

Summer School on Fire Dynamics Modeling 2017

Dr. Susanne Kilian
hhpberlin - Ingenieure für Brandschutz
10245 Berlin - Germany

August 8, 2017

The governing equations

- Basic conservation equations

- Navier-Stokes equations

The low Mach number assumption

- Flow regimes in real-life applications

- Pressure decomposition for low-Mach flow

The solution of the Pressure-Poisson equation

- Derivation of Pressure-Poisson Equation

- Discretization in space and time

Parallelization concepts in FDS

- Different parallelization concepts

- Multi-Mesh applications in FDS

The governing equations

Basic conservation equations

Navier-Stokes equations

Mathematical-physical model

"FDS solves numerically a form of the Navier-Stokes equations appropriate for low-speed ($Ma < 0.3$), thermally-driven flow with an emphasis on smoke and heat transport from fires"

Extract from FDS User's Guide

Navier-Stokes Equations

- non-linear, coupled system of partial differential equations (PDE)
- prediction of spacial and temporal evolution of fluid flows (liquid or gas)
- based on conservation laws for mass, momentum and energy
- applicable to full range of low up to high speed flows

Conservation Laws

Physical principles

- Mass is conserved
- Newton's second law
- First law of thermodynamics



Mathematical equations

- Continuity equation
- Momentum equation
- Energy equation

The only way to change the amount of a quantity ϕ in a control volume with time is to flux it through the boundary or create/consume it within the volume

Based on continuum assumption:

- molecular fluid structure ignored
- only described by macroscopic properties (density, pressure, temperature, velocity, ...)

General form of conservation equations

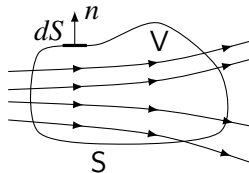
Integral form

temporal change on volume

flux through surface

effect of sources/sink

$$\int_V \frac{\partial}{\partial t} \phi \, dV + \int_S H_\phi \cdot \mathbf{n} \, dS = \int_V F_\phi \, dV$$



General form of conservation equations

Integral form

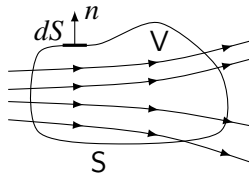
temporal change on volume

flux through surface

effect of sources/sink

$$\int_V \frac{\partial}{\partial t} \phi \, dV + \underbrace{\int_S H_\phi \cdot \mathbf{n} \, dS}_{\int_V \nabla \cdot H_\phi \, dV} = \int_V F_\phi \, dV$$

Divergence theorem



General form of conservation equations

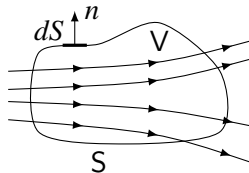
Integral form

temporal change on volume

flux through surface

effect of sources/sink

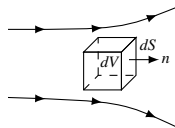
$$\int_V \frac{\partial}{\partial t} \phi \, dV + \underbrace{\int_S H_\phi \cdot \mathbf{n} \, dS}_{\int_V \nabla \cdot H_\phi \, dV \quad \text{Divergence theorem}} = \int_V F_\phi \, dV$$



infinitesimally small volume element

Differential form

$$\frac{\partial \phi}{\partial t} + \nabla \cdot H_\phi = F_\phi$$



Governing Equations in conservative form

Continuity equation: Mass is neither created nor destroyed

$$\phi = \rho \quad H_\phi = \rho \mathbf{u}, \quad F_\phi \equiv F_c(\dots) \quad \swarrow$$

mass production species

Momentum equation: Change of momentum = surface forces + volume forces

$$\phi = \rho \mathbf{u} \quad H_\phi = \rho \mathbf{u} \mathbf{u}, \quad F_\phi \equiv F_m(\dots) \quad \swarrow$$

viscosity, gravity, particle drag

Energy equation: Change of internal energy = heat transfer + work done

$$\phi = \rho e, \quad H_\phi = [\rho e + p] \mathbf{u}, \quad F_\phi \equiv F_e(\dots) \quad \swarrow$$

fire, conduction, radiation

The low Mach number assumption

Flow regimes in real-life applications

Pressure decomposition for low-Mach flow

Flow regimes in real-life applications

Typical situation

- flow field is characterized by multiple space and/or time scales
- principal flow velocities are small compared to speed of sound

————→ Standard numerical techniques are expensive or may fail

Key idea

- identify singular limit regime in which numerical methods fail
- analyze mathematical structure of singular limit and reasons for failure
- develop modifications/simplifications for numerical schemes to overcome difficulties

Spezifikation of flow regime

Mach number

$$M := \frac{u_{ref}}{c_{ref}} = \frac{\text{speed of flow}}{\text{speed of sound}}$$

Measure for compressibility of flow

$M = 0$: incompressible

$0 < M \ll 1$: weakly compressible

$M = 1$: fully compressible

→ Big influence of M on mathematical equations and their numerical solution

Note: Speed of sound corresponds to velocity by which small perturbations are propagated

Mach number for fire driven flows

Typical situation

- comprehensible part of flow is in low-Mach range
- sound waves travel much faster

$$M_{fire} \lesssim 0.03$$

very small !

Speed for typical fire applications:

$$u_{ref} \lesssim 10 \frac{m}{s}$$

Speed of sound in dry air:

$$c_{ref} \approx 343 \frac{m}{s}$$

Strong restrictions to full schemes

Note: General Navier-Stokes equations take into account all scales of velocity

CFL-condition is related to speed of sound

- extremely small time steps $\Delta t \sim \Delta x / c_{ref}$
- extremely many time steps to move just one cell $\sim 1/M$
- large influence of numerical dissipation

→ Troubles with efficiency and accuracy

Low Mach number asymptotics

Key observation for $M \rightarrow 0$

- compression effects become negligible
- sound-wave propagation becomes unnoticeable
- flow mainly driven by slower convection terms

————→ Flows of compressible fluids may be considered incompressible for $M < 0.3$

Idea: Decouple sound waves from equations

- focus on low-Mach flow only
- allow time steps to be bounded only by the speed of flow
- reduce numerical effort comprehensibly

Governing equations in conservative form

Continuity equation

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = F_c$$

Momentum equation

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \mathbf{u}) + \nabla p = F_m$$

Energy equation

$$\frac{\partial \rho e}{\partial t} + \nabla \cdot ([\rho e + p] \mathbf{u}) = F_e$$

Note: Species equation not listed here because not needed for further discussions

Governing equations in conservative form

Continuity equation

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = F_c$$

Momentum equation

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \mathbf{u}) + \nabla p = F_m$$

only pressure gradient appears
no absolute value needed

Energy equation

$$\frac{\partial \rho e}{\partial t} + \nabla \cdot ([\rho e + p] \mathbf{u}) = F_e$$

velocity components strongly
coupled with other equations

→ no separate evolution equation for the pressure (6 quantities, 5 equations)
additional equation of state $p = \rho RT$ to close system (ideal gas law)

Pressure decomposition for low-Mach flow

Basic strategy (according to Rehm and Baum)

- pressure is decomposed into two parts with different physical meaning

$$p(\mathbf{x}, t) = \bar{p}_m(z, t) + \tilde{p}(\mathbf{x}, t)$$

- the single parts account for the effects on different scales

Background pressure $\bar{p}(z, t)$

thermodynamic

resolves the large-scaled fluctuations

Perturbation pressure $\tilde{p}(\mathbf{x}, t)$


hydrodynamic

resolves the small-scaled fluctuations

Momentum equation

Fully resolved case

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \mathbf{u}) = -\nabla p - \nabla \cdot \boldsymbol{\tau} + (\rho - \rho_0) \mathbf{g}$$

 behaves like $O(\frac{1}{M^2})$

Observations for $M \rightarrow 0$

- pressure gradient contribution gets singular (tends towards infinity)
- introduces a significant source of inaccuracy

Momentum equation

Low-Mach limit

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \mathbf{u}) = -\nabla \tilde{p} - \nabla \cdot \boldsymbol{\tau} + (\rho - \rho_0) \mathbf{g}$$

only perturbation pressure retained

Simplifications

- only perturbation pressure has to be considered in momentum equation
- CFL-condition only depends on flow velocity (larger time steps!)
- perturbation pressure must be recomputed in every time step

Equation of state and energy equation

Fully resolved case

$$\rho e = \frac{p}{\gamma - 1} + \frac{\rho \mathbf{u} \mathbf{u}}{2} \quad \text{and} \quad p = \rho T R$$

kinetic part behaves like $O(M^2)$

Observations for $M \rightarrow 0$

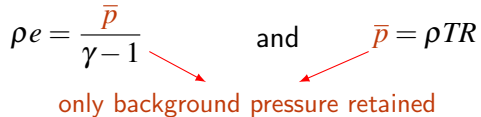
- temperature and density can be assumed to be inversely proportional
- kinetic part in energy density decreases with $O(M^2)$
- internal energy e and enthalpy h related via background pressure $h = e + \bar{p}/\rho$

Equation of state and energy equation

Low-Mach limit

$$\rho e = \frac{\bar{p}}{\gamma - 1} \quad \text{and} \quad \bar{p} = \rho TR$$

only background pressure retained



Simplifications

- energy conservation can be written in terms of sensible enthalpy h_s
- energy equation does not need to be solved explicitly
- no influence of perturbation pressure any more

Divergence condition

New energy conservation equation

$$\frac{\partial \rho h_s}{\partial t} + \nabla \cdot (\rho h_s \mathbf{u}) = \frac{D\bar{p}}{Dt} + \dot{q}''' - \nabla \cdot \dot{\mathbf{q}}''$$

Factoring out the divergence

$$\nabla \cdot \mathbf{u} = D - P \frac{\partial \bar{p}}{\partial t} := \frac{1}{\rho h_s} \left[\frac{D}{Dt} (\bar{p} - \rho h_s) + \dot{q}''' - \nabla \cdot \dot{\mathbf{q}}'' \right]$$

divergence directly related to background pressure and heat

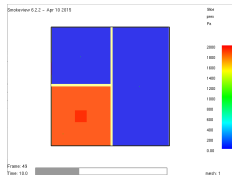
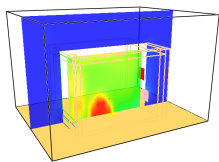
→ Energy equation only serves as divergence condition for velocity field

Note: Due to low-Mach assumption, the material derivative simplifies to $\frac{Dp}{Dt} = \frac{\partial p}{\partial t} + \mathbf{u} \cdot \nabla p \approx \frac{\partial \bar{p}}{\partial t} + w \frac{\partial \bar{p}}{\partial z}$

Handling of different pressure zones

Individual treatment of closed compartments

- \bar{p} is usually constant for a building except for stratification
- in specified compartments different background pressures \bar{p}_m can be used
e.g. if pressure is increased by fire or air is blown in a separated zone



- zones can be connected via HVAC-system, but no detailed flow equations must be solved within ducts
- if separation between zones breaks, consistency is guaranteed by time integration over zone volumes

Summary: The role of the different pressure variables

| Background pressure thermodynamic | Perturbation pressure hydrodynamic |
|--|--|
| function of time t and only z -coordinate | function of time t and space |
| resolves stratification of atmosphere | drives the fluid motion |
| only retains in the equations of energy | only retains in the momentum equation and state |
| spatially homogenous, varies only in big length scale, infinitely fast in small scales | represents local changes in the small-scaled flow structure |
| delivers divergence constraint for velocity field | guarantees fulfillment of the divergence constraint in momentum equation |
| not influenced by local pressure changes | no influence on energy density |

Simplified low-Mach equations

Limit of Navier-Stokes equations for $M \rightarrow 0$

- new set of equations which focus on advective transport of smoke, heat and momentum
- momentum and energy equations are decoupled
- energy equation isn't considered as separate conservation equation but its terms are used for definition of a divergence constraint for the velocity field
- divergence constraint becomes an elliptic equation for the pressure which is enforced by momentum equation
- when divergence constraint is fulfilled by velocity field, sensible enthalpy equation is satisfied by construction

The solution of the Pressure-Poisson equation

Derivation of Pressure-Poisson equation

Discretization in space and time

Derivation of Pressure-Poisson Equation

Non-conservative formulation of momentum equation

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} \right) + \nabla p = \rho \mathbf{g} + \mathbf{f}_b + \nabla \cdot \boldsymbol{\tau}_{ij}$$

Derivation of Pressure-Poisson Equation

Non-conservative formulation of momentum equation

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \underbrace{(\mathbf{u} \cdot \nabla) \mathbf{u}}_{\substack{\frac{\nabla |\mathbf{u}|^2}{2} - \mathbf{u} \times \boldsymbol{\omega} \\ \text{vector identity}}} \right) + \underbrace{\nabla p}_{\substack{\rho_0 \mathbf{g} + \nabla \tilde{p} \\ \text{pressure decomposition}}} = \rho \mathbf{g} + \mathbf{f}_b + \nabla \cdot \boldsymbol{\tau}_{ij}$$

Note: Hydrostatic pressure gradient $\rho_0 \mathbf{g}$ was subtracted from both sides

Derivation of Pressure-Poisson Equation

Replace terms and divide by density

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\nabla |\mathbf{u}|^2}{2} - \underbrace{\mathbf{u} \times \boldsymbol{\omega}}_{\downarrow} + \frac{1}{\rho} \nabla \tilde{p} = \frac{1}{\rho} \left[(\rho - \rho_0) \mathbf{g} + \mathbf{f}_b + \nabla \cdot \boldsymbol{\tau}_{ij} \right]$$
$$\nabla \left(\frac{\tilde{p}}{\rho} \right) - \tilde{p} \nabla \left(\frac{1}{\rho} \right)$$

Derivation of Pressure-Poisson Equation

Define stagnation energy: $H \equiv |\mathbf{u}|^2/2 + \tilde{p}/\rho$

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla H - \underbrace{\mathbf{u} \times \boldsymbol{\omega} - \frac{1}{\rho} [(\rho - \rho_n) \mathbf{g} + \mathbf{f}_b + \nabla \cdot \boldsymbol{\tau}_{ij}]}_{\mathbf{F}_A} - \underbrace{\tilde{p} \nabla \left(\frac{1}{\rho} \right)}_{\mathbf{F}_B} = 0$$

accounts for advective terms accounts for baroclinic torque

Derivation of Pressure-Poisson Equation

Simplified momentum equation

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla H + \mathbf{F} = 0 \quad \text{with} \quad \mathbf{F} = \mathbf{F}_A + \mathbf{F}_B$$

$$\downarrow$$
$$\nabla H = -\frac{\partial \mathbf{u}}{\partial t} - \mathbf{F}$$

Derivation of Pressure-Poisson Equation

Taking its divergence

$$\nabla^2 H = -\frac{\partial(\nabla \cdot \mathbf{u})}{\partial t} - \nabla \cdot \mathbf{F}$$

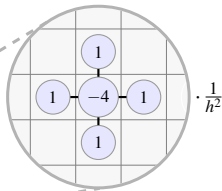
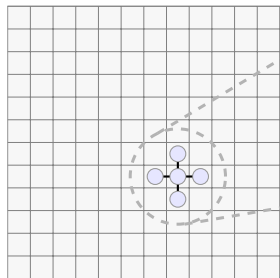
→ Elliptic partial differential equation for pressure term H

Advantages:

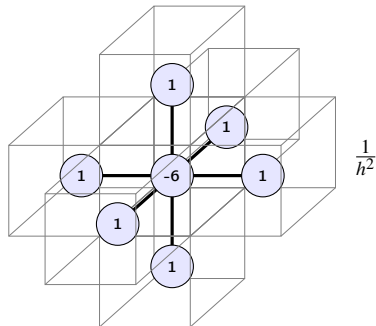
- separable linear algebraic system with constant coefficients
- solvable up to machine accuracy by optimized direct Fast Fourier method (FFT) (from elliptic solver package CRAYFISHPAK)

Space discretization by finite difference method

$$\begin{aligned} & \frac{H_{i+1,jk} - 2H_{ijk} + H_{i-1,jk}}{\delta x^2} + \frac{H_{i,j+1,k} - 2H_{ijk} + H_{i,j-1,k}}{\delta y^2} + \frac{H_{ij,k+1} - 2H_{ijk} + H_{ij,k-1}}{\delta z^2} \\ &= -\frac{F_{x,ijk} - F_{x,i-1,jk}}{\delta x} - \frac{F_{y,ijk} - F_{y,i,j-1,k}}{\delta y} - \frac{F_{z,ijk} - F_{z,ij,k-1}}{\delta z} - \frac{\delta}{\delta t}(\nabla \cdot \mathbf{u})_{ijk} \end{aligned}$$



second order accuracy



Time discretization by predictor-corrector method

Explicit time stepping methods

- easy to program
- simple backward substitution
- less computing time per time step
- less storage needs
- strong stability constraints (small time steps)

Implicit time stepping methods

- more complicated to program
- solution of system of equations
- high computing time per time step
- higher storage needs
- better stability (much bigger time steps)

Predictor-corrector methods

Combination of both to save the advantages

Pressure solutions per time step

2-stage Runge-Kutta method

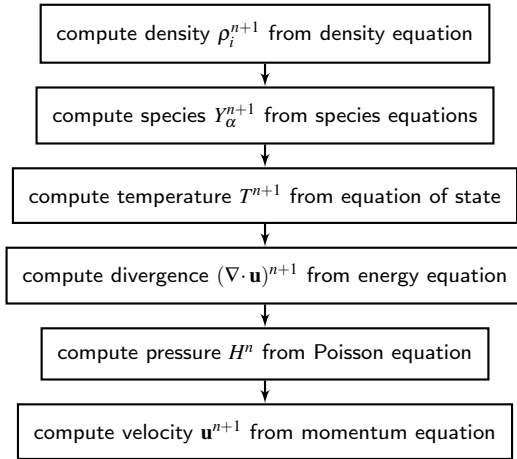
Predictor step

$$\nabla^2 H^n = -\frac{(\nabla \cdot \mathbf{u})^* - \nabla \cdot \mathbf{u}^n}{\delta t} - \nabla \cdot \mathbf{F}^n$$

Corrector step

$$\nabla^2 H^* = -\left[\frac{(\nabla \cdot \mathbf{u})^{n+1} - \frac{1}{2}(\nabla \cdot \mathbf{u}^* + \nabla \cdot \mathbf{u}^n)}{\delta t/2} \right] - \nabla \cdot \mathbf{F}^*$$

Basic Algorithm in space and time



One stage: Forward Euler

If p^n satisfies discrete Poisson equation, then \mathbf{u}^{n+1} is divergence-free at next time step

effective coupling of mass,
momentum, energy,
and equation of state

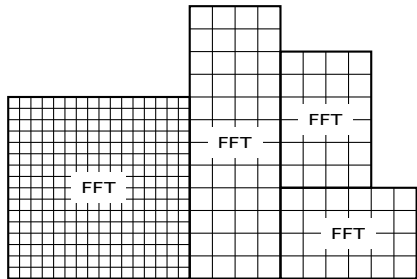
Boundary Conditions

| Boundary | Description | Type |
|-----------|---|-----------|
| open | <ul style="list-style-type: none">external boundary, free in/outflow | Dirichlet |
| solid | <ul style="list-style-type: none">external boundary, entirely solid or forced | Neumann |
| | <ul style="list-style-type: none">external boundary, mix of solid and open | Dirichlet |
| | <ul style="list-style-type: none">internal obstruction | IBM |
| interface | <ul style="list-style-type: none">common face of neighboring meshes | Dirichlet |

Note: Dirichlet (set function value), Neumann (set normal derivativ), IBM (Immersed boundary method)

Decomposition into multiple meshes

Local FFT-methods with interface update



- **Fast Fourier Transformation:**
spectral solver with nearly optimal complexity $O(N \log(N))$
- exploits the fact that sine and cosine are eigenfunctions of Laplace operator
- only works for purely rectilinear domains
- locally correct solutions to the related local Poisson problems
- mesh solutions are clustered together to global one (averaging at interfaces)

Possible troubles with the pressure solution

Along mesh interfaces

- H is continuous along interfaces, but its discretized gradient is not
- normal velocity components at the interface may differ at t_{n+1}

Along internal solids

- normal component of \mathbf{F} is set equal to previous value of gradient of H
- normal component of velocity is not exactly zero

Different values of \tilde{p} in equation

- perturbation pressure \tilde{p} included in \mathbf{F} is from previous time step
- value of \tilde{p} implicit in H will not equal value in \mathbf{F} after solving Poisson equation

Iterative procedure for updating velocity

Specify tolerance for velocity components

```
&PRES VELOCITY_TOLERANCE = 0.001, MAX_PRESSURE_ITERATIONS = 100
```

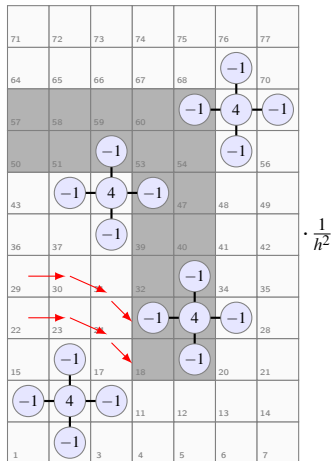
Multiple solutions of the Poisson equation per time step

- until normal component of velocity at internal solids and mesh boundaries converges within tolerance
- until old and new values of the perturbation pressure \tilde{p} converge to within tolerance

→ Additional overhead depending on geometry and mesh decomposition

Handling of internal obstructions

Structured cartesian grids

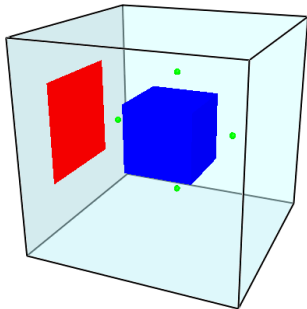


- Internal obstructions are represented as masked grid cells
- no-flux condition cannot be directly prescribed
- possible penetration of velocity field into internal solids
- iterative correction of normal velocity components along internal solids by **immersed boundary method** up specified to tolerance

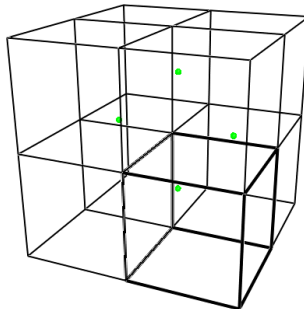
Test case: 3D-cube with obstruction

Using additional multi-mesh decomposition

Different pressure devices



8-mesh decomposition



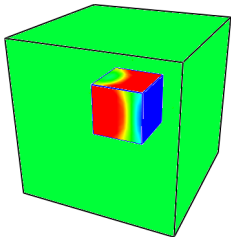
Note: Cyclic inflow from the left (via ramp), open outflow on the right

Test case: 3D-cube with obstruction

Single-mesh case: Velocity errors for different tolerances, 24^3 cells

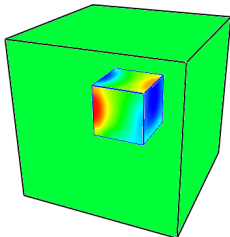
$$tol = 10^{-2}$$

Ø 1 pressure iteration



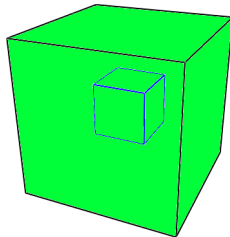
$$tol = 10^{-6}$$

Ø 3.5 pressure iterations



$$tol = 10^{-16}$$

Ø 30 pressure iterations

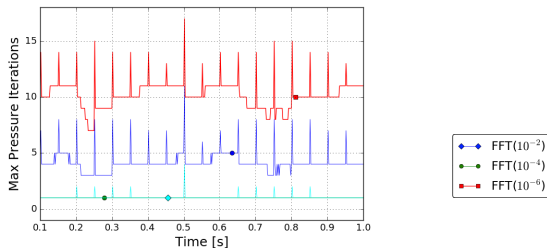


Note: Pressure iteration successfully reduces velocity error along internal obstruction

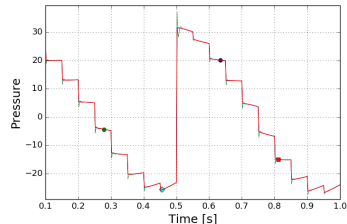
Test case: 3D-cube with obstruction

1-mesh case: pressure iterations and course of devices

Number of pressure iterations per time step



Pressure device on the right

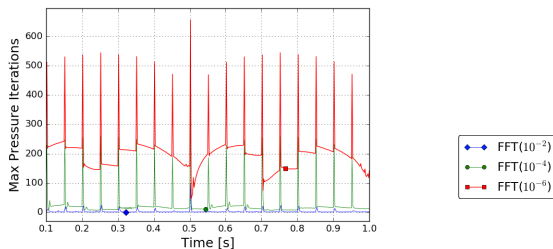


→ Requires more pressure iterations if tolerance is decreased (≈ 4 for $\text{tol}=10^{-4}$, ≈ 10 for $\text{tol}=10^{-6}$)

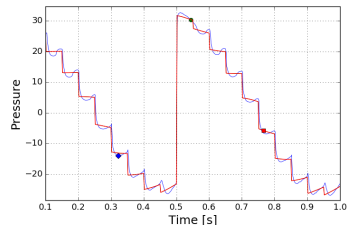
Test case: 3D-cube with obstruction

8-mesh case: pressure iterations and course of devices

Number of pressure iterations per time step



Pressure device on the right

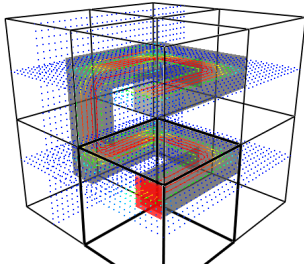


→ Handling of mesh interfaces requires still much more pressure (≈ 25 for $\text{tol}=10^{-4}$, ≈ 200 for $\text{tol}=10^{-6}$)

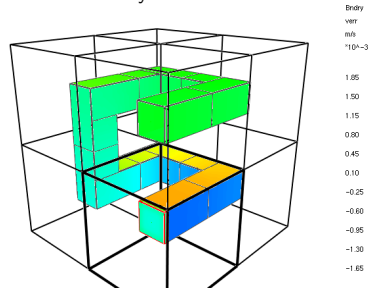
Test case: Flow through a pipe

Velocity error along internal obstructions

Velocity field



Velocity error for $tol = 10^{-4}$

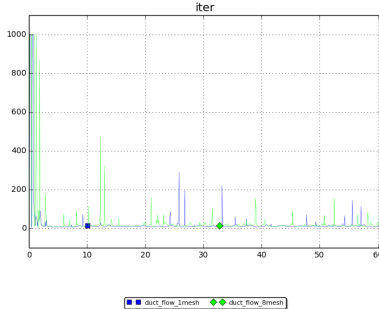


Note: Hard test case due to many changes of direction and multiple obstructions

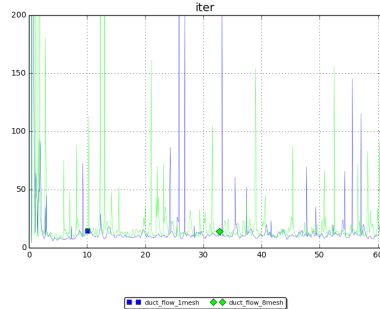
Test case: Flow through a pipe

Comparison of pressure iterations

1-mesh versus 8-mesh



1-mesh versus 8-mesh - zoomed



➔ Number of pressure iterations increases comprehensively

Different test cases to run

- **zone_break**: Definition of pressure zones which open at specified times
- **door_crack**: Fire in sealed compartment with leakages to outside
- **stack_effect**: Modeling stack effects in a building
- **duct_flow**: Flow through a angled 3D-pipe
- **dancing_eddies**: Flow through a 2D-channel with obstruction
- **cube3d_obstruction**: Flow through a 3D-cube with obstruction

Parallelization concepts in FDS

Different parallelization concepts

Multi-mesh applications in FDS

Parallelization

Why parallelization?

- highly complex geometries with huge spatial extents (millions of unknowns)
- many interacting physical and chemical quantities over very long simulation times
- storage restrictions on single processor systems

→ extreme needs of memory and computing power

Main objectives

- enlarging the range of computable problems
- improving resulting accuracy by use of higher resolutions

Parallelization of algorithms

Idea behind

- reduce computational time by use of additional resources (CPU's, cores)
- in ideal case parallel execution time should be inversely proportional to number of processors

————→ Realistic goal?

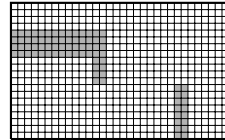
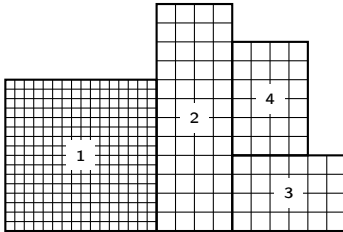
Basic programming concepts

- Message Passing Interface (MPI) → Distributed memory
- Open Multi-Processing (OpenMP) → Shared memory

Grid decomposition in FDS

Multi-mesh applications

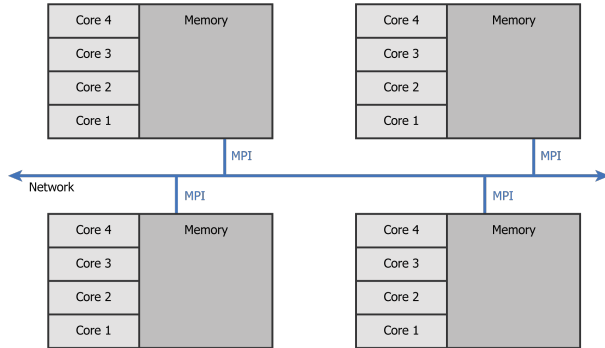
- computational domain is subdivided into multiple meshes
- workload can be distributed among multiple CPUs
- each mesh consists of a rectilinear uniformly spaced grid
- different resolutions are possible on single meshes (must match by integer ratio)
- internal solids are represented as rectangular obstructions snapping to grid



→ if possible keep mesh interfaces free of complicated phenomena

Distributed Memory

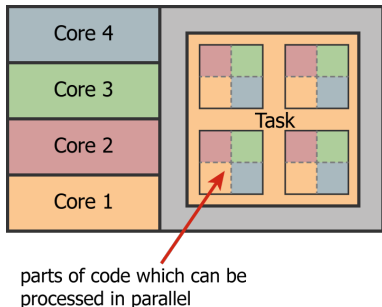
Message Passing Interface (MPI)



- based on a network connected CPU cluster
- given problem is subdivided into subprocesses which are assigned to one or more CPUs
- subprocesses act independently with own private memory
- data must be transferred explicitly via MPI-library (communication routines)

Shared Memory

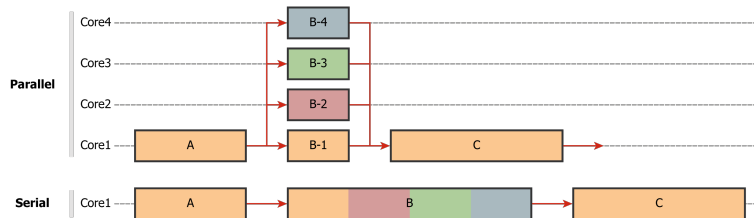
Open Multi-Processing (OpenMP)



- multiple cores on single CPU are exploited by **threads**
- all threads have direct access to common physical memory
- parallelization is handled via directive-based programming constructs
- number of available cores must be specified by **OMP_NUM_THREADS**

OpenMP: Handling single-mesh cases

Single mesh per CPU - only one core used

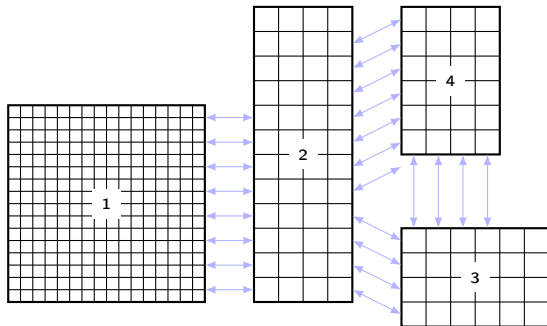


```
$OMP PARALLEL DO
do i=1,n
    z(i) = a*x(i) + b*y(i)
enddo
$OMP END PARALLEL DO
```

- creates a set of threads, executes a block of code in parallel, then joins the threads
- default number of available cores set via `OMP_NUM_THREADS`
- number of used cores is printed at job start in FDS

Grid decomposition in FDS

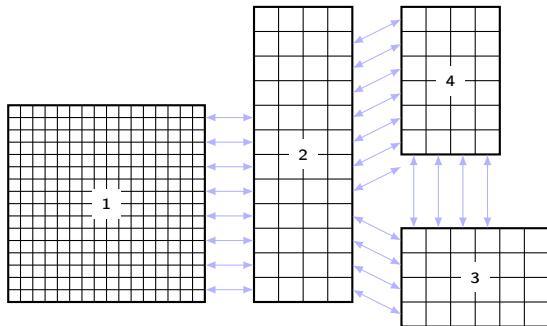
Usage of MPI in FDS



- each mesh corresponds to a separate **processes** which is assigned to a CPU

Grid decomposition in FDS

Usage of MPI in FDS



- each mesh corresponds to a separate **processes** which is assigned to a CPU
- in every time step different data along mesh interfaces have to be transferred over network

MPI versus OpenMP

| MPI | OpenMP |
|---|--|
| <ul style="list-style-type: none">+ potentially good scalability properties (memory scales with number of processors)- programmer responsible for communication and synchronization between processes- additional overhead by data exchange (network transfer, synchronization)- mapping of existing global data structures to distributed organization difficult- optimized serial codes can't be used without special adaptations | <ul style="list-style-type: none">- potential scalability restricted by architecture of CPU (number of cores)- programmer responsible for identification of parallelism and thread synchronization+ no communication needed between threads+ no reorganisation of memory organisation needed- speedup less than number of cores (by far) |

Hybrid usage of shared and distributed memory

Mixture of OpenMP and MPI

- multiple MPI processes communicating via MPI-library
- each MPI process itself has several OpenMP threads

→ Hybrid versions preferred on modern HPC-computers

Careful balancing of resources necessary

- cores of CPU can be used for both, OpenMP and MPI
- optimal distribution of programming tasks necessary to get maximum speedup

Command line syntax

Mac OSX and Linux

- MPI installation, e.g. by www.open-mpi.org (in addition to FDS installation)
- for different computers my_hosts.txt-File with specification of slots

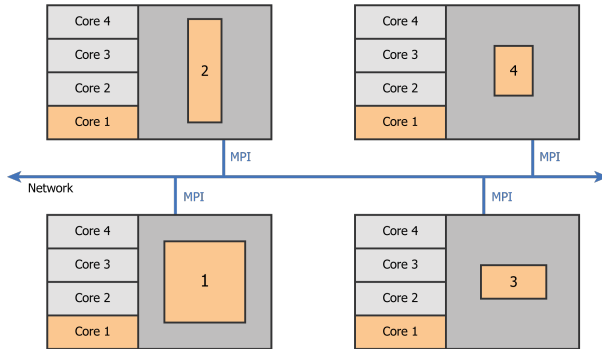
```
mpirun -np <NPROCESSES> fds -hostfile myhosts.txt chid.fds
```

Windows

- MPI-routines bundled with the FDS download (only FDS installation needed)
- system environment variables set automatically, default with 4 OpenMP threads

```
mpiexec -n <NPROCESSES> fds chid.fds
```

Processing multi-mesh cases in FDS



- each mesh assigned to single CPU
- data transfer between CPU's handled via MPI over network

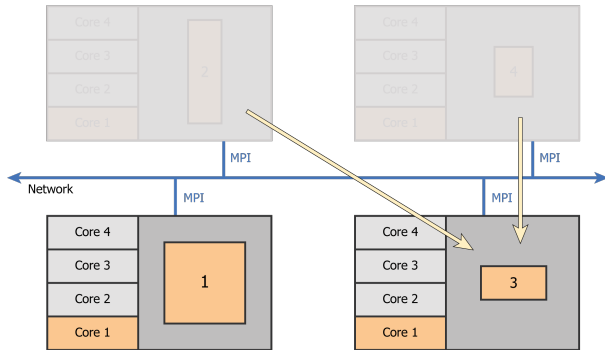
Use of 4 processes for 4 meshes:

```
&MESH ID='mesh1', IJK=..., XB=... /  
&MESH ID='mesh2', IJK=..., XB=... /  
&MESH ID='mesh3', IJK=..., XB=... /  
&MESH ID='mesh4', IJK=..., XB=... /
```

```
mpirun -np 4 fds.exe chid.fds
```

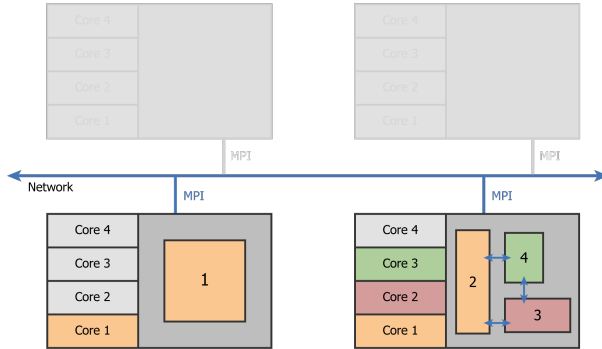
→ **MPI only:** one core per CPU for all meshes

Processing multi-mesh cases in FDS



- larger meshes should run exclusively on separate CPU's
- smaller meshes can be collected on a single CPU (must fit to storage!)
- may be more efficient due to communication overhead and waiting times

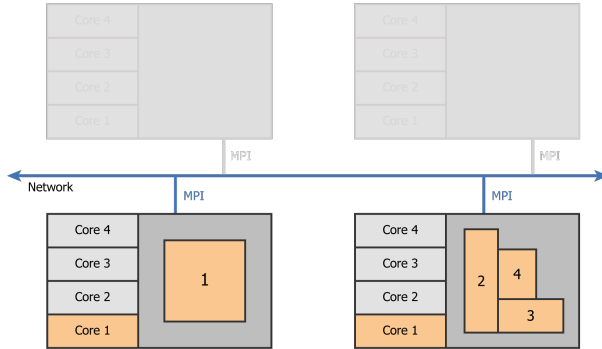
Processing multi-mesh cases in FDS



- three smaller meshes sampled on single CPU
- but separate process/memory for each mesh
- data transfer between meshes within CPU handled via MPI

→ **MPI only:** multiple cores used for meshes 2, 3, 4

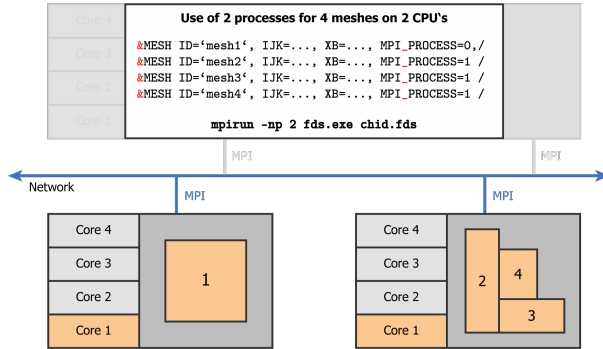
Processing multi-mesh cases in FDS



- three meshes are processed serially within one process
- no data transfer needed (common storage)

→ **MPI only:** single core used for meshes 2, 3, 4

Processing multi-mesh cases in FDS



- three meshes are processed serially within one process
- no data transfer needed (common storage)
- use parameter `MPI_PROCESS`

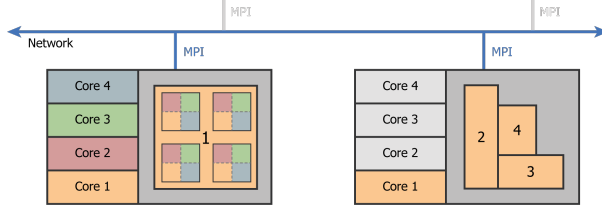
→ **MPI only:** single core used for meshes 2, 3, 4

Processing multi-mesh cases in FDS

Use of 2 processes for 4 meshes on 2 CPU's and OpenMP for mesh1:

```
&MESH ID='mesh1', IJK=..., XB=..., MPI_PROCESS=0, N_THREADS=4/  
&MESH ID='mesh2', IJK=..., XB=..., MPI_PROCESS=1 /  
&MESH ID='mesh3', IJK=..., XB=..., MPI_PROCESS=1 /  
&MESH ID='mesh4', IJK=..., XB=..., MPI_PROCESS=1 /
```

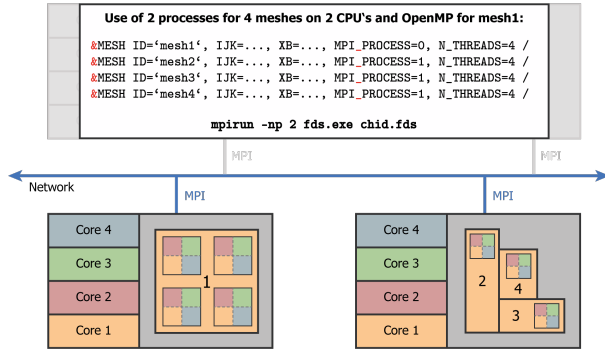
```
mpirun -np 2 fds.exe chid.fds
```



- parallelize suitable parts of code via corresponding OpenMP-directives
- use parameter N_THREADS

→ **Hybrid-MPI-OpenMP:** also using OpenMP for mesh 1

Processing multi-mesh cases in FDS



- parallelize suitable parts of code via corresponding OpenMP-directives
- use parameter N_THREADS
- can also be applied for the serial processing of meshes 2, 3, 4

→ **Hybrid-MPI-OpenMP:** also using OpenMP for all meshes

Efficiency issues

Limits of speedup

- scalability for realistic applications has its limits
- the optimal speedup is fairly never reached (linear growth with number of processors)
- fixed portion of purely serial components in algorithm cannot be performed in parallel
- with growing number of processors, speedup stronger depends on this serial fraction
- adding of more processors can even deteriorate the speedup in worst case

Challenge load balancing

- difficult to get a proper load balancing (comparable number of cells per mesh)
- probably long idle times due to waiting and synchronization
- slowest process will determine overall performance

Proper balance of MPI and OpenMP resources

Observations for FDS

- speedup of about 2, regardless of number of cores beyond 4
- OpenMP with more than 4 cores doesn't give appreciable speedup
- given same number of cores to run on, most of the speed up is achieved by MPI

Strategy

- experiment with different mesh configurations
- in case of troubles try to move mesh boundaries away from areas of activity
- limit the number of threads used by each MPI process
(Example: for 2 Quad-Cores don't use all 8 cores in an OpenMP simulation)