Towards smoke & fire simulation

with grid adaptive FEM

February 02, 2017 | Marc Fehling | Jülich Supercomputing Centre

JÜLICH
FORSCHUNGSZENTRUM

Introduction

- Marc Fehling
- born 1990
- M. Sc. Physics,
  Ruhr University Bochum, 2015

- since November 2015:
  PhD candidate at Jülich
  Supercomputing Centre,
  Division Civil Security & Traffic

Member of the Helmholtz-Association

Table of contents

Motivation: CFD in fire safety

- Modern architecture and large–scale projects (like BER) may not fit in 'building code'.
- Individual considerations necessary:
  - Designed to be cost efficient.
  - Model experiments, CFD investigations, ...

Figure: Interior of BMW World, Munich.

Modeling fires

Processes to model:

- Fluid dynamics.
  - Navier–Stokes equations.
  - Turbulence modeling.
- Radiation.
  - Discrete Transfer Radiation.
  - Discrete Ordinates.
  - Monte–Carlo.
- Combustion
  - Mixture fraction.
  - Finite rate kinetics.
- Pyrolysis.

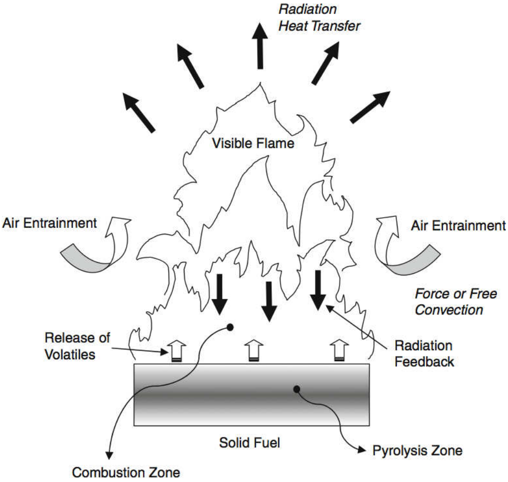


Figure: Burning solid fuel in air with physical processes involved. [1]

Modeling: Equations for smoke propagation

- Smoke propagation with incompressible Navier–Stokes (INS) equations:

$$\begin{aligned} \nabla \cdot \mathbf{u} &= 0 \\ \rho_0 \left[\partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla)\mathbf{u}\right] + \nabla p - \nabla(2\,\mu\,\epsilon_{ij}(\mathbf{u})) &= \mathbf{f}(T) \\ \rho_0 \left[\partial_t T + (\mathbf{u} \cdot \nabla) T\right] - 2\,\frac{\mu}{c_\mathrm{p}}\,\epsilon_{ij}(\mathbf{u}) : \nabla \mathbf{u} - \nabla \cdot \left(\frac{\mu}{\mathrm{Pr}} \nabla T\right) &= \gamma \end{aligned}$$

  with strain rate tensor $\epsilon_{ij}(\mathbf{u}) = \frac{1}{2}\left[\nabla \mathbf{u} + (\nabla \mathbf{u})^T\right]$.
- Turbulence model: Smagorinsky–Lilly LES [2]:

$$\nu = \nu_{mol} + \nu_{turb} \qquad \text{with} \qquad \nu_{turb} = (C_s h)^2 \left|\left|\epsilon_{ij}(\mathbf{u})\right|\right|_2 \;.$$

Numerical methods

Spatial discretization methods for computational fluid dynamics and software packages using it:

- Finite Difference Method (FDM).
  - Open source: NIST Fire Dynamics Simulator (FDS), ...
- Finite Volume Method (FVM).
  - Open source: FireFOAM (OpenFOAM), ...
- Finite Element Method (FEM).
- Lattice–Boltzmann Method (LBM).

FEM: Features

Why use FEM?

FEM: Features

Why use FEM?

- Unstructured grids.
  - Allow for better domain representation without aliasing.

FEM: Features

Why use FEM?

- Unstructured grids.
  - Allow for better domain representation without aliasing.
- hp–adaptivity.
  - Dynamic resolution of numerical grid.
  - Adaptive polynomial degree of basis functions.

  $\rightarrow$ Increase accuracy where the action is happening!

FEM: Features

Why use FEM?

- Unstructured grids.
  - Allow for better domain representation without aliasing.
- hp–adaptivity.
  - Dynamic resolution of numerical grid.
  - Adaptive polynomial degree of basis functions.

    $\rightarrow$ Increase accuracy where the action is happening!
- Discontinuous Galerkin (DG) methods.
  - Allow discontinuities across cell borders.
  - Continuous Galerkin (CG) methods unstable for advection like problems, thus require stabilization.

FEM: Formulation

- Solve variational equation from 'weak' formulation of the differential equation with bilinear form $a(u, v)$:

$$\exists u \in V : \forall v \in V : a(u, v) = f(v)$$

- Choose subspace V_{h} with basis w_{i}, out of which the approximate solution $u_{\mathrm{h}} = \sum u_i w_i \in V_{\mathrm{h}}$ will be constructed:

$$a(u_{\mathrm{h}}, w_{\mathrm{j}}) = \sum a(w_i, w_{\mathrm{j}})\ u_i = f(w_{\mathrm{j}}) \quad \rightarrow \quad AU = F$$

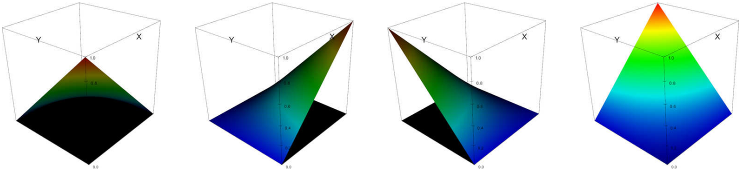


Figure: Q_1 elements in 2D (source: deal.II)

JuFire

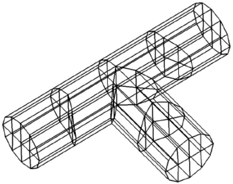
- Investigate applicability of Finite Element Methods (FEM) for fire simulation.
- Use open–source library deal.II [3].
  - ’Toolbox’ for the creation of FEM codes.

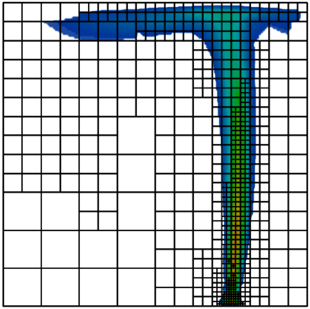
Differential **E**quations **A**nalysis **L**ibrary

JuFire: Features

Implemented:

- Unstructured grids.
- Continuous Galerkin Methods.
- Adaptive mesh refinement (AMR).
- MPI parallelization.
- Utilization of CAD models as manifolds for mesh refinement.





Current field of activity:

- Verification of flow solver.

Future work:

- Discontinuous Galerkin methods.
- p–adaptivity.

Verification: Determination of convergence order

- **FEM approach:** Error estimation with Bramble–Hilbert lemma [4].

$$||\mathbf{u} - \mathbf{u}_h||_{\mathcal{L}^2} \leq C\, h^{k+1} \; (||\mathbf{u}||_{\mathcal{H}^{k+1}} + ||p||_{\mathcal{H}^k})$$
$$||\mathbf{u} - \mathbf{u}_h||_{\mathcal{H}^1} \leq C\, h^{k} \; (||\mathbf{u}||_{\mathcal{H}^{k+1}} + ||p||_{\mathcal{H}^k})$$

- **FDS approach:** Root mean square (RMS) error at each point [5].

$$E_{\mathrm{rms}}(\mathbf{x}) = \sqrt{\frac{1}{M} \sum_{m=1}^{M} \left[u_h(\mathbf{x}, t_m) - u(\mathbf{x}, t_m)\right]^2} \;,$$

- RMS error resembles $\mathcal{L}^2$–error, so similar convergence orders for both RMS and Bramble–Hilbert approaches are expected.

Verification: Analysis

- Analysis with manufactured solution for INS equations from McDermott [6].
- Compare E_{rms} errors of FDS and JuFire.

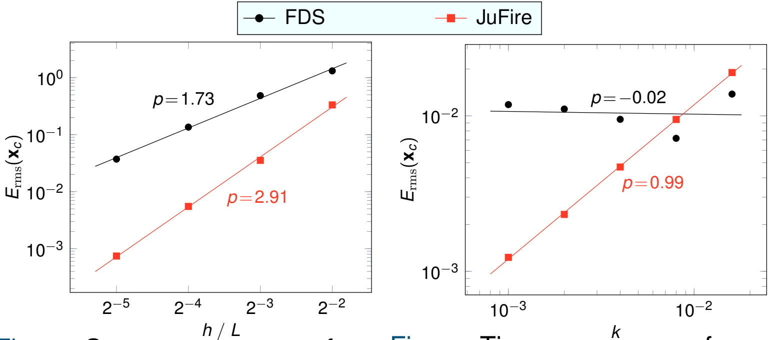


Figure: Space convergence for $k = 10^{-4}$, $\mu = 0$.

Figure: Time convergence for $h = \pi/16$, $\mu = 0$.

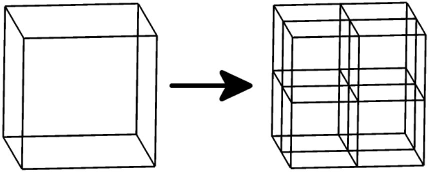
Adaptive mesh refinement (AMR)

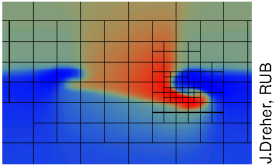
- Solution of differential equations with *FEM* requires:
  - Discretization of space in cells of length h.
  - Shape functions with polynomial degree p.
- <u>But:</u> The more accurate the solution by choosing h and p,
  the longer the computation.

Adaptive mesh refinement (AMR)

- Solution of differential equations with *FEM* requires:
  - Discretization of space in cells of length *h*.
  - Shape functions with polynomial degree *p*.
- But: The more accurate the solution by choosing *h* and *p*, the longer the computation.

→ **Adaptive refinement** as a 'compromise'.
  - Adjustment of parameters locally where necessary.
  - Variable mesh precision at runtime.





AMR: Algorithm

How to determine refinement criteria?

Set up decision criterion for refinement/coarsening.
Our choice:

- Calculate $||\nabla u||$ on each cell.
- Normalize values with respect to all cells.

AMR: Algorithm

How to determine refinement criteria?

Set up decision criterion for refinement/coarsening.
Our choice:

- Calculate $||\nabla u||$ on each cell.
- Normalize values with respect to all cells.
- Flag top/lower 30% for refinement/coarsening at each step.
- Define max/min refinement levels as upper/lower bounds.

JÜLICH FORSCHUNGSZENTRUM

AMR: Algorithm

How to determine refinement criteria?

Set up decision criterion for refinement/coarsening.
Our choice:

- Calculate $||\nabla u||$ on each cell.
- Normalize values with respect to all cells.
- Flag top/lower 30% for refinement/coarsening at each step.
- Define max/min refinement levels as upper/lower bounds.
- Cells are not allowed to differ by more than one level of refinement.

AMR: Example

- Demonstration of adaptive mesh refinement via moving vortex test case as a shape–preserving potential stream.

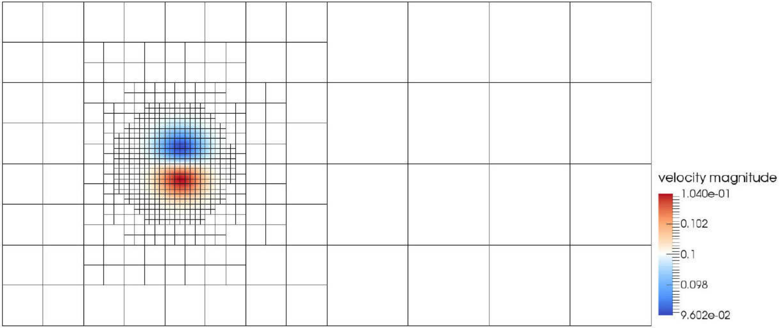
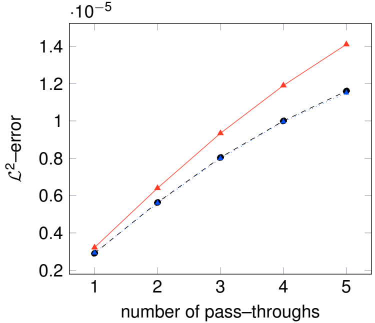


Figure: Video of velocity magnitude of moving vortex, overlayed with corresponding mesh.

AMR: Benefits

- Comparison of runtime and accuracy between uniform and adaptively refined meshes with the moving vortex example.



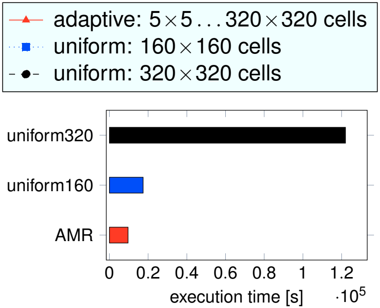


Figure: Global $\mathcal{L}^2$–errors at each periodic pass–through of the vortex.

Figure: Execution time of the vortex simulation, run in serial on a common desktop computer.

HPC on JURECA: Parallelization

- `MPI` parallelization with `Trilinos` and `p4est` through `deal.II` backends.

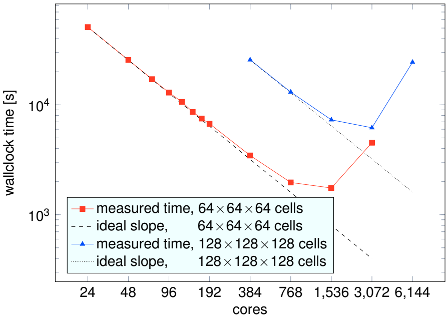


Figure: Strong scaling on JURECA for fixed 3D problems with 262,144 and 2,097,152 cells, respectively.

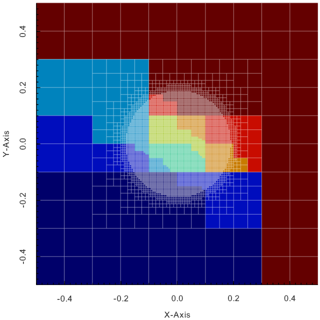


Figure: Examplary domain decomposition with `p4est` in an AMR case.

Summary & Outlook

Summary:

- Adaptive Mesh Refinement works satisfactorily (for now).
- Flow solver accomplishes first verification tests.

Future work:

- Implementation of models.
  - Buoyancy, radiation, combustion, pyrolysis, ...
- Extension of numerical methods.
  - DG methods, p-adaptivity, ...
- Comparison with other fire solvers.
- Validation using experiments.

Thank you for your kind attention!

References I

- Guan Heng Yeoh and Kwok Kit Yuen.
  Computational Fluid Dynamics in Fire Engineering – Theory, Modelling and Practice.
  Butterworth–Heinemann, 1st edition, 2009.
- Joseph Smagorinsky.
  General circulation experiments with the primitive equations.
  Monthly Weather Review, 91(3):99–164, March 1963.

References II

W. Bangerth, D. Davydov, T. Heister, L. Heltai, G. Kanschat, M. Kronbichler, M. Maier, B. Turcksin, and D. Wells.
The `deal.II` library, version 8.4.
Journal of Numerical Mathematics, 24, 2016.

Vivette Girault and Pierre-Arnaud Raviart.
Finite Element Methods for Navier–Stokes Equations: Theory and Algorithms, volume 5 of *Springer Series in Computational Mathematics*.
1986.

References III

Kevin McGrattan, Simo Hostikka, Randall McDermott, Jason Floyd, Craig Weinschenk, and Kristopher Overholt.

Fire Dynamics Simulator Technical Reference Guide Volume 2: Verification.
NIST Special Publication 1018–2, sixth edition, August 2016.

Randy McDermott.
A non–trivial analytical solution to the 2–D incompressible Navier–Stokes equations, August 2003.
[cited 26/09/2016].

References IV

- Cedric Taylor and P. Hood.
  A numerical solution of the Navier–Stokes equations using the finite element technique.
  Computers & Fluids, 1:73–100, 1973.
- Bernardino Roig.
  One–step Taylor Galerkin methods for convection–diffusion problems.
  Journal of Computational and Applied Mathematics, 204:95–101, 2007.

References V

N.D. Heavner and L.G Rebholz.
Locally chosen grad–div stabilization parameters for finite elemnt discretizations of incompressible flow problems.
SIAM Undergraduate Research Online (SIURO), 7.

Towards smoke and fire simulation: Addendum

with grid adaptive FEM

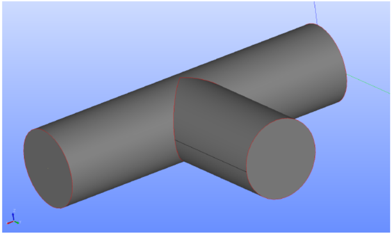
February 02, 2017 | Marc Fehling | Jülich Supercomputing Centre

JuFire: Algorithm

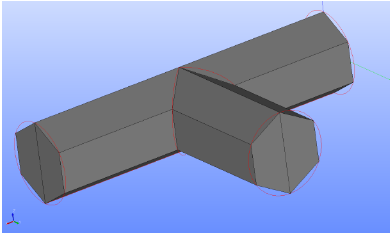
- Time evolution with backward differentiation formula (BDF)
- Taylor–Hood elements [7]: $(\mathbf{u}, p) \in Q_{k+1}^{\mathrm{dim}} \times Q_k$
- Decoupling of $(\mathbf{u}, p)$ by projection scheme
  - Leads to the pressure–Poisson equation
- Stabilization of momentum equation
  - Taylor–Galerkin stabilization [8] for Taylor–Hood elements
    - Additional diffusion in flow direction
  - Grad–div stabilization [9] to enforce $\nabla \cdot \mathbf{u} = 0$
- Neumann series for fast matrix assembly
- Boussinesq approximation for buoyancy force density

Unstructured Grids: Example: T–pipe

- Why T–pipe?
  - Compound bodies.
  - Crooked areas (→ cylinder casing area).
  - Discontinuous edges (→ 'welding seam').
- Develop procedure for the creation of appropriate initial meshes.



CAD model.



Initial mesh.

Iterations of global refinement: