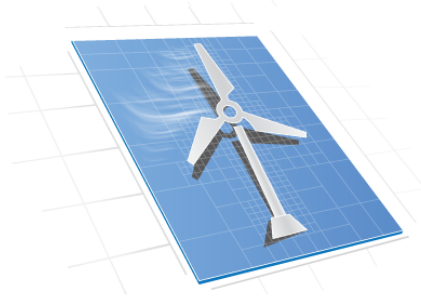


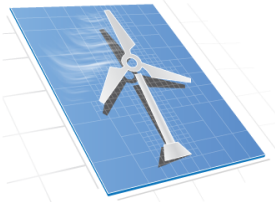
HyperCFD
Fertınaz Yazılım Ltd.

OpenFOAM Workshop
Teknopark Istanbul

05 Numerics
January 2017

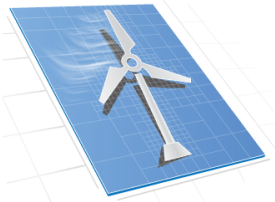
HyperCFD – Disclaimer

This offering is not approved or endorsed by OpenCFD Limited, the producer of the OpenFOAM software and owner of the OpenFOAM and OpenCFD trademarks.



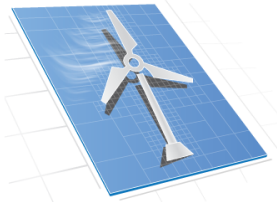
HyperCFD – OpenFOAM Numerical Methods

- OpenFOAM solves Navier-Stokes equations using Finite Volume Method (FVM) like many other commercial CFD packages
- Most general form can be found in literature
- For simplification one can assume density and dynamic viscosity of the fluid are constant
- Then it reduces to a set of 2 equations
  - div $\mathbf{u} = 0$
  - $\rho \left(\mathbf{u}_t + (\mathbf{u} \cdot \nabla) \mathbf{u} \right) = \rho \mathbf{g} - \nabla p + \mu \Delta \mathbf{u}$
- This is a nonlinear PDE classified as mixed parabolic-hyperbolic



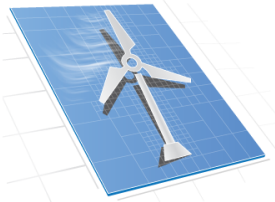
HyperCFD – OpenFOAM Numerical Methods

- FVM very basically integrates the general transport equation in space over arbitrary control volumes
- Since OpenFOAM uses FVM, it discretizes the governing equations and transforms them into a set of algebraic equations
- Then it solves these equations to find the field quantities
- Therefore it requires the following information
  - That is the mesh we generate and store in `costant/polyMesh`
  - Initial and boundary conditions: Defined in `0/` folder
  - Physical quantities, turbulence model: Defined in `constant/`
  - Simulation settings: Defined in `system/`



HyperCFD – OpenFOAM Numerical Methods

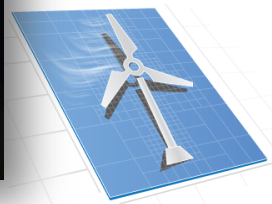
- Discretization of equations – a sample
- Given the 1D PDE
  - $\mathbf{u_t} = \mathbf{u_{xx}}$ with BC $u(0,t) = \alpha(t)$ and $u(1,t) = \beta(t)$ and IC $u(x,0) = u_0(x)$
- Discretize grid: $x_i = ih_x$ where $i = 0, 1, N+1$ and $t_k = kh_t$ where $k = 0, 1, 2, \ldots$
- Discretize equation: Explicit Euler for Time Derivative and Central Difference for Space
- $(u_{i,k+1} - u_{i,k}) / h_t = (u_{i+1,k} - 2u_{i,k} + u_{i-1,k}) / (h_x * h_x)$
- $u_{i,k+1} = \sigma\, u_{i-1,k} + (1-2\sigma)\, u_{i,k} + \sigma\, u_{i+1,k}$ where $\sigma = h_t / (h_x * h_x)$
- Discretize IC and BC: $u_{i,0} = u_0(x_i)$ and $u_{0,k} = \alpha(t_k)$ and $u_{N+1,k} = \beta(t_k)$



HyperCFD – OpenFOAM Numerical Methods

- OpenFOAM applies a similar procedure
- Discretized grid is the mesh generated
- Equation is discretized according to the methods defined in `system/fvSchemes`
- Discretization method for each term is specified in this file
  - `ddtSchemes` is time derivatives
  - `gradSchemes` is gradient term
  - `divSchemes` is the convection term
  - `laplacianSchemes` is the Laplacian terms
  - `snGradSchemes` is the discretization of the surface normal gradients evaluated at the faces

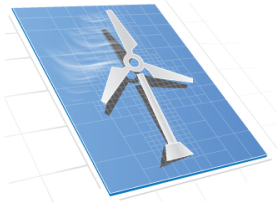
```
ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
    grad(U)         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      Gauss linearUpwind Gauss linear;
    div(phi,k)      Gauss upwind;
    div(phi,epsilon) Gauss upwind;
    div(phi,R)      Gauss upwind;
    div(R)          Gauss linear;
    div(phi,nuTilda) Gauss upwind;
    div((nuEff*dev(grad(U).T()))) Gauss linear;
}

laplacianSchemes
{
    default         none;
    laplacian(nuEff,U) Gauss linear corrected;
    laplacian((1|A(U)),p) Gauss linear corrected;
    laplacian(DkEff,k) Gauss linear corrected;
    laplacian(DepsilonEff,epsilon) Gauss linear corrected;
    laplacian(DREff,R) Gauss linear corrected;
    laplacian(DnuTildaEff,nuTilda) Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
    interpolate(U)  linear;
}

snGradSchemes
{
    default         corrected;
}
```



HyperCFD – OpenFOAM Numerical Methods

- To find the available time discretization schemes:
  - `cd src`
  - `cd finiteVolume/finiteVolume/ddtSchemes`
- First order methods are stable
- Second order methods are more accurate but tend to oscillate
- One can try backward Euler and Crank Nicolson for accuracy but should be careful with possible oscillations

```
fertinaz 4096 Haz 25  2016 backwardDdtScheme
fertinaz 4096 Haz 25  2016 boundedDdtScheme
fertinaz 4096 Haz 25  2016 CoEulerDdtScheme
fertinaz 4096 Haz 25  2016 CrankNicolsonDdtScheme
fertinaz 4096 Haz 25  2016 ddtScheme
fertinaz 4096 Haz 25  2016 EulerDdtScheme
fertinaz 4096 Haz 25  2016 localEulerDdtScheme
fertinaz 4096 Haz 25  2016 SLTSDdtScheme
fertinaz 4096 Haz 25  2016 steadyStateDdtScheme
```

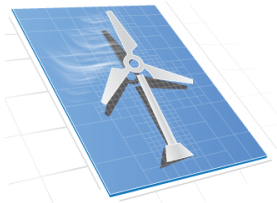


HyperCFD – OpenFOAM Numerical Methods

- Discretization schemes for convective terms
  - `upwind`: First order accurate
  - `linear`: Second order accurate
  - `linearUpwind`: Second order accurate
  - `vanLeer`: Second order accurate
  - `SuperBee`: Second order accurate
  - `Minmod`: Second order accurate

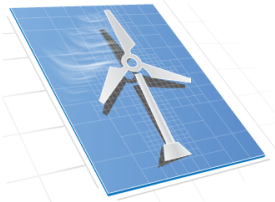
```
--> FOAM FATAL IO ERROR:
Unknown discretisation scheme banana

Valid schemes are :

64
```

- First order methods are bounded and stable but diffusive
- Second order methods are accurate, but they might become oscillatory
- Choosing a second order method is not enough for accuracy. At the end of the day we want stable, bounded and accurate solution.



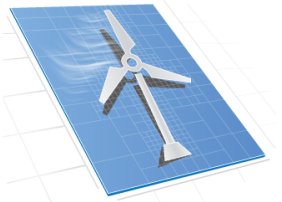
HyperCFD – OpenFOAM Numerical Methods

- Discretization schemes for gradient terms
- All methods for gradient discretization are second order accurate
  - Gauss
  - leastSquares
- There are also limiters to suppress the possible wiggle occurrences
  - Cell-to-cell
  - Face-to-cell
- It increases the stability adds diffusion and reduces accuracy



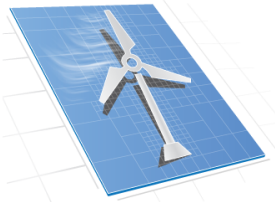
HyperCFD – OpenFOAM Numerical Methods

- Discretization schemes for Laplacian terms
- Mesh related choice
  - orthogonal: Second order accurate and bounded on perfect hex meshes (`blockMesh`)
  - corrected: Recommended on meshes with grading
  - limited: Recommended on meshes with grading
  - uncorrected: Recommended on bad quality meshes



HyperCFD – OpenFOAM Numerical Methods

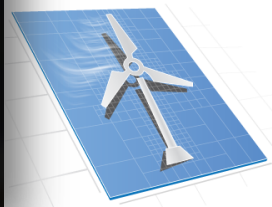
- OpenFOAM linear solvers
- Discretized PDEs are converted into a matrix vector system: $Ax = b$
- A is the coefficient matrix
  - Its form depends on the discretization methods
  - Highly sparse, block diagonal
- A linear solver is an iterative solution technique applied to find the solution of a matrix vector system – in Matlab: $x = A\backslash b$
- Some linear solvers:
  - Gauss-Seidel
  - Conjugate Gradient
  - Multigrid



HyperCFD – OpenFOAM Numerical Methods

- OpenFOAM reads linear solvers from `system/fvSolution`
- There are 4 quantities to be solved corresponding to the 4 equations we have
  - Momentum, continuity, turbulent kinetic energy and dissipation
- Solver for each field is defined with appropriate preconditioners and solution tolerances
  - `Tolerance` checks the residuals: r = Ax – b
  - `relTol` is the ratio between the initial residual and current residual
  - Default value for number of min and max iterations are 0 and 1000, user can assign specific values using `minIter` and `maxIter`
- There is also additional information regarding the flow solver algorithm `SIMPLE`

```
solvers
{
    p
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-06;
        relTol          0.01;
    }
    U
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-05;
        relTol          0.1;
    }
    k
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-05;
        relTol          0.1;
    }
    epsilon
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-05;
        relTol          0.1;
    }
}

SIMPLE
{
    nNonOrthogonalCorrectors 0;

    convergence          1e-4;
}

relaxationFactors
{
    p               0.3;
    U               0.7;
    k               0.7;
    epsilon         0.7;
}
```

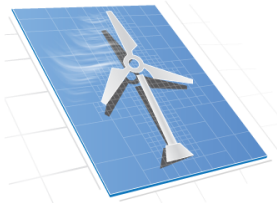


HyperCFD – OpenFOAM Numerical Methods

- OpenFOAM linear solvers – This is the foam-extend version 3.1
  - Symmetric matrix: ($A = A^T$) Example: Pressure
  - Asymmetric matrix: ($A = -A^T$) Example: Velocity

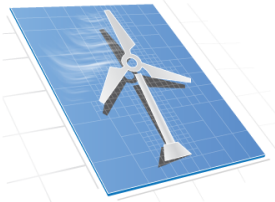
```
--> FOAM FATAL IO ERROR:
Unknown asymmetric matrix solver banana
Valid asymmetric matrix solvers are :

12
(
BICCG
BiCG
BiCGStab
FPEAMG
GAMG
GMRES
MPEAMG
PBiCG
RREAMG
amgSolver
deflation
smoothSolver
)
```

```
--> FOAM FATAL IO ERROR:
Unknown symmetric matrix solver banana

Valid symmetric matrix solvers are :

13
(
BICCG
BiCGStab
CG
FPEAMG
GAMG
GMRES
ICCG
MPEAMG
PCG
RREAMG
amgSolver
deflation
smoothSolver
)
```



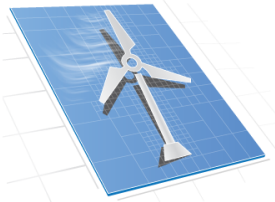
HyperCFD – OpenFOAM Numerical Methods

- OpenFOAM linear solvers
  - `smoothSolver`: Iterative solver using smoother for symmetric and asymmetric matrices
  - `PCG`: Preconditioned conjugate gradient solver for symmetric lduMatrices using a run-time selectable preconditiioner
  - `PBiCG`: Preconditioned bi-conjugate gradient solver for asymmetric lduMatrices using a run-time selectable preconditioner
  - `GAMG`: Geometric agglomerated algebraic multigrid solver



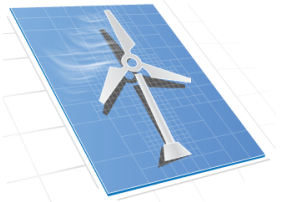
HyperCFD – OpenFOAM Numerical Methods

- OpenFOAM preconditioners
  - Preconditioning is a transformation method that produces a matrix vector system which is faster to solve than the original one
  - `DILU`: Simplified diagonal-based incomplete LU preconditioner for asymmetric matrices.
  - `DIC`: Simplified diagonal-based incomplete Cholesky preconditioner for symmetric matrices (symmetric equivalent of DILU)



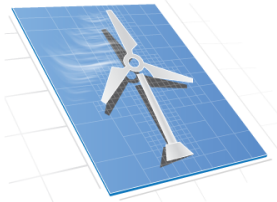
HyperCFD – OpenFOAM Numerical Methods

- OpenFOAM smoothers: Reduce the mesh dependency of the number of iterations required by the linear solver
- List of smoothers:
  - `GaussSeidel`
  - `DILU`: Simplified diagonal-based incomplete LU smoother for asymmetric matrices
  - `DILUGaussSeidel`: Combined DILU/GaussSeidel smoother for asymmetric matrices in which DILU smoothing is followed by GaussSeidel to ensure that any "spikes" created by the DILU sweeps are smoothed-out
  - `nonBlockingGaussSeidel` : Variant of `GaussSeidel` that expects processor boundary cells to be sorted last and so can block later. Only when the cells are actually visited does it need the results to be present. It is expected that there is little benefit to be gained from doing this on a patch by patch basis since the number of processor interfaces is quite small and the overhead of checking whether a processor interface is finished might be quite high (call into mpi). Also this would require a dynamic memory allocation to store the state of the outstanding requests.



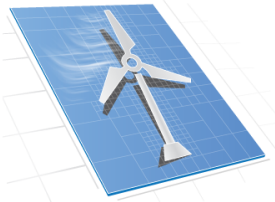
HyperCFD – OpenFOAM Numerical Methods

- There is no generally accepted the best linear solver / preconditioner / smoother combination
  - `smoothSolver` with `GaussSeidel` seems to work fine with asymmetric matrices
  - `GAMG` is usually the first option to check for symmetric matrices
  - If `GAMG` fails then `PCG` would worth to try
- Follow residuals to understand how the linear solvers behave



HyperCFD – OpenFOAM Numerical Methods

- OpenFOAM: How the NS equations are solved?
- OpenFOAM uses pressure based methods:
  - Velocity field is obtained from the momentum equations
  - The pressure field is extracted by solving pressure correction equation which is obtained by manipulating continuity and momentum equations
  - Additional equations for other scalars such as turbulence and volume fraction are then solved
  - The solution process keeps iterating over the entire set of governing equations until the solution converges to a given criterion or the user decides to stop the simulation



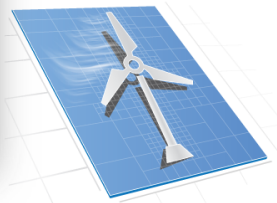
HyperCFD – OpenFOAM Numerical Methods

- OpenFOAM uses pressure based methods:
  - `SIMPLE:` Steady state simulations
  - `SIMPLEC:` Steady state simulations
  - `PISO:` Transient simulations
  - `PIMPLE:` Hybrid model
- Cannot discuss details of each method but all are pressure-based, predictor-corrector algorithms and continue to iterate over solutions until the convergence is achieved
- Specifications should be made inside the `system/fvSolution`

```
SIMPLE
{
    nNonOrthogonalCorrectors 0;

    convergence           1e-4;
}
```

```
PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 0;
    pRefCell        0;
    pRefValue       0;
}
```

```
PIMPLE
{
    nOuterCcorrectors 2;
    nCorrectors     2;
    nNonOrthogonalCorrectors 0;
    pRefCell        0;
    pRefValue       0;
}
```



HyperCFD – OpenFOAM Numerical Methods References

- Some recommended references on numerical methods:
  - *Finite Volume Methods For Hyperbolic Problems*, LeVeque
  - *Error Analysis and Estimation for the FVM with Applications to Fluid Flow*, Jasak
  - *Fluid Mechanics*, Kundu