Solving the high-dimensional Vlasov equation with deal.II and hyper.deal

Eighth deal.II Users and Developers Workshop

Peter Munch[1,2], Katharina Kormann[3], Martin Kronbichler[1]

[1]Institute for Computational Mechanics, Technical University of Munich, Germany

[2]Institute of Materials Research, Materials Mechanics, Helmholtz-Zentrum Geesthacht, Germany

[3]Max Planck Institute for Plasma Physics and Department of Mathematics, Technical University of Munich, Germany

May 26, 2020

Overview

Vlasov equation: non-linear, high-dimensional, hyperbolic PDE

$$\frac{\partial f}{\partial t} + \vec{v} \cdot \nabla_{\vec{x}} f + \vec{a}(t, f, \vec{x}, \vec{v}) \cdot \nabla_{\vec{v}} f = 0 \quad \leftrightarrow \quad \frac{\partial f}{\partial t} + \begin{pmatrix} \vec{v} \\ \vec{a}(t, f, \vec{x}, \vec{v}) \end{pmatrix} \cdot \begin{pmatrix} \nabla_{\vec{x}} \\ \nabla_{\vec{v}} \end{pmatrix} f = 0$$

... with additional term with derivative of the solution with respect to $\vec{v}$

Table of contents:

9.2 = new deal.II feature (release 9.2)

Overview (cont.)

hyper.deal

- FEM library with specialized algorithms for high dimensions ($2 \leq d \leq 6$)
- deal.II-based & 9.2-ready
- freely available under LGPL 3.0 license
- hosted at `https://github.com/hyperdeal/hyperdeal`
- 2 tutorials: `examples` $\rightarrow$ `advection` & `examples` $\rightarrow$ `vlasov_poisson`

Munch, Kormann, Kronbichler, hyper.deal: An efficient, matrix-free finite-element library for high-dimensional partial differential equations, arXiv:2002.08110, 2020

Part 1:

Application: computational plasma physics

Goal

Describe the evolution of a plasma and its interaction in magnetic fields. A field of application is fusion energy research, in which the plasma in fusion reactors (e.g., tokamak and stellarator) are investigated.

Mathematical descriptions:

1. particle model: description of the motion of each particle ▷ n-body problem

$$\frac{\partial^2 x_i}{\partial t^2} = \frac{q_i}{m_i}\left(\vec{E}(t,\vec{x}) + \vec{v} \times \vec{B}(t,\vec{x})\right)$$

2. kinetic model: described by a distribution function f, which evolves according to the Vlasov equation coupled to a system of Maxwell's equations
3. fluid model: coupling of the Navier–Stokes equations to a system of Maxwell's equations

Vlasov–Maxwell/Poisson equations

Vlasov equation: with a single particle species with charge q and mass m

$$\frac{\partial f}{\partial t} + \vec{v} \cdot \nabla_{\vec{x}} f + \vec{a}(t, f, \vec{x}, \vec{v}) \cdot \nabla_{\vec{v}} f = 0 \qquad \vec{a}(t, f, \vec{x}, \vec{v}) = \frac{q}{m}\left(\vec{E}(t, \vec{x}) + \vec{v} \times \vec{B}(t, \vec{x})\right)$$

which is coupled to the Maxwell's equations (or in simple cases to the Poisson equation) for the self-consistent fields.

Part 2:

Discretization with discontinuous Galerkin methods

Discontinuous Galerkin discretization

- short-hand notation:

$$\frac{\partial f}{\partial t} + \vec{a} \cdot \nabla f = 0 \quad \text{with} \quad \vec{a} := \begin{pmatrix} \vec{v} \\ \vec{a} \end{pmatrix} \quad \text{and} \quad \nabla := \begin{pmatrix} \nabla_{\vec{x}} \\ \nabla_{\vec{v}} \end{pmatrix}$$

- discontinuous Galerkin discretization expressed in reference space:

$$\left(g, |\mathcal{J}| \frac{\partial f}{\partial t} \right)_{\Omega_0^{(e)}} - \left(\mathcal{J}^{-T} \nabla_{\vec{\xi}} g, \, |\mathcal{J}| \vec{a} f \right)_{\Omega_0^{(e)}} + \left\langle g, \, |\mathcal{J}| \, \vec{n} \cdot (\vec{a} f)^* \right\rangle_{\Gamma_0^{(e)}} = 0$$

Part 3:

Solving with deal.II and hyper.deal

Limitation of with deal.II for high dimensions

To solve an advection equation in `deal.II`: ▷ step-67 9.2

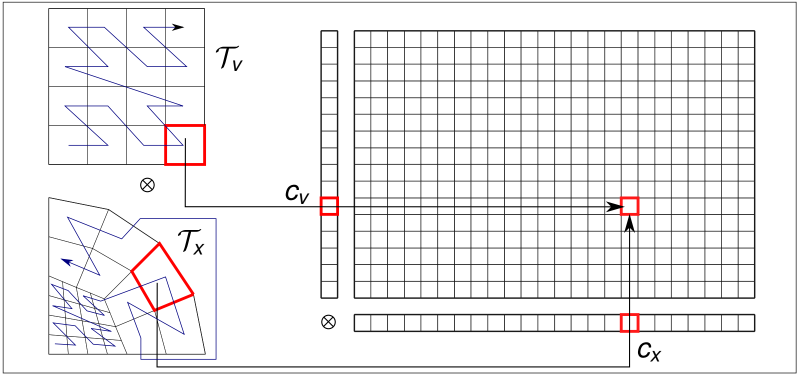
- distributed triangulation:
  - `parallel::distributed::Triangulation<dim>`
  - `parallel::fullydistributed::Triangulation<dim>` 9.2
- `FE_DGQ<dim>(k)` and `QGauss<dim>(k+1)`
- `MatrixFree<dim>` infrastructure

Problem: dim $\leq$ 3! In our case, dim=2, 3, **4**, **5**, ...!

Question: What can we reuse from deal.II?

Solution: tensor product of two (deal.II) triangulations! $\rightarrow$ hyper.deal

hyper.deal: triangulation

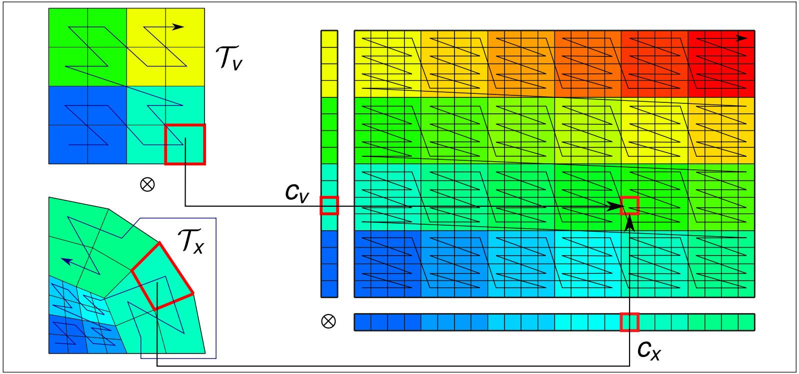


$$\mathcal{T} := \mathcal{T}_{\vec{x}} \otimes \mathcal{T}_{\vec{v}}$$

$$c := (c_{\vec{x}}, c_{\vec{v}})$$

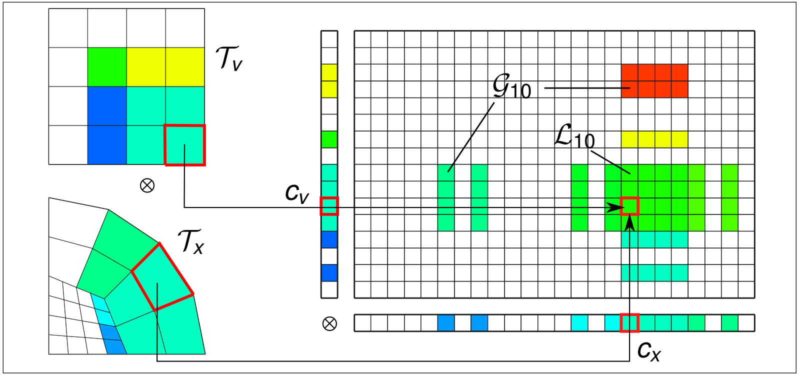
hyper.deal: partitioning

- partition $\mathcal{T}_x$ and $\mathcal{T}_v$ independently
- checkerboard partitioning



$$\mathcal{T}^{f(i,j)} := \mathcal{T}^i_{\vec{x}} \otimes \mathcal{T}^j_{\vec{v}} \qquad \text{with} \qquad f(i,j) := j \cdot p_{\vec{x}} + i$$

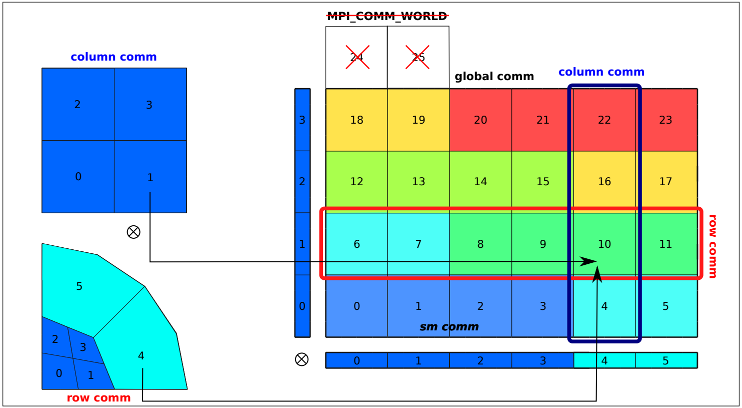
hyper.deal: partitioning (cont.)



- number of neighbors: larger
- neighboring cells: disjunct

hyper.deal: partitioning (cont.)

- virtual topology
- shared memory (MPI-3.0)
- consensus algorithms 9.2



hyper.deal: shape functions and quadrature

- due to tensor-product structure, an extension of the shape function to higher dimensions is simple:

$$\mathcal{P}_k^{d_{\vec{x}}+d_{\vec{v}}} = \underbrace{\mathcal{P}_k^1 \otimes \cdots \otimes \mathcal{P}_k^1}_{\times d_{\vec{x}}} \otimes \underbrace{\mathcal{P}_k^1 \otimes \cdots \otimes \mathcal{P}_k^1}_{\times d_{\vec{v}}} = \mathcal{P}_k^{d_{\vec{x}}} \otimes \mathcal{P}_k^{d_{\vec{v}}}$$

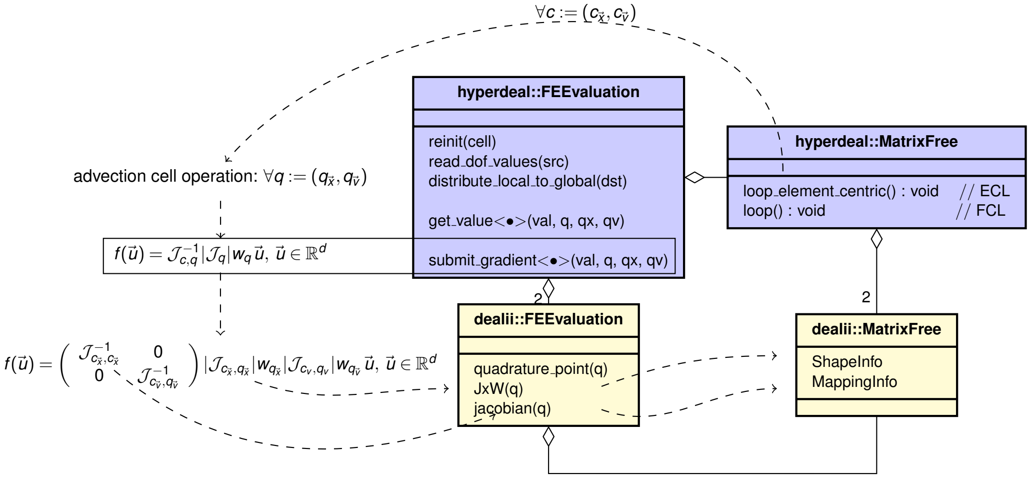
... with $\mathcal{P}_k^{d_{\vec{x}}} = \underbrace{\mathcal{P}_k^1 \otimes \cdots \otimes \mathcal{P}_k^1}_{\times d_{\vec{x}}}$ and $\mathcal{P}_k^{d_{\vec{v}}} = \underbrace{\mathcal{P}_k^1 \otimes \cdots \otimes \mathcal{P}_k^1}_{\times d_{\vec{v}}}$

- the same is true for the quadrature rules
- as a consequence, the mapping data from lower-dimensional spaces (e.g., $\mathcal{J}_x$ and $\mathcal{J}_v$) from deal.II can be reused

hyper.deal: matrix-free infrastructure

- construct two `dealii::MatrixFree` instances
- loop over a pair of cell batches
  - cell loop, face-centric loop, element-centric loop 9.2
- access mapping information via `FEEvaluation` and `FEFaceEvaluation`

hyper.deal: matrix-free infrastructure (cont.)



hyper.deal: matrix-free infrastructure (cont.)

deal.II-like interface:

```
// create hyperdeal::FEEvaluation
FEEvaluation<dim_x, dim_v, degree, n_points, Number, VectorizedArrayType> phi(
  matrix_free, dof_no_x, dof_no_v, quad_no_x, quad_no_v); // configure underlying FEEvaluation

// loop over cells (i.e. cell pairs)
matrix_free.cell_loop([&](const auto &, auto & dst, const auto &src, const auto cell){
    // reinit
    phi.reinit(cell);
    phi.read_dof_values(src);
    // evaluate

    // loop over quadrature points
    for (unsigned int qv = 0, q = 0; qv < phi.n_q_points_v; qv++)
      for (unsigned int qx = 0; qx < phi.n_q_points_x; qx++, q++)
        {
          // operation on quadrature point level
          const auto q_point = phi.get_quadrature_point(qx, qv);
          /*...*/
        }

    // integrate
    phi.distribute_local_to_global(dst);
  }, dst, src);
```

hyper.deal: vectorization

- vectorization over a batch of elements
- constructing a batch of elements

$$\texttt{VectoriedArray<T, N>} = \underbrace{\texttt{VectoriedArray<T, N>}}_{\text{x cell batch} \rightarrow \text{vectorized}} \times \underbrace{\texttt{VectoriedArray<T, 1>}}_{\text{v cell batch} \rightarrow \text{not vectorized}}$$

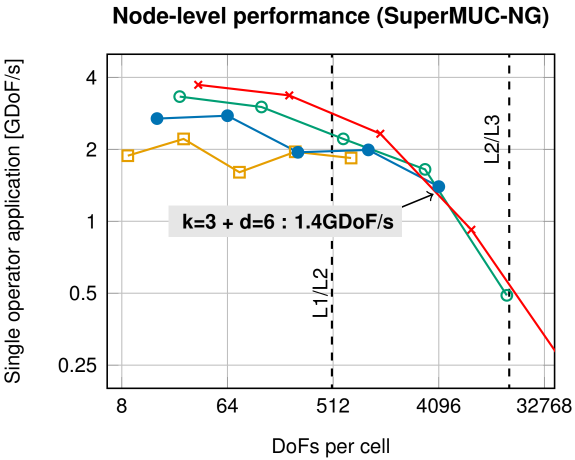
... via the new template argument of `VectorizedArray` 9.2

- setup of internal data structures of x-/v-MatrixFree appropriately

```
MatrixFree<dim, T, VT>
```

... via the new template argument of `VT := VectorizedArray<T, N>` 9.2

hyper.deal: node-level performance



- setup:
  tensor product of two hyper-cubes
  (see: `examples` $\rightarrow$ `advection`)
- challenge:
  temporal data size of cell batch $\mathcal{O}(k^d)$
- outlook: vectorization within elements

$k = 2$ $k = 3$ $k = 4$ $k = 5$

for different dimensions

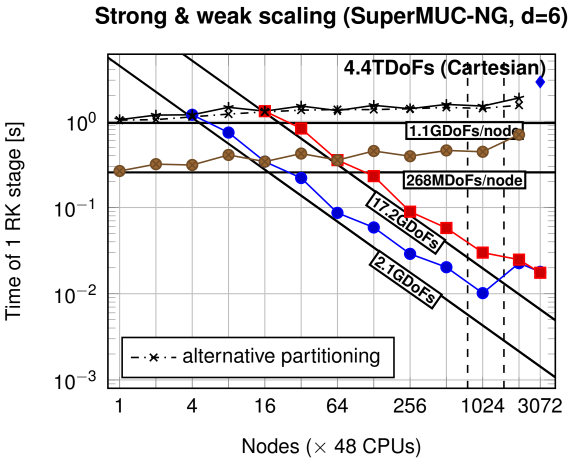
hyper.deal: strong and weak scaling

largest simulation:

- $4.4 \cdot 10^{12} = (2.1 \cdot 10^6)^2 = 128^6$DoFs

challenges:

- communication pattern
- communication data volume



Part 4:

Extension to Vlasov–Poisson equation

Vlasov-Poisson equation

Vlasov equation for electrons in a neutralizing background in the absence of magnetic fields:

$$\frac{\partial f}{\partial t} + \begin{pmatrix} \vec{v} \\ -\vec{E}(t,\vec{x}) \end{pmatrix} \cdot \nabla f = 0,$$

where the electric field can be obtained from the Poisson problem:

$$\rho(t,\vec{x}) = 1 - \int f(t,\vec{x},\vec{v})\,\mathrm{d}v, \qquad -\nabla_{\vec{x}}^2 \phi(t,x) = \rho(t,\vec{x}), \qquad \vec{E}(t,\vec{x}) = -\nabla_{\vec{x}}\phi(t,\vec{x}).$$

▷ full code online: `examples` $\rightarrow$ `vlasov_poisson`

Coupling with a deal.II-based Poisson solver

- in each Runge-Kutta step solve:

$$\rho(t,\vec{x}) = 1 - \int f(t,\vec{x},\vec{v})\,\mathrm{d}v$$

$$\frac{\partial f}{\partial t} + \begin{pmatrix} \vec{v} \\ -\vec{E}(t,\vec{x}) \end{pmatrix} \cdot \nabla f = 0$$

$$\nabla^2_{\vec{x}} \phi(t,x) = -\rho(t,\vec{x})$$
$$\vec{E}(t,\vec{x}) = -\nabla_{\vec{x}} \phi(t,\vec{x})$$

$\Omega_x \times \Omega_v \to$ hyper.deal

$\Omega_x \to$ deal.II $\to$ geometric multigrid

... the same approach is applicable to the Vlasov–Maxwell equations!

Part 5:

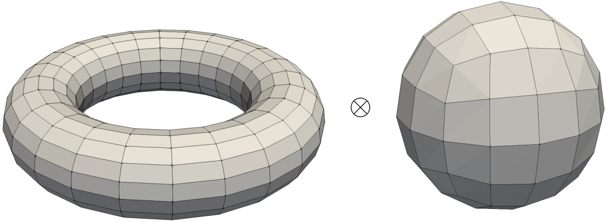
Conclusions & outlook

Conclusions & outlook

Conclusions:

- possibility to solve high-dimensional PDEs with deal.II + hyper.deal
- tensor product of triangulations & on-the-fly combination of mapping
- easy access to deal.II data structures and possibility of coupling to deal.II
- clear separation of responsibilities (low vs. high dimensions)

Conclusions & outlook (cont.)



Outlook:

- new features: AMR, unstructured meshes
- other applications ($\rightarrow$ new tutorials?): cosmic microwave background radiation

Feel free to contribute new features and new tutorials!