HiFlow^3: an Open Source Multi-Purpose Finite Element Software

Vincent Heuveline, Chen Song

Heidelberg University

chen.song@iwr.uni-heidelberg.de

01 November 2019

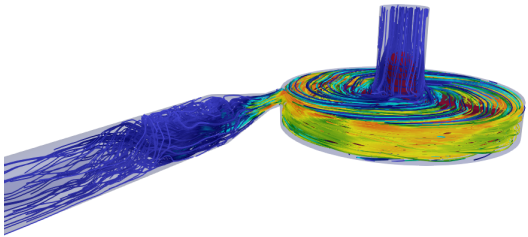
Numerics for PDEs

Most of physical phenomena can be modeled by a system of partial differential equations (PDEs).

Numerics for PDEs

Most of physical phenomena can be modeled by a system of partial differential equations (PDEs).

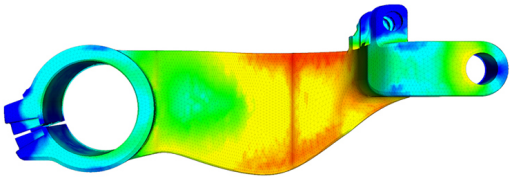
- Fluid dynamics.



Numerics for PDEs

Most of physical phenomena can be modeled by a system of partial differential equations (PDEs).

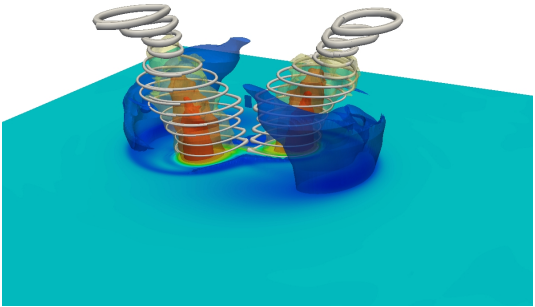
- Fluid dynamics.
- Structure dynamics.



Numerics for PDEs

Most of physical phenomena can be modeled by a system of partial differential equations (PDEs).

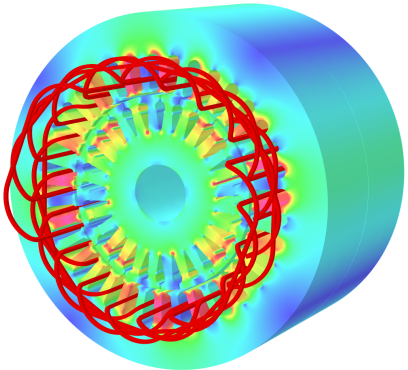
- Fluid dynamics.
- Structure dynamics.
- Meteorology.



Numerics for PDEs

Most of physical phenomena can be modeled by a system of partial differential equations (PDEs).

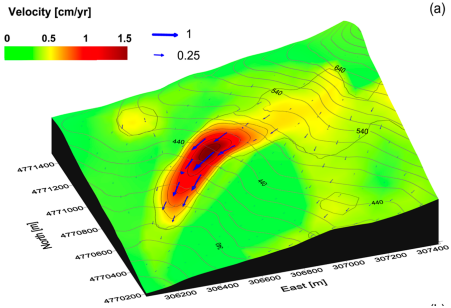
- Fluid dynamics.
- Structure dynamics.
- Meteorology.
- Electromagnetic dynamics.



Numerics for PDEs

Most of physical phenomena can be modeled by a system of partial differential equations (PDEs).

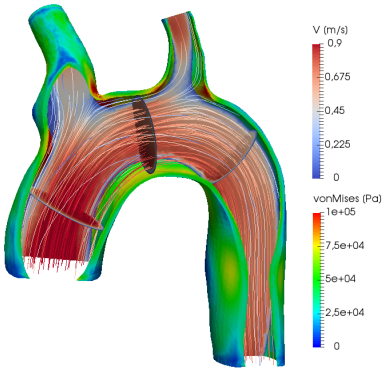
- Fluid dynamics.
- Structure dynamics.
- Meteorology.
- Electromagnetic dynamics.
- Geophysics.



Numerics for PDEs

Most of physical phenomena can be modeled by a system of partial differential equations (PDEs).

- Fluid dynamics.
- Structure dynamics.
- Meteorology.
- Electromagnetic dynamics.
- Geophysics.
- Bio-engineering.
- etc.

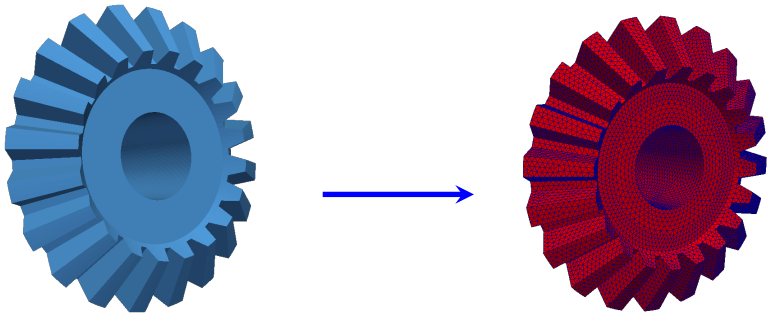


Numerics for PDEs

Three common frameworks for numerical solution of PDEs:

- Finite element method (FEM).
- Finite volume method (FVM).
- Finite difference method (FDM).

The philosophy is the same: **Divide and Conquer**.



geometry

generated mesh

Numerics for PDEs: Poisson Equation (example)

Strong formulation

$$
\begin{aligned}
-\Delta u &= f, && \text{in } \Omega, \\
u &= g_D, && \text{on } \partial\Omega_D, \\
\nabla u \cdot \boldsymbol{n} &= g_N, && \text{on } \partial\Omega_N.
\end{aligned}
$$

Weak (Variational) formulation

find $u \in V$, such that:

$$
a(u, v) = L(v), \qquad \forall v \in V,
$$

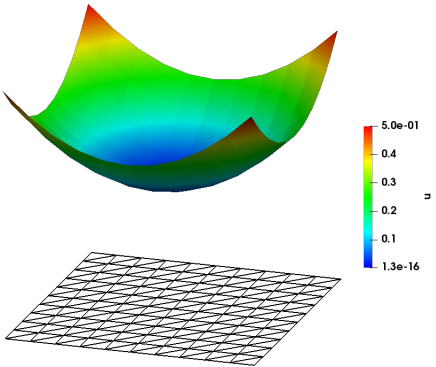
where V is a suitable function space, and

$$
\begin{aligned}
a(u, v) &= \int_\Omega \nabla u \cdot \nabla v d\boldsymbol{x}, \\
L(v) &= \int_\Omega f v d\boldsymbol{x} + \int_{\partial\Omega_N} g_N v d\boldsymbol{x}.
\end{aligned}
$$

Numerics for PDEs: Poisson Equation (example)

In this talk, we consider an example of Poisson Equation with following conditions:

- Physical domain: $\Omega \in [0,1] \times [0,1]$.
- Analytical solution: $u_e(x,y) = (x-0.5)^2 + (y-0.5)^2$.
- $f = -4$.
- $g_N = 0$.
- $g_D = (x-0.5)^2 + (y-0.5)^2$, on $\partial\Omega$.



$\Omega \in [0,1] \times [0,1]$

Numerics for PDEs: Poisson Equation (example)

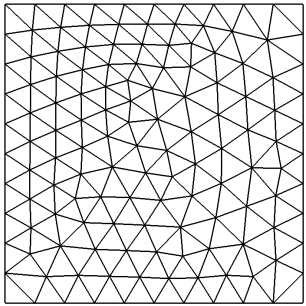
How can we solve this PDE by using some numerical methods?

- Describe physical domain.
- Approximate functions locally.
- Build linear system.
- Solve linear system.
- Parallelization.
- Visualize solution.
- Quantifying uncertainties in the model.

Numerics for PDEs: Poisson Equation (example)

How can we solve this PDE by using some numerical methods?

- Describe physical domain. $\Rightarrow$ **Mesh**
- Approximate functions locally.
- Build linear system.
- Solve linear system.
- Parallelization.
- Visualize solution.
- Quantifying uncertainties in the model.

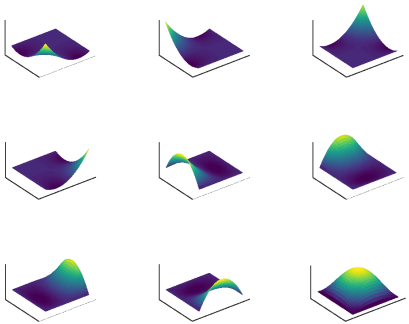


$\Omega \in [0,1] \times [0,1]$

Numerics for PDEs: Poisson Equation (example)

How can we solve this PDE by using some numerical methods?

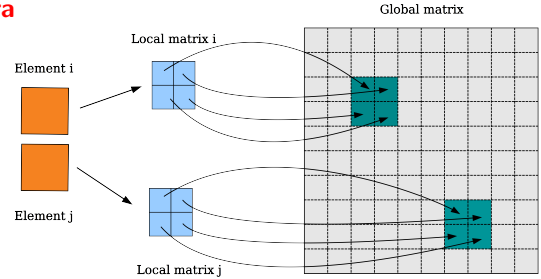
- Describe physical domain. $\Rightarrow$ **Mesh**
- Approximate functions locally. $\Rightarrow$ **FEM**
- Build linear system.
- Solve linear system.
- Parallelization.
- Visualize solution.
- Quantifying uncertainties in the model.



Numerics for PDEs: Poisson Equation (example)

How can we solve this PDE by using some numerical methods?

- Describe physical domain. $\Rightarrow$ **Mesh**
- Approximate functions locally. $\Rightarrow$ **FEM**
- Build linear system. $\Rightarrow$ **Assembly, Linear Algebra**
- Solve linear system.
- Parallelization.
- Visualize solution.
- Quantifying uncertainties in the model.



Numerics for PDEs: Poisson Equation (example)

How can we solve this PDE by using some numerical methods?

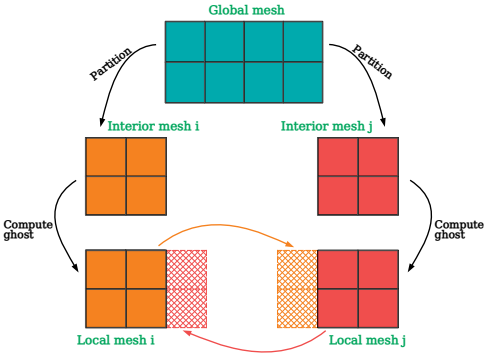
- Describe physical domain. $\Rightarrow$ **Mesh**
- Approximate functions locally. $\Rightarrow$ **FEM**
- Build linear system. $\Rightarrow$ **Assembly, Linear Algebra**
- Solve linear system. $\Rightarrow$ **Linear Solver**
- Parallelization.
- Visualize solution.
- Quantifying uncertainties in the model.

$$Ax = b$$

Numerics for PDEs: Poisson Equation (example)

How can we solve this PDE by using some numerical methods?

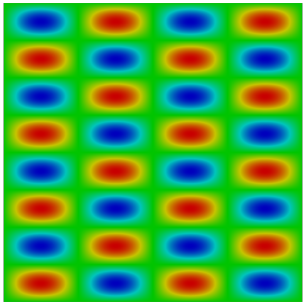
- Describe physical domain. $\Rightarrow$ **Mesh**
- Approximate functions locally. $\Rightarrow$ **FEM**
- Build linear system. $\Rightarrow$ **Assembly, Linear Algebra**
- Solve linear system. $\Rightarrow$ **Linear Solver**
- Parallelization. $\Rightarrow$ **Domain Decomposition (Mesh)**
- Visualize solution.
- Quantifying uncertainties in the model.



Numerics for PDEs: Poisson Equation (example)

How can we solve this PDE by using some numerical methods?

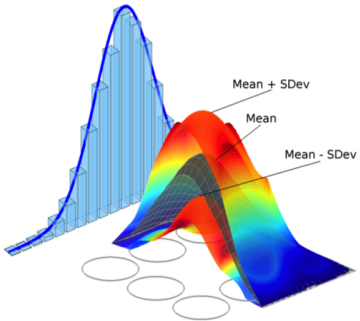
- Describe physical domain. $\Rightarrow$ **Mesh**
- Approximate functions locally. $\Rightarrow$ **FEM**
- Build linear system. $\Rightarrow$ **Assembly, Linear Algebra**
- Solve linear system. $\Rightarrow$ **Linear Solver**
- Parallelization. $\Rightarrow$ **Domain Decomposition (Mesh)**
- Visualize solution. $\Rightarrow$ **Visualization**
- Quantifying uncertainties in the model.



Numerics for PDEs: Poisson Equation (example)

How can we solve this PDE by using some numerical methods?

- Describe physical domain. $\Rightarrow$ **Mesh**
- Approximate functions locally. $\Rightarrow$ **FEM**
- Build linear system. $\Rightarrow$ **Assembly, Linear Algebra**
- Solve linear system. $\Rightarrow$ **Linear Solver**
- Parallelization. $\Rightarrow$ **Domain Decomposition (Mesh)**
- Visualize solution. $\Rightarrow$ **Visualization**
- Quantifying uncertainties in the model. $\Rightarrow$ **UQ**



Numerics for PDEs: Poisson Equation (example)

How can we solve this PDE by using some numerical methods?

- Describe physical domain. $\Rightarrow$ **Mesh**
- Approximate functions locally. $\Rightarrow$ **FEM**
- Build linear system. $\Rightarrow$ **Assembly, Linear Algebra**
- Solve linear system. $\Rightarrow$ **Linear Solver**
- Parallelization. $\Rightarrow$ **Domain Decomposition (Mesh)**
- Visualize solution. $\Rightarrow$ **Visualization**
- Quantifying uncertainties in the model. $\Rightarrow$ **UQ**

Efficient numerics software is needed ...

1. Numerics for PDEs

2. HiFlow^3
   - Mesh
   - DoF/FEM
   - Linear Algebra and Linear Solver
   - Uncertainty Quantification
   - Parallelization
   - Visualization
   - External Library

3. Scalability and Applications
   - Scalability
   - Applications

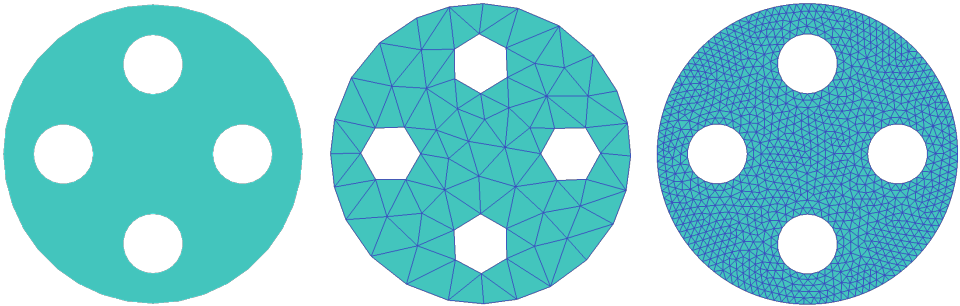
4. Demos

About HiFlow^3

- Parallel finite element software.
- Developed by EMCL (Engineering Mathematics and Computing Lab), IWR, Heidelberg University.
- 16 years of development and experience.
- Open source: European Union Public License, version 1.2 (EUPL-1.2).

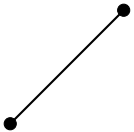
Mesh: HiFlow^3 mesh module

- Describing the computational domain (discrete).
- Each cell represents an individual solution of the PDEs.
- Mesh module is of one core modules in HiFlow^3.
- Grid data handling is also the fundamental issue in finite element software development.
- General interface through the computation in HiFlow^3.

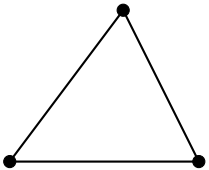


computational domain | coarse mesh | fine mesh

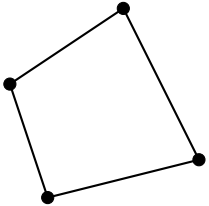
Mesh: supporting mesh cells



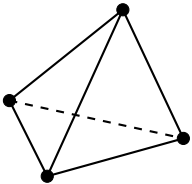
line



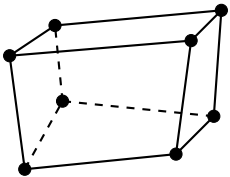
triangle



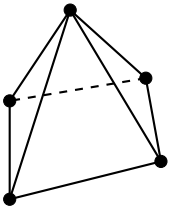
quadrilateral



tetrahedron

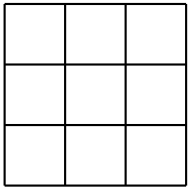


hexahedron

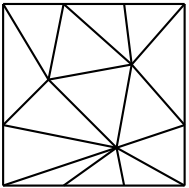


pyramid

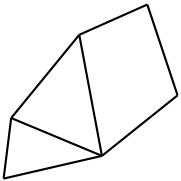
Mesh: different types of cells



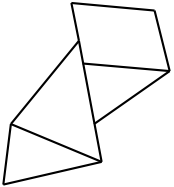
structured



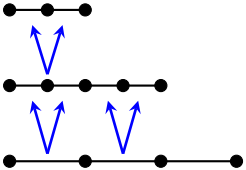
unstructured



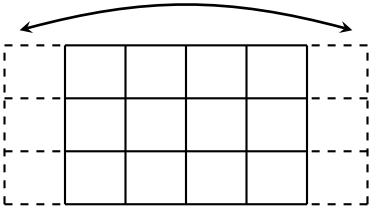
conforming



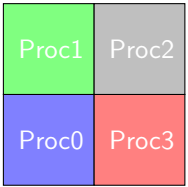
non-conforming



nested



periodic

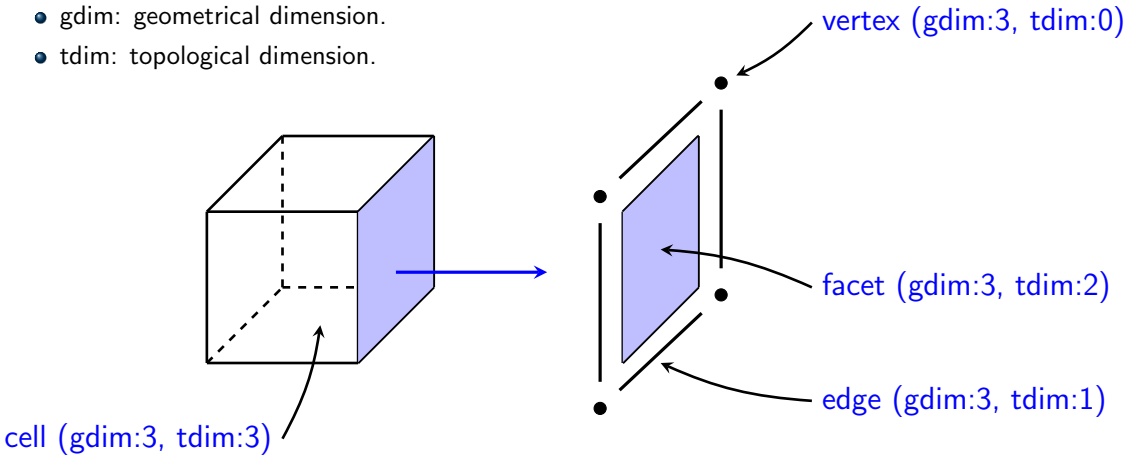


parallel

Mesh: HiFlow^3 mesh cell is a container of entities

Entities are: vertices, line, edges, facet, cell.
Each cell has two kinds of dimension:

- gdim: geometrical dimension.
- tdim: topological dimension.



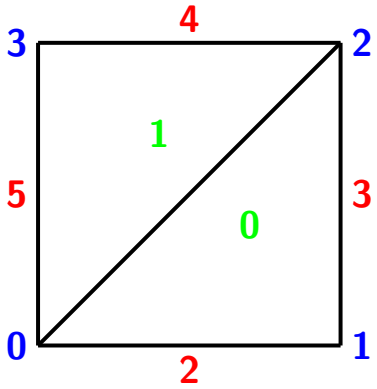
Mesh: HiFlow^3 mesh is a container of mesh cells

- HiFlow^3 mesh contains a set of cells.
- multiple element types.
- graph decomposition.
- conforming, non-conforming.
- hierarchical.
- local refinement.
- customized cell refinement scheme.
- parallel mesh handling.
- dynamic load balancing.
- ...

Mesh: code example

Initial mesh data:

```
4 6 0 0 0
0       0.0     0.0     0.0
1       1.0     0.0     0.0
2       1.0     1.0     0.0
3       0.0     1.0     0.0
0 10 tri 0 1 2
1 10 tri 0 2 3
2 11 line 0 1
3 11 line 1 2
4 11 line 2 3
5 11 line 3 0
```

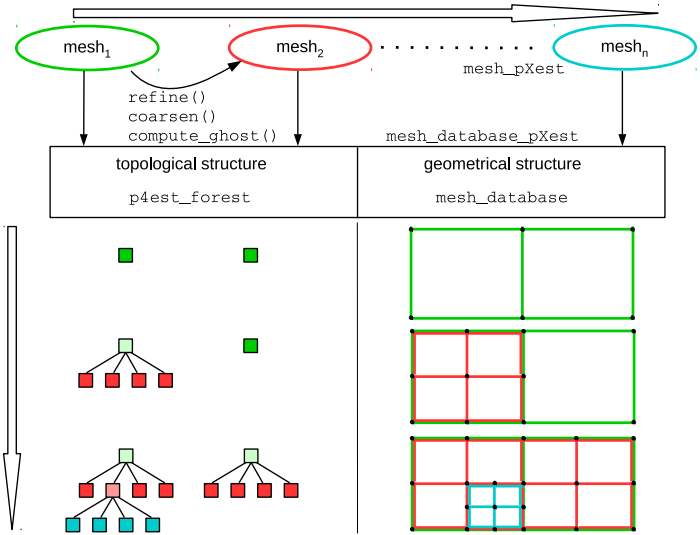


Mesh: code example

Code

```
if (rank_ == MASTER_RANK) {
    master_mesh_ = read_mesh_from_file(mesh_filename, DIMENSION, DIMENSION, 0);
    for (r = 0; r != refinement_level_; ++r) {
        master_mesh_ = master_mesh_->refine();
        ++r;
    }
}
int uniform_ref_steps;
MeshPtr local_mesh = partition_and_distribute(master_mesh, MASTER_RANK, comm_,
uniform_ref_steps);
SharedVertexTable shared_verts;
mesh_ = compute_ghost_cells(*local_mesh, comm_, shared_verts);
```

Mesh: local refinement



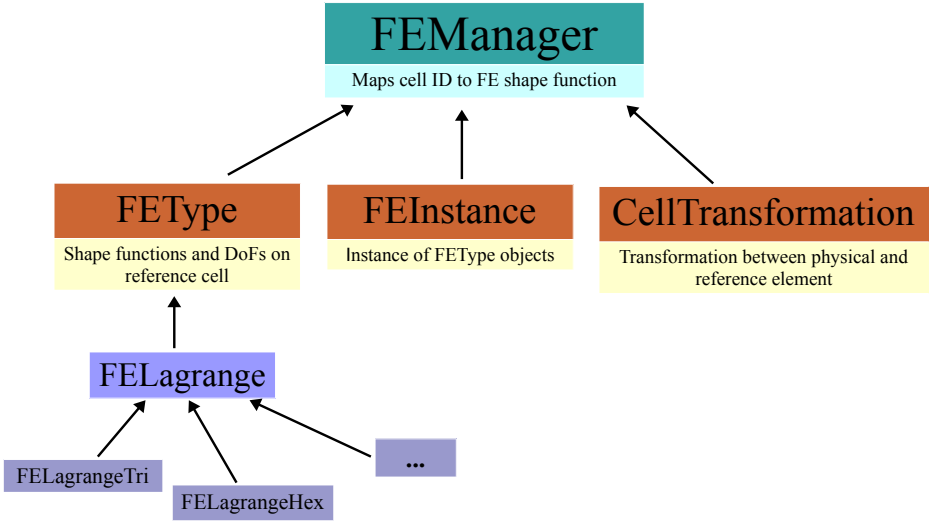
DoF/FEM

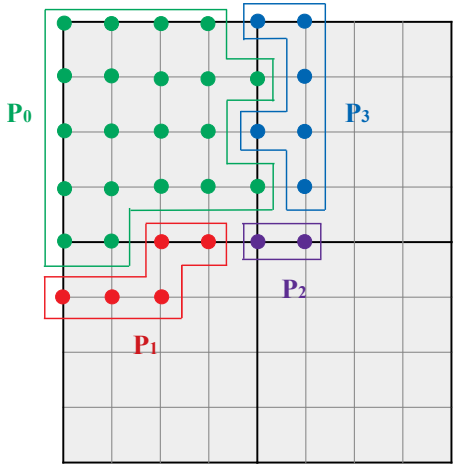
DoF: degrees of freedom.
FEM: finite element method (module).

The important features in DoF/FEM modules are:

- 8 local finite elements. 1D: 1. 2D: 3. 3D: 4.
- High order shape functions, up to 20.
- 3 DoF numbering strategies: `Classic`, `CUTHILL-MCKEE` and `KING`.
- Hanging nodes interpolation.
- Periodic boundary condition.
- Economical Gauss quadrature points and customized possibility.
- Continuous and Discontinuous Galerkin method.
- ...

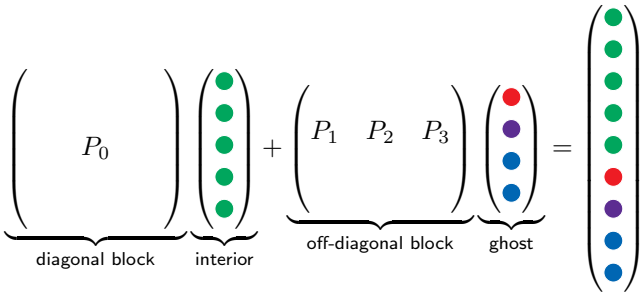
[1] Besides the *pyramid* element, it can only be constructed up to degree 2.


FEManager
Maps cell ID to FE shape function
FEType
Shape functions and DoFs on reference cell
FEInstance
Instance of FEType objects
CellTransformation
Transformation between physical and reference element
FELagrange
FELagrangeTri
FELagrangeHex
...

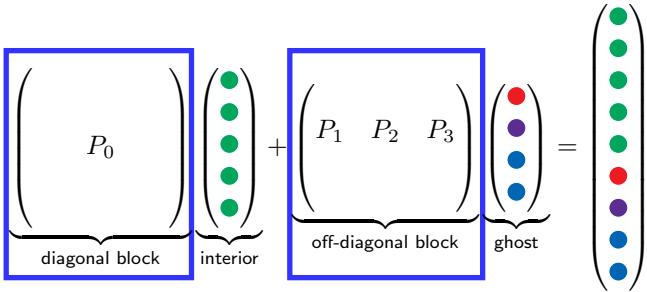


$$\underbrace{\begin{pmatrix} P_0 & & \\ & & \\ & & \end{pmatrix}}_{\text{diagonal block}} \underbrace{\begin{pmatrix} \\ \\ \\ \\ \\ \end{pmatrix}}_{\text{interior}} + \underbrace{\begin{pmatrix} P_1 & P_2 & P_3 \\ & & \\ & & \end{pmatrix}}_{\text{off-diagonal block}} \underbrace{\begin{pmatrix} \\ \\ \\ \end{pmatrix}}_{\text{ghost}}$$

Matrix-Vector multiplication (distributed):

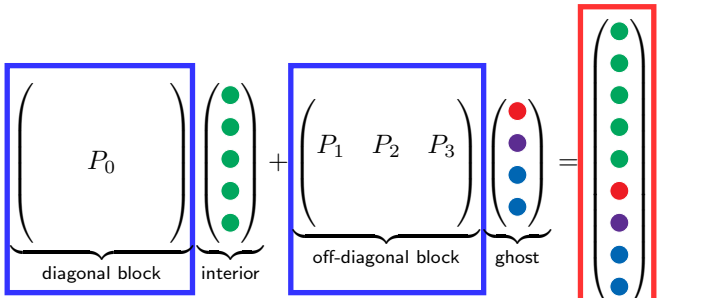


Matrix-Vector multiplication (distributed):

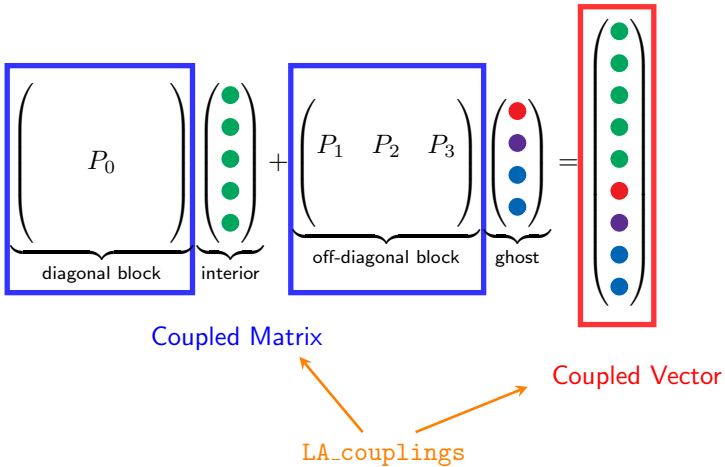


Coupled Matrix

Matrix-Vector multiplication (distributed):



Matrix-Vector multiplication (distributed):



Linear Algebra: code example

Code

```
space_.Init(degrees, mesh_, is_cg, HIFLOW_CLASSIC);
couplings_.Init(comm_);
std::vector< int > global_offsets, ghost_dofs, ghost_offsets;
space_.GetLaCouplings(global_offsets, ghost_dofs, ghost_offsets);
couplings_.InitializeCouplings(global_offsets, ghost_dofs, ghost_offsets);
matrix_.Init(comm_, couplings_);
rhs_.Init(comm_, couplings_);
sol_.Init(comm_, couplings_);
matrix_.InitStructure(
  vec2ptr(sparsity.diagonal_rows), vec2ptr(sparsity.diagonal_cols),
  sparsity.diagonal_rows.size(), vec2ptr(sparsity.off_diagonal_rows),
  vec2ptr(sparsity.off_diagonal_cols), sparsity.off_diagonal_rows.size());
```

Linear Solvers

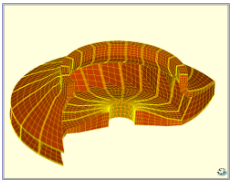
Implemented linear solvers/preconditioners:

- Conjugate Gradient (CG).
- Flexible/Generalized Minimal Residual (F/GMRES).
- BiConjugate Gradient Stabilized (BiCG-Stab).
- Diagonal Block solver - general implementation.
- Schur-Complement block solver/preconditioner.
- Geometric Multigrid.
- Vanka preconditioner/smoother.
- Incomplete LU factorization.
- Block/Jacobi preconditioner.
- Naive parallel preconditioner.

Linear Solvers

In combination with external libraries:

- PETSc.
- Hypre:
  - boomer AMG.
  - bicgstab.
  - cg.
  - gmres.
  - euclid.
  - parasails.
  - pilut.
- SLEPc.
- UMFPACK.
- MUMPS.
- ...

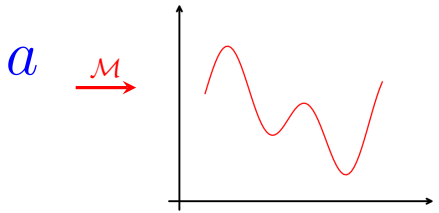


Uncertainty Quantification (UQ)

Uncertainty Quantification (UQ) quantifies the influence on the simulation results, when the input parameters are uncertain.

Single input

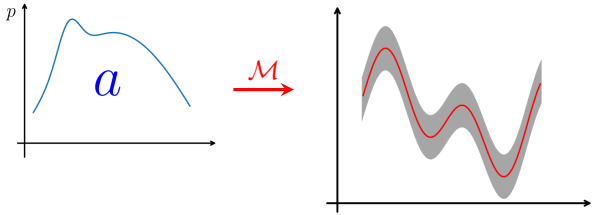
Single output



Deterministic.

Stochastic input

Probability outcome



Uncertainty Quantification.

Uncertainty Quantification (UQ)

Two categories for propagating uncertainties approaches:

- Non-intrusive: Monte-Carlo, Collocation, Sparse-Grid,
- Intrusive: Stochastic Galerkin.

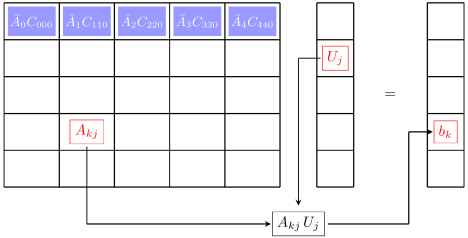
In HiFlow^3, the intrusive method is implemented based on generalized Polynomial Chaos Expansion (gPCE):

$$U(\boldsymbol{x}, t, \boldsymbol{\xi}) = \sum_{i=0}^{\infty} u_i(\boldsymbol{x}, t)\psi_i(\boldsymbol{\xi}) \approx \sum_{i=0}^{P} u_i(\boldsymbol{x}, t)\psi_i(\boldsymbol{\xi}).$$

$\boldsymbol{\xi}$: vector of random variables.
$\psi_i(\boldsymbol{\xi})$: orthogonal Polynomial Chaos basis.
$P := \frac{(M+N_0)!}{M!N_o!}$.
$M = \dim(\boldsymbol{\xi})$.
N_o: polynomial order.

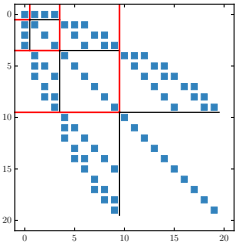
Uncertainty Quantification (UQ): Linear Algebra - Preconditioner

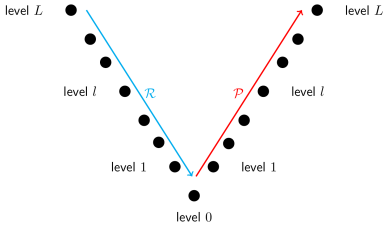
PCE Matrix-Vector multiplication:



$$A_{kj}U_j := \sum_{i=0}^{P} C_{ijk}\bar{A}_i U_j = b_k,$$
$$j,k = 0,...,P.$$

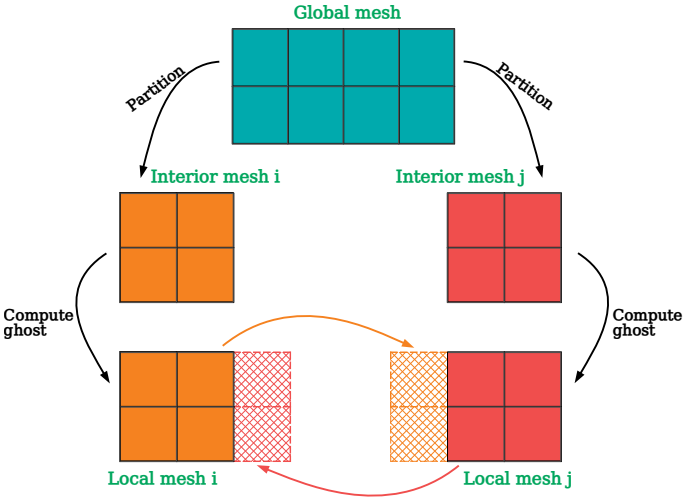
PCE Multilevel preconditioner:





Parallelization

Parallelization starts with mesh decomposition:



Visualization

In order to visualize the numerical solution, HiFlow^3 follows the Parallel VTK Unstructured Data.

- pvtu: summary file of parallel vtu data.
- vtu: containing the mesh data and solution information.
  - points.
  - cells.
  - point data.
  - cell data.
  - ...

Visualization

In order to visualize the numerical solution, HiFlow^3 follows the Parallel VTK Unstructured Data.

- pvtu: summary file of parallel vtu data.
- vtu: containing the mesh data and solution information.
  - points.
  - cells.
  - point data.
  - cell data.
  - ...

However, for large computations (time-dependent, big amount of processors), parallel .vtu format results numerous files. $\Rightarrow$ Difficult to manage on clusters.

Visualization

In order to visualize the numerical solution, HiFlow^3 follows the Parallel VTK Unstructured Data.

- pvtu: summary file of parallel vtu data.
- vtu: containing the mesh data and solution information.
  - points.
  - cells.
  - point data.
  - cell data.
  - ...

However, for large computations (time-dependent, big amount of processors), parallel .vtu format results numerous files. $\Rightarrow$ Difficult to manage on clusters.

HiFlow^3 offers also XDMF format (eXtensible Data Model and Format):

- separating light data and heavy data.
- light data: XML.
- heavy data: DHF5.

Visualization: code example

Code

```
int num_intervals = 2;
ParallelCellVisualization< double > visu(space_, num_intervals, comm_, MASTER_RANK);
std::stringstream name;
name << "solution" << refinement_level_;
visu.visualize(FeEvalOnCellScalar< LAD >(space_, *(sol_)), "u");
visu.write(name.str());
```

External Library

Minimum requirements:

- CMake: cross-platform build managing.
- TinyXml2: operating system-independent XML parser.
- Compiler: GCC or Intel.
- MPI: OpenMPI or Intel MPI.
- Boost: C++ Boost.

Optional:

- OpenMP: shared memory multiprocessing.
- CUDA, OpenCL: GPU accelerators.
- CLABS, CLAPACK, MKL, ATLAS: linear algebra routines and linear solvers.
- METIS, ParMETIS: graph partitioner, mesh decomposition.
- MUMPS: parallel sparse direct solver.
- UMFPACK: multifrontal LU factorization.

External Library

- Hypre: linear solvers and multigrid methods.
- PETSc: portable and extensible toolkit for scientific computation.
- SLEPc: PETSc based eigenvalue solver.
- HDF5: large data management and storage.
- p4est: dynamic management of adaptive octrees.
- ILU++: incomplete LU factorization.
- gaussq: Gauss quadrature for numerical integration.

Additionally:

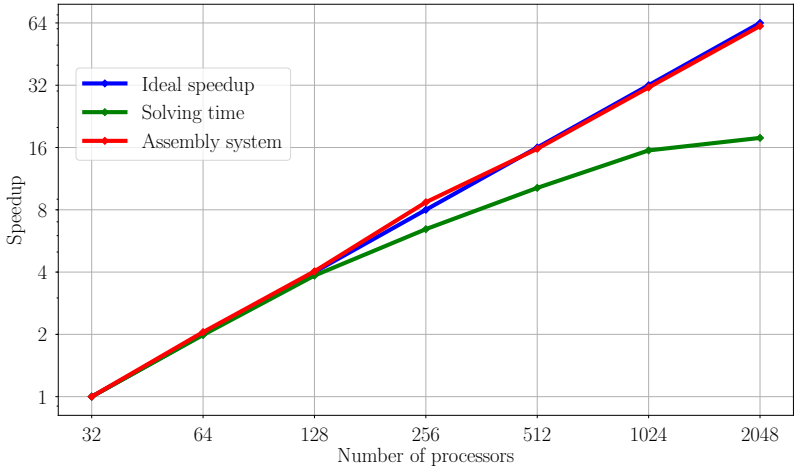
- Paraview: visualization.

Scalability: Poisson equation

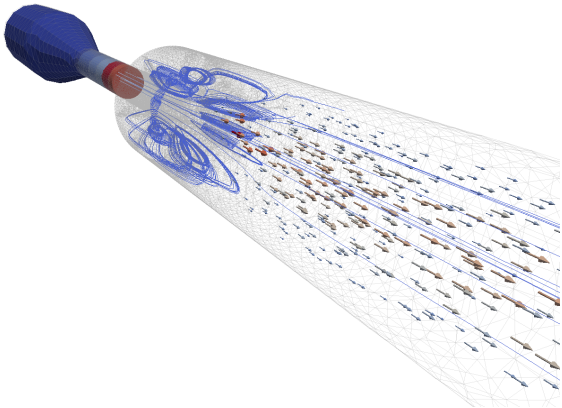
- Unit cube, 8 refinements.
- Discretization with linear elements.
- About 17 Mio. unknowns.
- CG iterative linear solver.
- Algebraic Multigrid preconditioner.

Scalability: Poisson equation

Scaling of Poisson equation on **BWforCluster**, Heidelberg University:



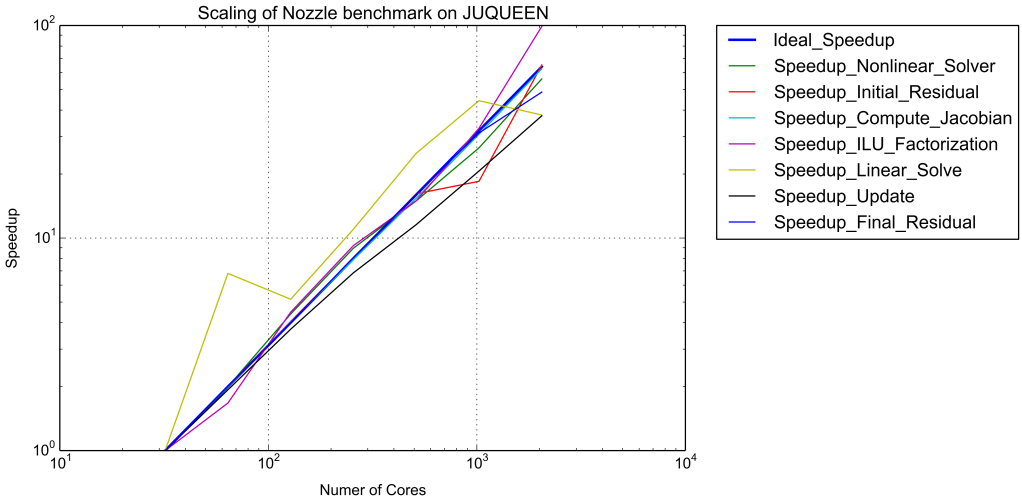
Scalability: nozzle benchmark



- Incompressible Navier-Stokes equations.
- Discretization with P2/P1 elements.
- about 8 Mio. unknowns.
- Block preconditioning with ILU++.
- GMRES iterative linear solver.
- Newton method.

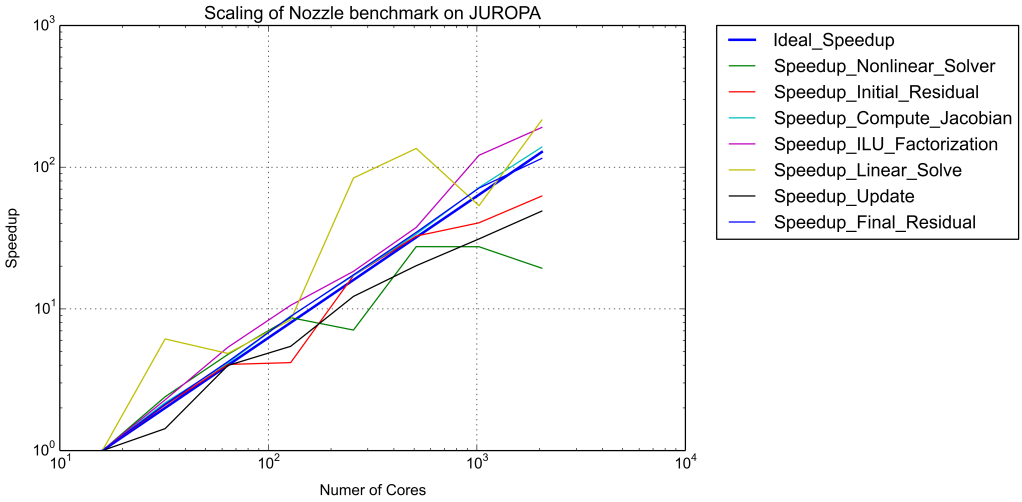
Scalability: nozzle benchmark

Scaling of Nozzle benchmark on **JUQUEEN**, FZ Jülich:



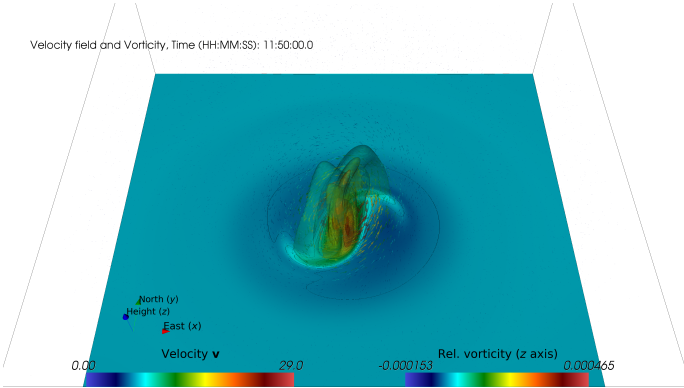
Scalability: nozzle benchmark

Scaling of Nozzle benchmark on **JUROPA**, FZ Jülich:



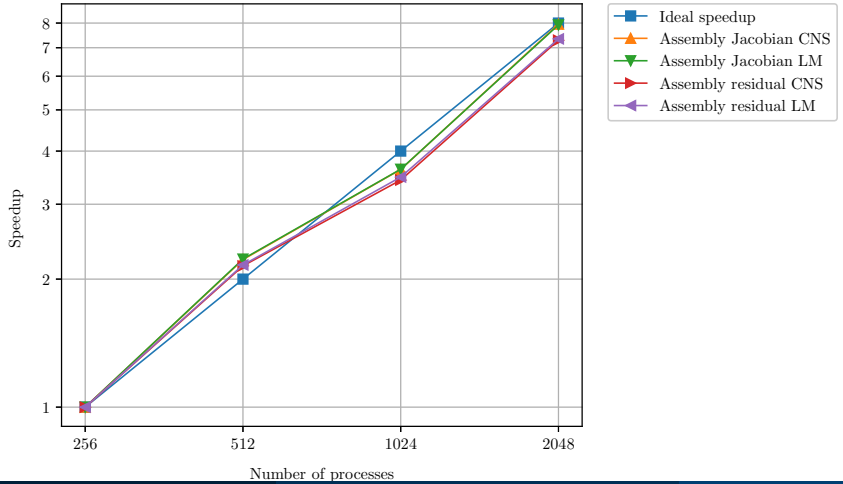
Scalability: Cyclone-Cyclone simulation

- Diameters of tropical cyclones: several 100 km.
- Computational domain: 4,000,000 $\times$ 4,000,000 $\times$ 13,000.
- Stabilized elements.
- 31.6 Mio degrees of freedom (DoFs).
- Numerical models:
  - Compressible Navier-Stokes equations (CNS).
  - Low Mach model (LM).



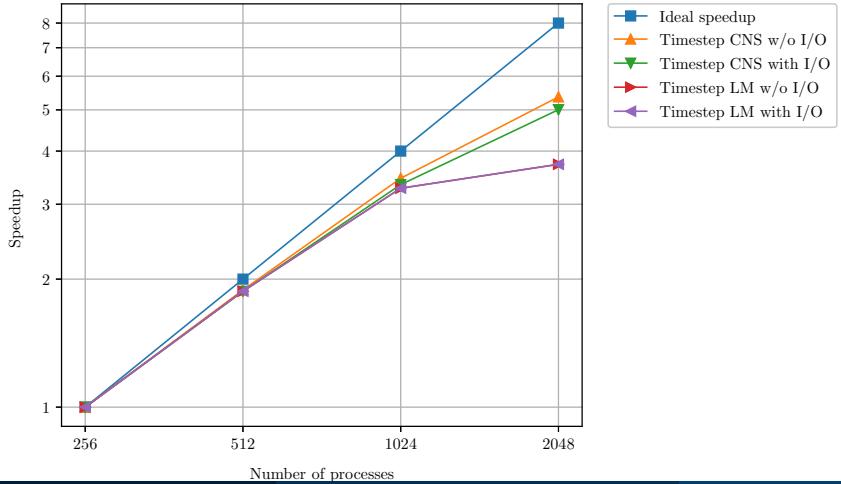
Scalability: Cyclone-Cyclone simulation

Assembly:

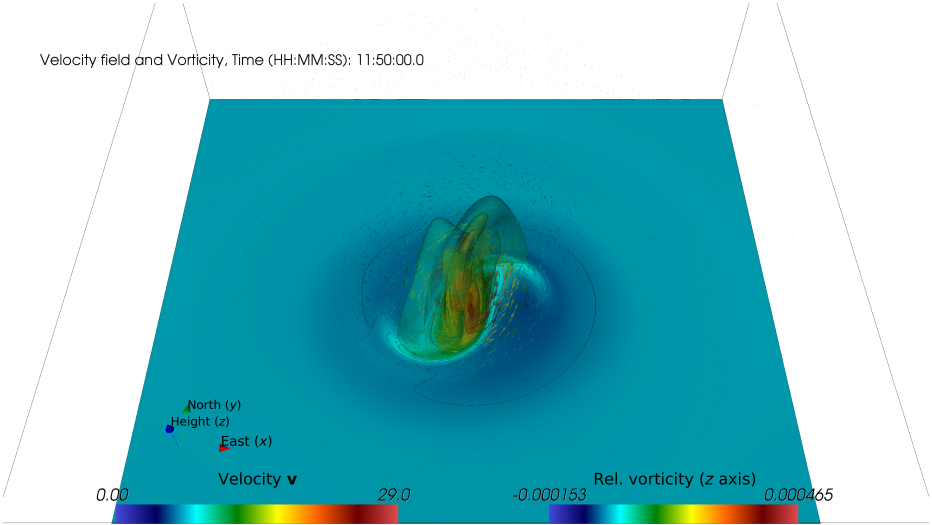


Scalability: Cyclone-Cyclone simulation

Solving time:



Applications: tropical cyclones

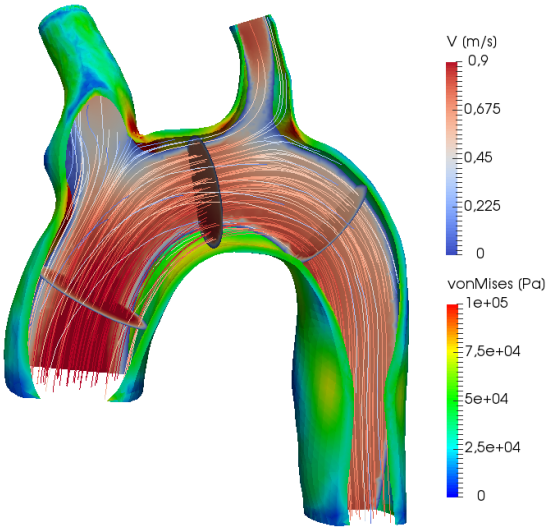


cyclone-cyclone interaction

Applications: tropical cyclones

cyclone-cyclone interaction

Applications: FSI aorta

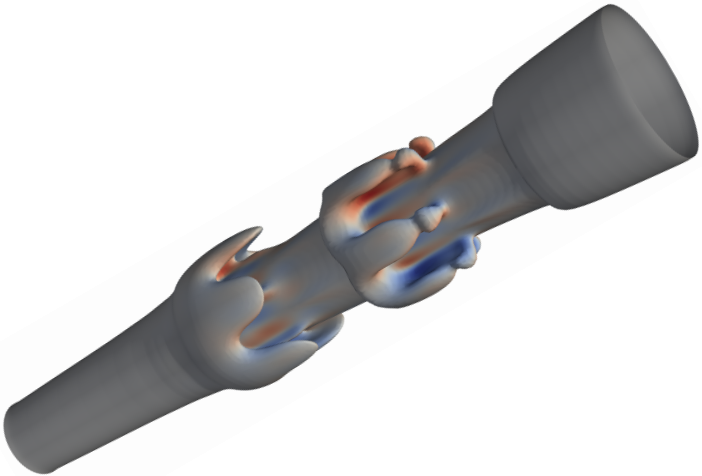


Velocity and Von Mises stress

Applications: FSI aorta

mean value and stress overload probability

Applications: thermal electro-hydrodynamics

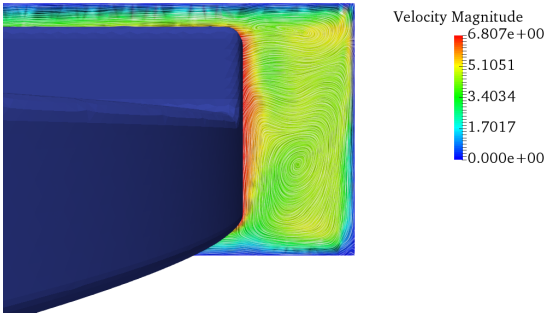


Applications: thermal electro-hydrodynamics

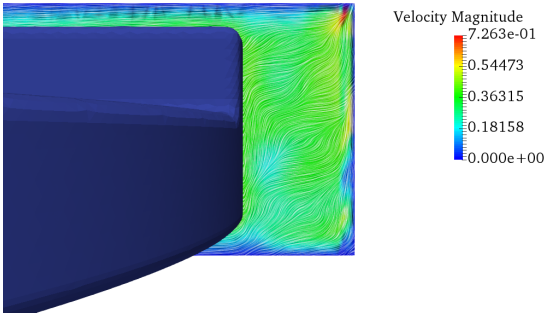
Applications: drug delivery in lung

Applications: drug delivery in lung

Applications: FDA blood pump

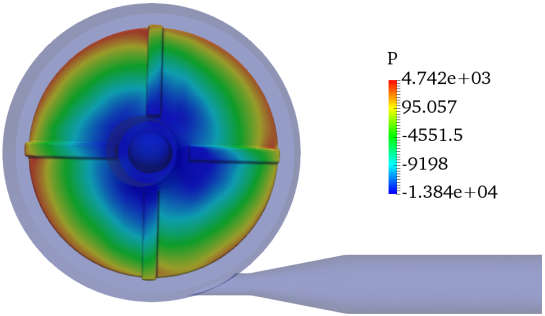


Mean value (m/s).

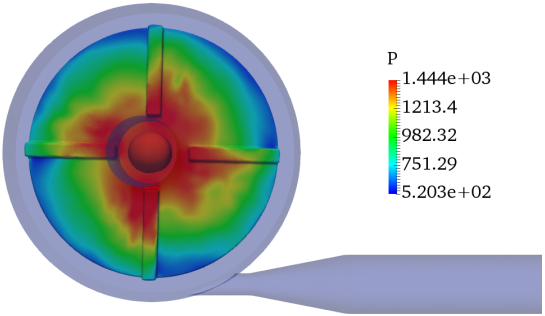


Standard deviation (m/s).

Applications: FDA blood pump



Mean value (Pa).



Standard deviation (Pa).

Applications: FDA blood pump

Mean value (m/s).

Demos

- General compilation.
- Poisson equation.
- UQ - Polynomial Chaos Expansion (PCE).
- Adaptive mesh refinement.
- HDF5 visualization.

Thanks for your attention

http://hiflow3.org/