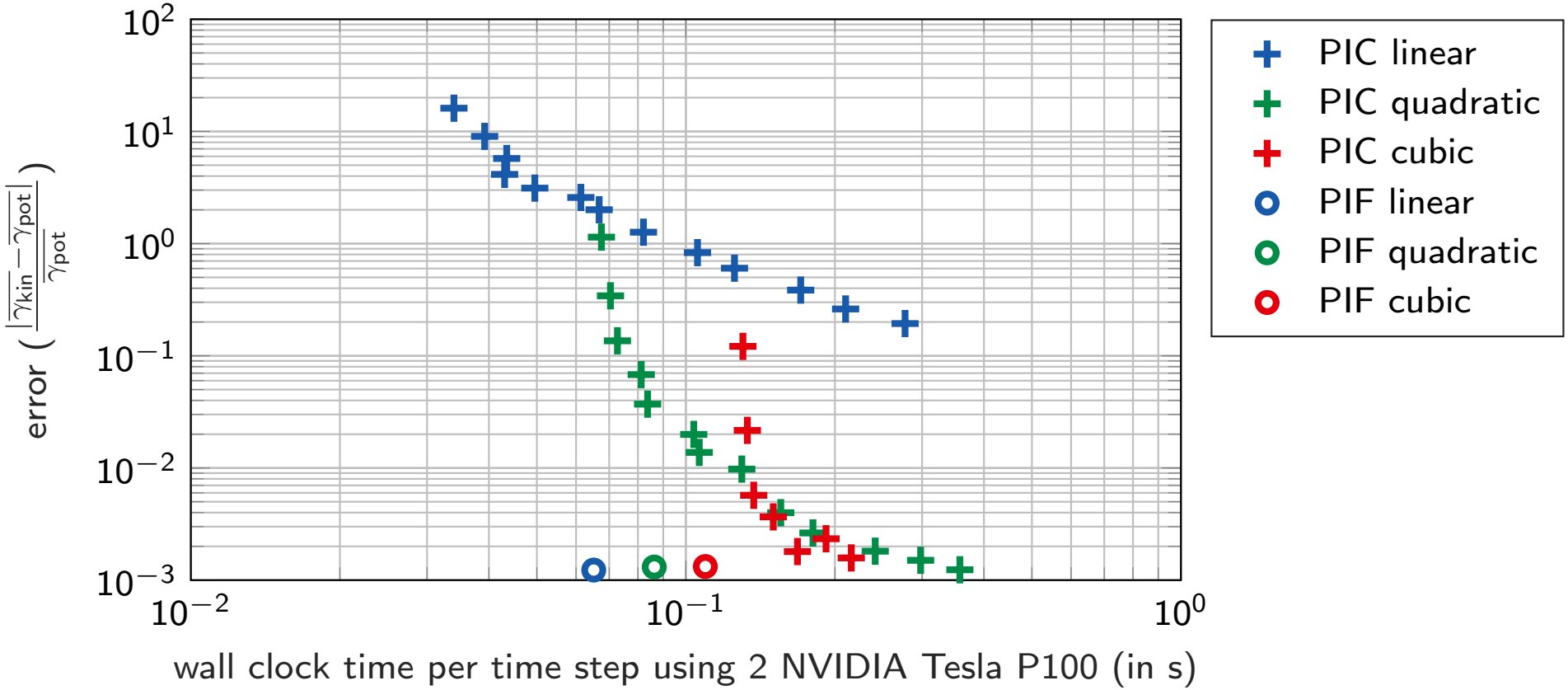
- Single Ion-Temperature-Gradient (ITG) mode $(m, n) = (-49, 35)$ (physical parameters from [Görler, 2016])
- Convergence with number of finite elements versus PIF:



- The saturation level depends on the other parameters.

Energy conservation

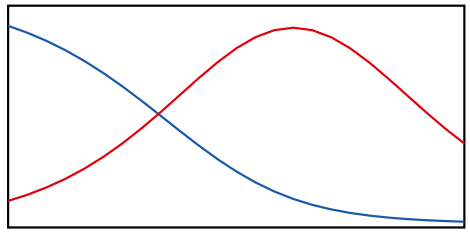
- Single Ion-Temperature-Gradient (ITG) mode $(m, n) = (-49, 35)$ (physical parameters from [Görler, 2016])
- Convergence with number of finite elements versus PIF:



- The saturation level depends on the other parameters.

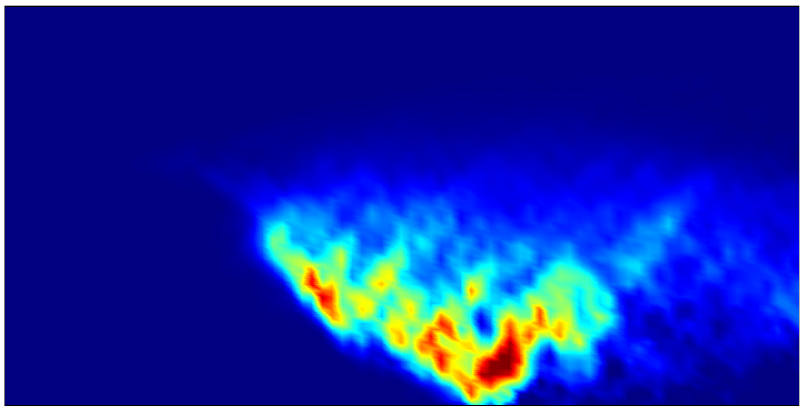
ITG turbulence

- ITG non-linear simulation

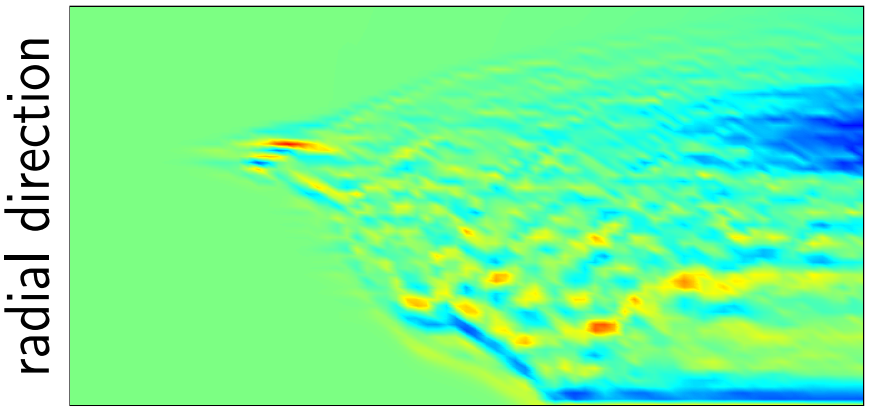


- Full Fourier spectrum (15 toroidal modes times 11 poloidal modes)
- PIC method:

Heat flux



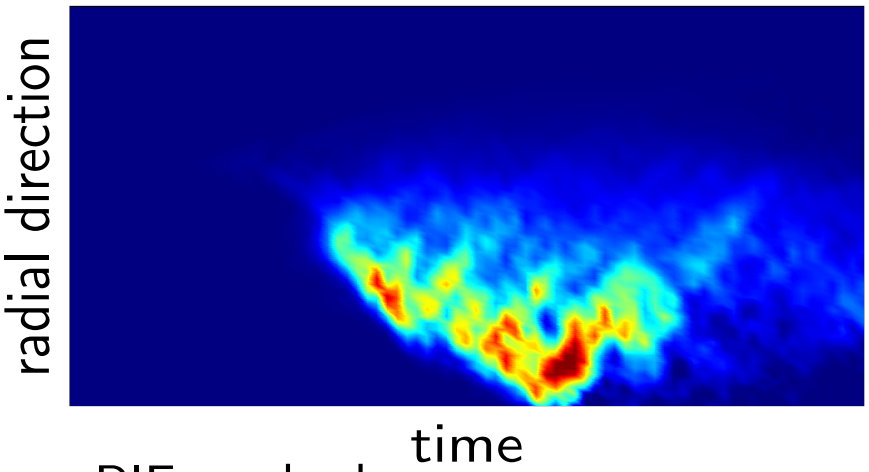
Zonal flow shearing rate



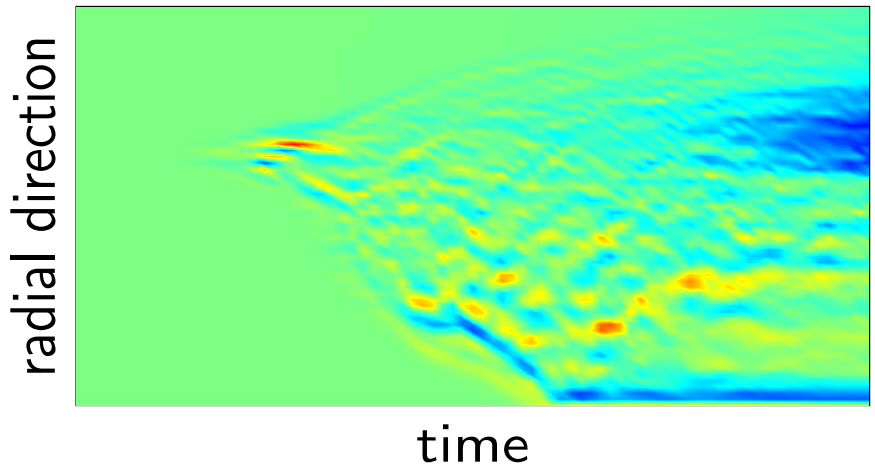
ITG turbulence

- PIC method:

Heat flux

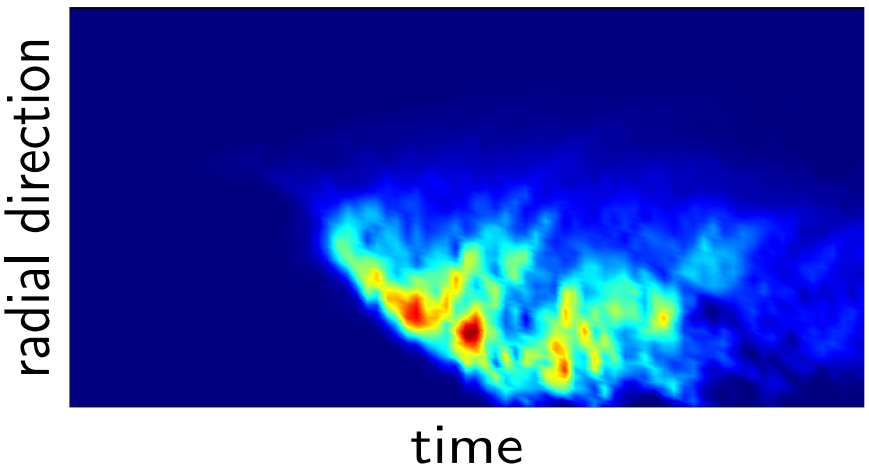


Zonal flow shearing rate

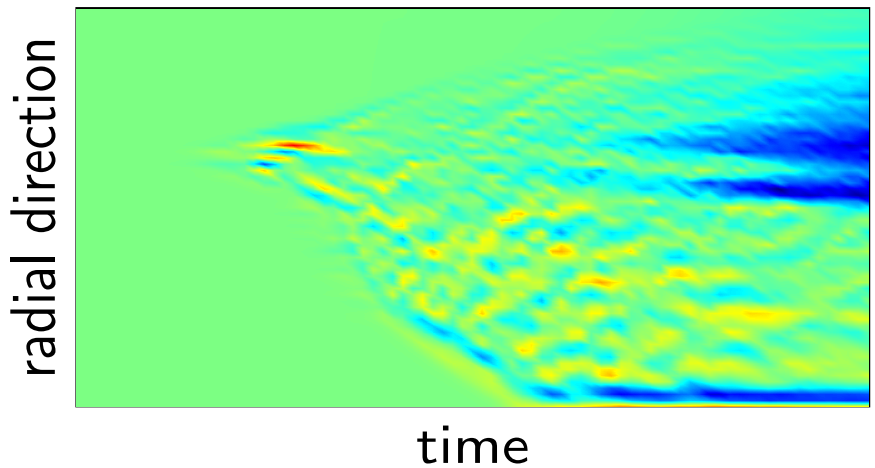


- PIF method:

Heat flux

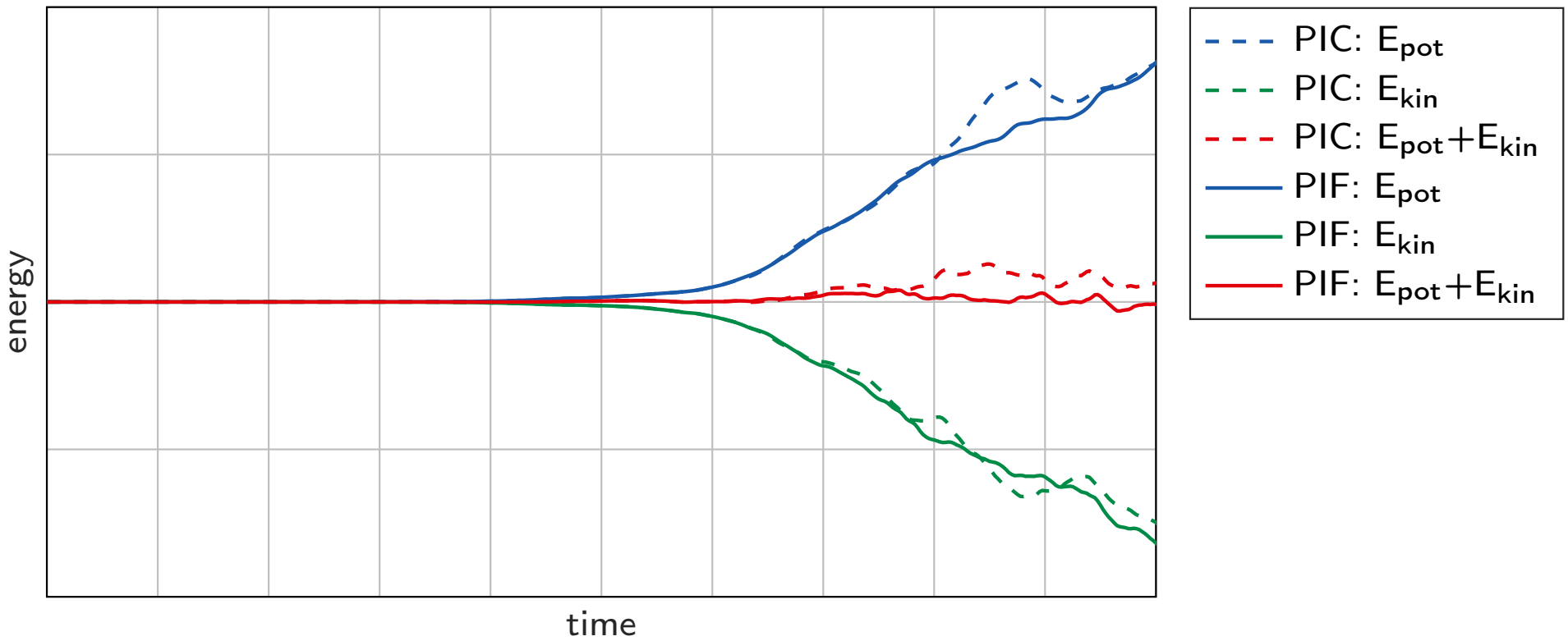


Zonal flow shearing rate



Energy conservation

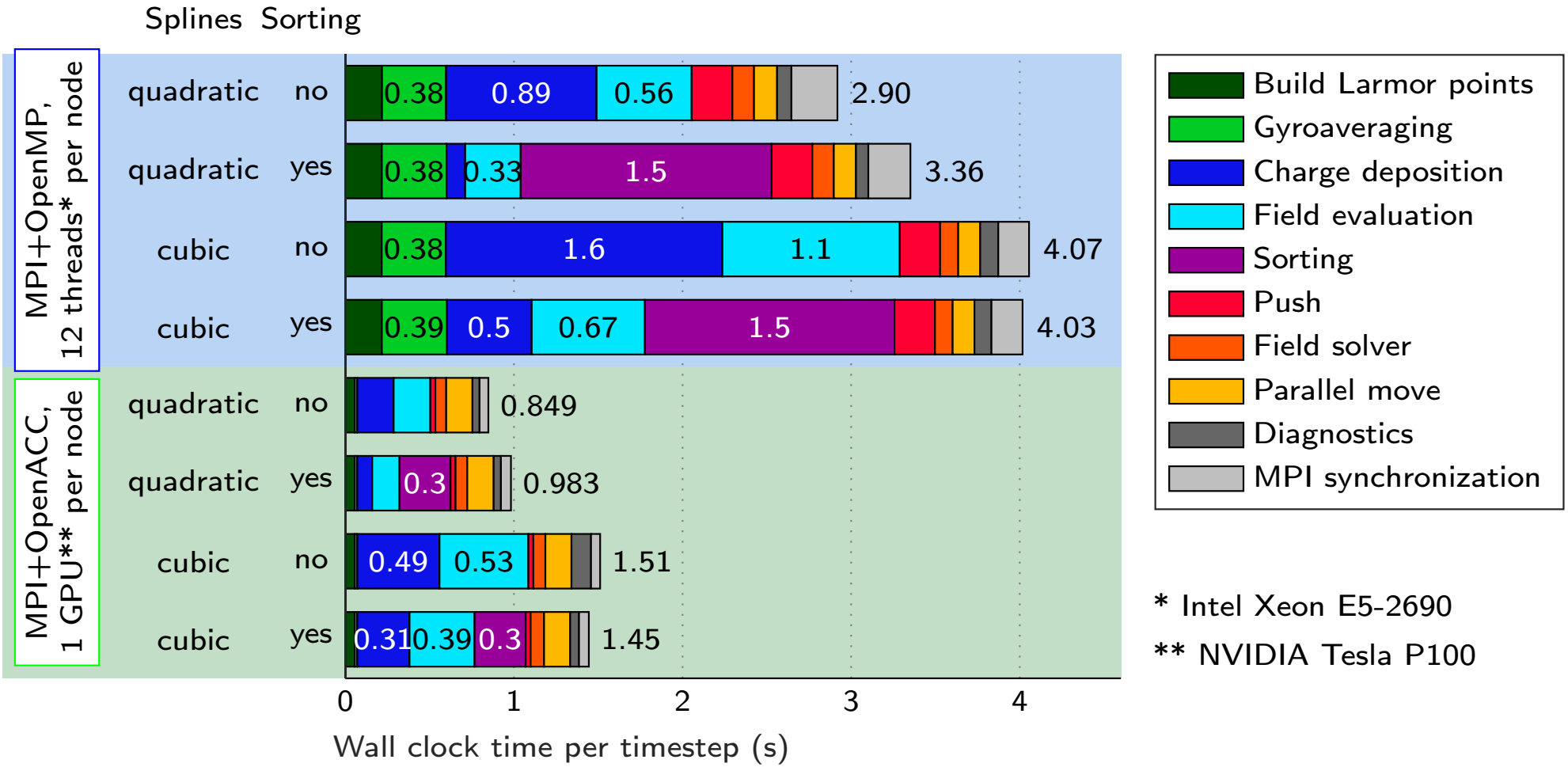
- Check energy conservation over time:



- Error with PIF method always lower than with PIC

Sorting particles in grid cells

- PIC, 32 nodes, 128M particles, $128 \times 512 \times 128$ grid cells



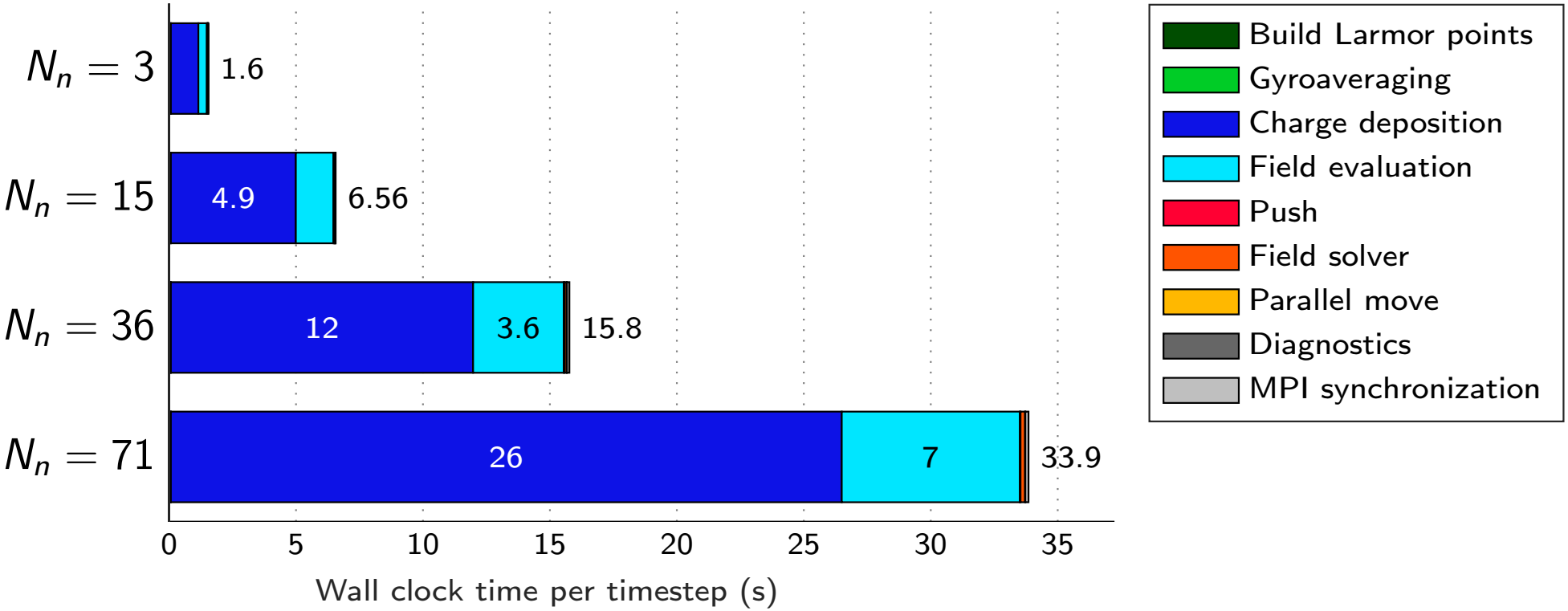
* Intel Xeon E5-2690

** NVIDIA Tesla P100

- Sorting interesting for cubic splines, not for quadratic
- GPU** up to 3.5 times faster than CPU*

Scanning number of toroidal modes

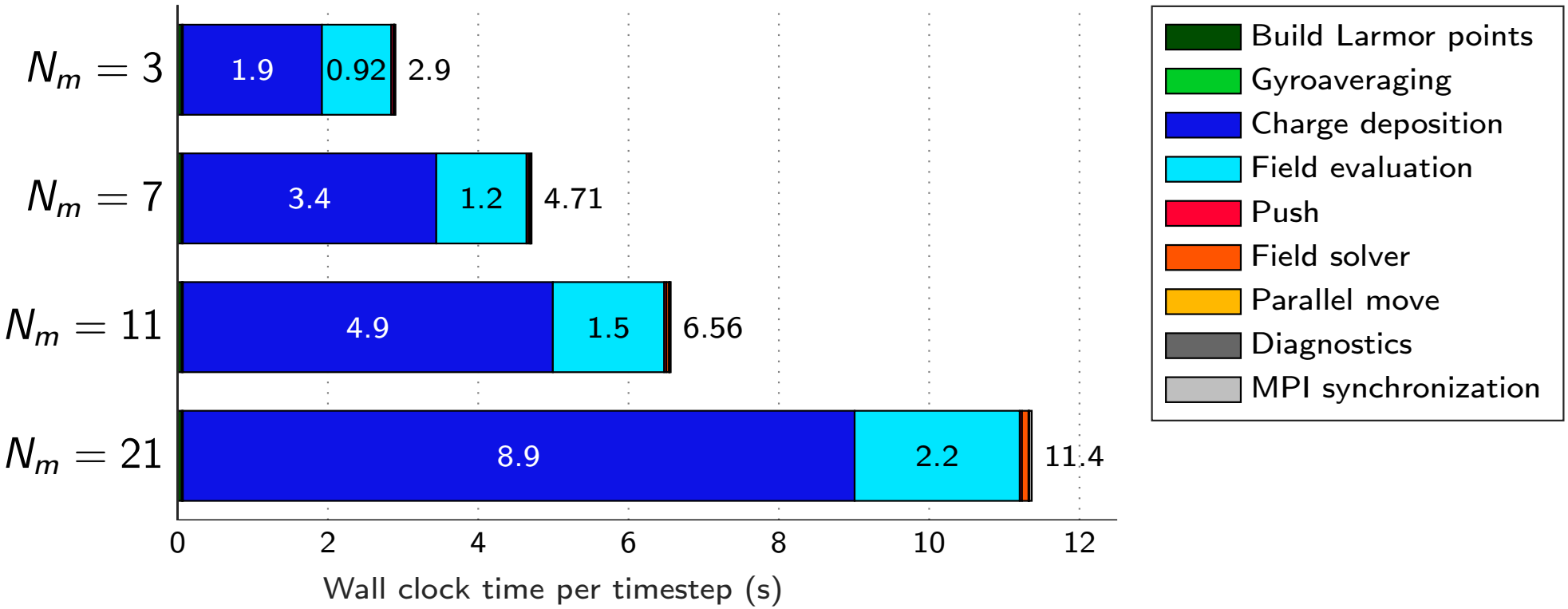
- PIF, 32 nodes, 128M particles, 128 radial cells, $N_m = 11$



- Wall clock time roughly proportional to N_n

Scanning number of poloidal modes

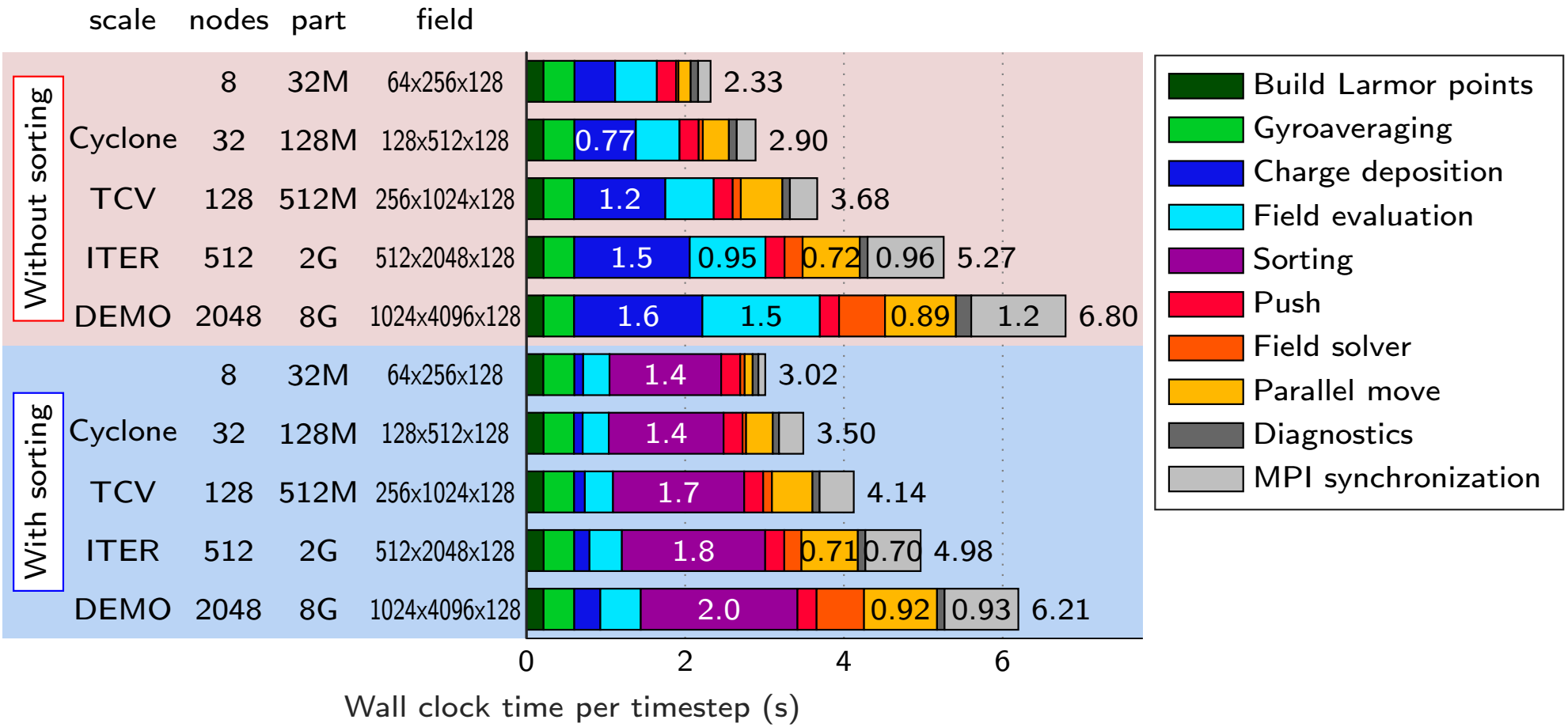
- PIF, 32 nodes, 128M particles, 128 radial cells, $N_n = 15$



- Wall clock time scales less than linearly with N_m because exponential is computed with prefactor successive multiplications.

Parallel scalability of PIC on CPU

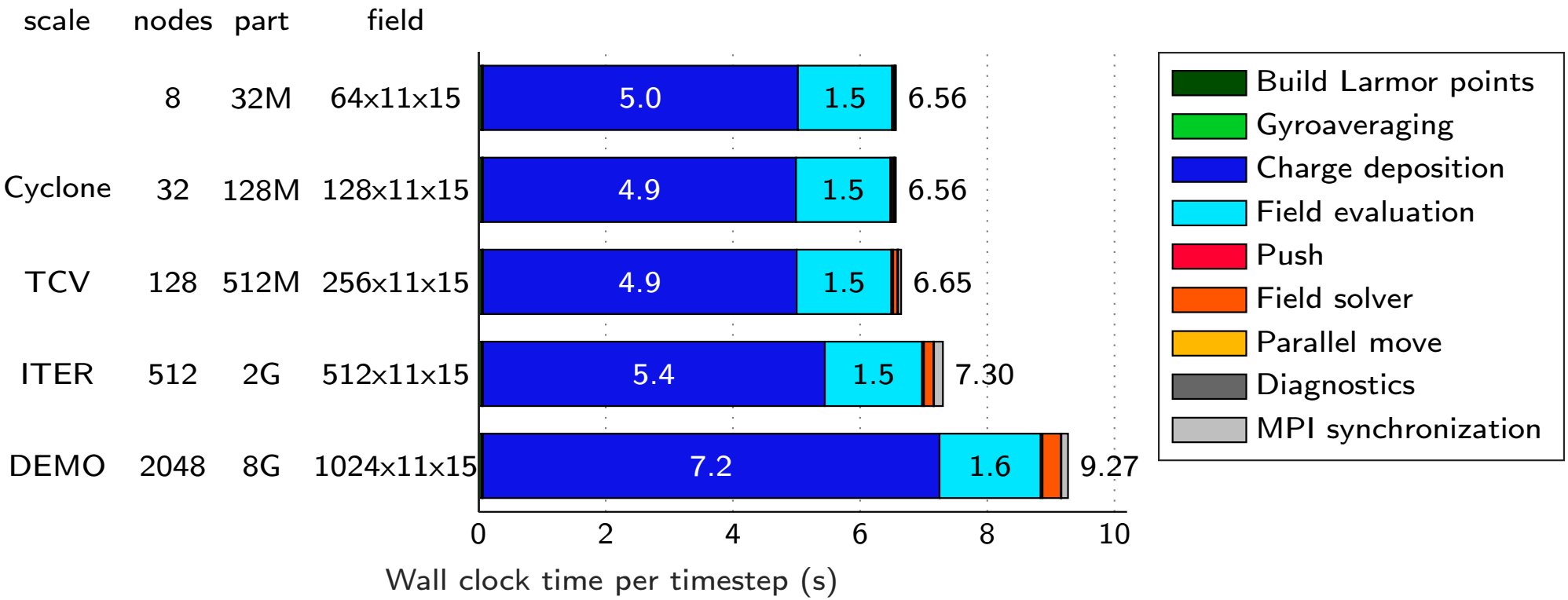
- Physics-based weak scaling, PIC, quad. splines, 12 OpenMP threads per node



- Sorting useful for large size systems
- Number of particle per cell decreases, could be avoided with 2D domain decomposition (see A. Jocksch's presentation)

Parallel scalability of PIF on GPU

- Physics-based weak scaling, PIF, quadratic splines, 1 GPU per node



- Number of particles per mode increases ⇒ atomic conflicts

Conclusion

- What to take home:
  - **Spline order**: choice for best time-to-solution depends on required precision.
  - **PIC or PIF**: PIF is more precise than PIC, but can be slower if many Fourier modes are kept.
  - **CPU or GPU**: GPU is about 3 times faster than CPU for PIC (on Piz Daint), and essential for PIF.
- Future work:
  - Apply those results to application runs with ORB5 (see A. Scheinberg's poster and E. Lanti's presentation)

Bibliography

- T.M. Tran, K. Appert,M. Fivaz, G. Jost, J. Vaclavik and L. Villard
  Global gyrokinetic simulation of ion-temperature-gradient-driven instabilities using particles
  Theory of Fusion Plasmas, Int. Workshop (Editrice Compositori, Bologna, Societa Italiana di Fisica), 45, 1999
- S. Jolliet
  Gyrokinetic particle-in-cell global simulations of ion-temperature-gradient and collisionless-trapped-electron-mode turbulence in tokamaks
  PhD thesis, EPFL, 2009
- A. Fasoli, S. Brunner, W.A. Cooper, J.P. Graves, P. Ricci, O. Sauter and L. Villard
  Computational challenges in magnetic-confinement fusion physics
  Nature Physics, 12(5), 411-423, 2016
- T. Görler, N. Tronko, W.A. Hornsby, A. Bottino, R. Kleiber, C. Norscini, V. Grandgirard, F. Jenko, and E. Sonnendrücker
  Intercode comparison of gyrokinetic global electromagnetic modes
  Physics of Plasmas, 23(7), 072503, 2016

Power balance

- Balance equation $\frac{d\mathcal{E}_{\text{pot}}}{dt} = -\frac{d\mathcal{E}_{\text{kin}}}{dt}$
  - Potential energy $\mathcal{E}_{\text{pot}} = \int \frac{1}{2} e\langle\phi\rangle f \; d^3\mathbf{x} \; d^3\mathbf{v}$
  - Kinetic energy $\mathcal{E}_{\text{kin}} = \int \left(\frac{1}{2} m v_{/\!/}^2 + \mu B\right) f \; d^3\mathbf{x} \; d^3\mathbf{v}$
- Linear growth rate
  - $\gamma_{\text{pot}} = \frac{1}{2} \frac{d \ln(\mathcal{E}_{\text{pot}})}{dt}$ computed by finite difference in time
  - $\gamma_{\text{kin}} = \frac{-1}{2\mathcal{E}_{\text{pot}}} \frac{d\mathcal{E}_{\text{kin}}}{dt}$ computed instantaneously at each timestep

Exponential computation

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
SWISS PLASMA CENTER

Algorithm 1 Explicit method

for $n \in [n_{\min}, n_{\max}]$ **do**
  einz $= \exp(2\pi \mathrm{i} nz/L_z)$
  for $m \in [-nq-\Delta m, -nq+\Delta m]$ **do**
    eimy $= \exp(2\pi \mathrm{i} my/L_y)$
    ... $=$ ...$\times$ eimy $\times$ einz
  end for
end for

Algorithm 2 Prefactor method

eiz $= \exp(2\pi \mathrm{i} z/L_z)$
einz $=$ eiz^$n_{\min}$
for $n \in [n_{\min}, n_{\max}]$ **do**
  eiy $= \exp(2\pi \mathrm{i} y/L_y)$
  eimy $=$ eiy^$(-nq-\Delta m)$
  for $m \in [-nq-\Delta m, -nq+\Delta m]$ **do**
    ... $=$ ...$\times$ eimy $\times$ einz
    eimy $=$ eimy $\times$ eiy
  end for
  einz $=$ einz $\times$ eiz
end for