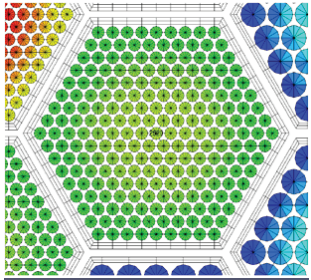
Development of Numerical Schemes for solving Multiphase Flows on General Meshes

Antoine Gerschenfeld, Yannick Gorsse

CEA - DM2S/STMF

14/06/2022

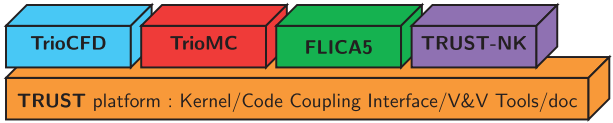
Introduction



Motivation

Development of the TrioMC code :

- two-phase sodium flows ($\rho_l/\rho_g \sim 2000$)
- based on the TRUST open-source platform developed at CEA (together with FLICA5, TrioCFD,...)
- non-regular meshes: hexahedra, prisms, tetrahedra...



Introduction

Equations

Large kinematic and thermal disequilibria $\rightarrow$ need the Euler-Euler system:

$$
\begin{array}{lll}
(\mathcal{M}_k) & \frac{\partial \alpha_k \rho_k}{\partial t} + \nabla.(\alpha_k \rho_k \vec{v}_k) & = \Gamma_k \\
(\mathcal{Q}_k) & \frac{\partial \alpha_k \rho_k \vec{v}_k}{\partial t} + \nabla.(\alpha_k \rho_k \vec{v}_k \otimes \vec{v}_k) & = -\alpha_k \nabla p + \vec{F}_{ki} + \Gamma_k \vec{v}_{ki} + \nabla.\alpha_k \mu_k (\nabla \vec{v}_k + {}^t \nabla \vec{v}_k) \\
(\mathcal{E}_k) & \frac{\partial \alpha_k \rho_k e_k}{\partial t} + \nabla.(\alpha_k \rho_k e_k \vec{v}_k) & = q_{ki} + \Gamma_k h_{ki} + \nabla.(\alpha_k \lambda_k \nabla T_k)
\end{array}
$$

- equations : mass/momentum/energy conservation per-phase $\rightarrow 3N$
- unknowns : α_k ($\sum \alpha_k = 1$), T_k, v_k, p (single-pressure) $\rightarrow 3N$
- equations of state: $\rho_k(p, T_k)$, $e_k(p, T_k)$
- transport properties: $\mu_k(p, Tk)$, $\lambda_k(p, Tk)$
- closure laws: Γ_k (phase change), F_{ki} (interfacial friction), q_{ki} (interfacial heat transfer)

Cartesian meshes

An industrial solution method

1. spatial discretisation → use MAC scheme (finite-volume) :
   - scalar variables (α_k, p, T_k) → cell averages : $[p]_c = \frac{1}{|c|}\int_c p(x)dV$
   - vector variables (v_k) → normal component averages: $[v_k]_f = \frac{1}{|f|}\int f \vec{v}_k \dot{d}\vec{S}$
2. time discretisation → semi-implicit:

$$
\begin{array}{ll}
(\mathcal{M}_k) & \frac{\alpha_k^+\rho_k^+ - \alpha_k^-\rho_k^-}{\Delta t} + \nabla.(\alpha_k^-\rho_k^- v_k^+) = \Gamma_k^+ \\
(\mathcal{Q}_k) & \alpha_k^-\rho_k^- \frac{v_k^+ - v_k^-}{\Delta t} + \nabla.(\alpha_k^-\rho_k^- v_k^- \otimes v_k^-) = -\alpha_k^-\nabla p^+ + F_{ki}^+ + D[v_k^-] + ... \\
(\mathcal{E}_k) & \frac{\alpha_k^+\rho_k^+ e_k^+ - \alpha_k^-\rho_k^- e_k^-}{\Delta t} + \nabla.(\alpha_k^-\rho_k^- e_k^- v_k^+) = q_{ki}^+ + q_{kp}^+ + D[T_k^-] + ...
\end{array}
$$

- p → fully implicit
- $(\mathcal{Q}_k)$ → all scalar variables explicit, only local terms (F_{ki}) implicit
- $(\mathcal{M}_k), (\mathcal{E}_k)$ → local terms explicit, transport terms implicit

Cartesian meshes

An industrial solution method

1. semi-implicit: time discretisation $\rightarrow$ why?
   $\rightarrow$ gives a block-diagonal Jacobian when solving Newton iterations:

$$\begin{pmatrix} \frac{\partial \mathcal{M}}{\partial \alpha} & \frac{\partial \mathcal{M}}{\partial T} & \frac{\partial \mathcal{M}}{\partial \vec{v}} \\ \frac{\partial \mathcal{E}}{\partial \alpha} & \frac{\partial \mathcal{E}}{\partial T} & \frac{\partial \mathcal{E}}{\partial \vec{v}} \\ 0 & 0 & \frac{\partial \mathcal{Q}}{\partial \vec{v}} \end{pmatrix} . \begin{pmatrix} \delta\alpha \\ \delta T \\ \delta \vec{v} \end{pmatrix} = \begin{pmatrix} \delta\mathcal{M} \\ \delta\mathcal{E} \\ \delta\mathcal{Q} \end{pmatrix} - \begin{pmatrix} \frac{\partial \mathcal{M}}{\partial p} \\ \frac{\partial \mathcal{E}}{\partial p} \\ \frac{\partial \mathcal{Q}}{\partial p} \end{pmatrix} .\delta p$$

   Solution method $\rightarrow$ pressure reduction :
   - inverse green block at each face $\rightarrow$ gives $\Delta v = A_v \delta p + b_v$
   - inverse blue block at each cell $\rightarrow$ gives $\Delta\alpha = A_\alpha \delta p + b_\alpha$, $\Delta T = A_T \delta p + b_T$
   - inject into $\sum \alpha_k = 1 \rightarrow$ gives linear system in δp
     - homogeneous $\rightarrow$ no scaling/precision problems
     - similar to Poisson equation $\rightarrow$ multigrid preconditioners apply
     - avoids the (p, v) saddle-point!

- limited by material CFL, but can be extended to CFL>1 using prediction steps

Challenges

Advantages/Disadvantages

- \+ all source terms (very strong) are implicit → very robust
- \+ staggered discretization → no low-Mach spurious modes
- \+ large reduction in final linear system → low cost
- − MAC scheme limited to Cartesian meshes
- − pressure reduction requires block-diagonal structure
  → constraint on potential alternative discretizations

⇒ can we find an alternative scheme applicable to polyhedral meshes?

Search

General recipe

1. choose a numerical scheme for scalar diffusion $-\nabla.(\Lambda\nabla u) = s$ with a finite-volume interpretation :

$$-\sum_{f\vee c} |f| F_{cf}([u]_C) = \int_c s dV \quad , \sum_{c\vee f} F_{cf}([u]_C) = 0$$

2. use the fluxes F_{fc} to discretize:
   - the thermal diffusion term $\nabla.(\alpha_k \lambda_k \nabla T_k)$ in the energy equations;
   - the pressure gradient term $-\alpha_k \nabla p$ in the momentum equtions
3. all other terms are easy, except for the momentum convection/diffusion terms
   $\rightarrow$ use tricks for these!

Search

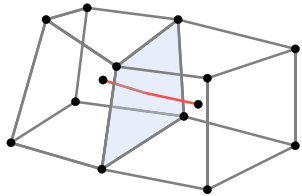
First scheme

Start from Hybrid Mixed Mimetic schemes (Eymard, Droniou, Bonnelle, da Veiga, Lipnikov, Manzini...)

- unconditionally stable on star-shaped meshes
- first formulation $\rightarrow$ mixed:

$$-\nabla.(\Lambda\nabla u) = f \Rightarrow \left\{ \begin{array}{l} \vec{\varphi} = -\lambda\nabla u \\ \nabla.\vec{\varphi} = s \end{array} \right. \Rightarrow \left\{ \begin{array}{l} M_2(\lambda)[\varphi]_f = [\varphi]_{\bar{f}} = |f|([u]_{am(f)} - [u]_{av(f)}) \\ \sum_{f \vee c} |f|[\varphi]_{cf} = |c|[s]_c \end{array} \right.$$

with $M_2(\lambda)$ a SPD matrix relating $[\varphi]_f$ to the dual face integral $[\varphi]_{\bar{f}} = \int_{am(f)}^{av(f)} \vec{\varphi}.d\vec{l}$



Search

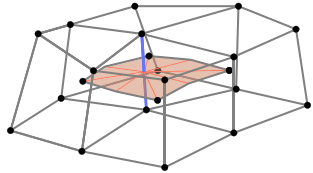
First scheme

- extended to Stokes by Bonnelle (then to Navier-Stokes by B. Koren et al.) using the identity $\Delta\vec{v} = -\nabla \wedge \vec{\omega}$ with $\vec{\omega} = \nabla \wedge \vec{v}$ the vorticity:

$$\left\{ \begin{array}{l} \frac{\partial \vec{v}}{\partial t} = -\nabla p + \nu \Delta \vec{v} \\ \nabla . \vec{v} = 0 \end{array} \right. \Rightarrow \left\{ \begin{array}{rl} \frac{\partial \vec{v}}{\partial t} + \nabla \wedge \vec{\omega} + \nabla p & = \vec{0} \\ -\frac{1}{\nu}\vec{\omega} + \nabla \wedge \vec{v} & = \vec{0} \\ \nabla . \vec{v} & = 0 \end{array} \right.$$

$$\Rightarrow \left\{ \begin{array}{rl} M_2 \partial_t [v]_F + M_2 R_F [\omega]_E + G[p]_C & = 0 \\ -M_1(\nu^{-1})[\omega]_A + R_A M_2 [v]_F & = 0 \\ D[v]_F & = 0 \end{array} \right.$$

with M_1 another matrix relating edge dual to edge integrals of $\vec{\omega}$ and R_A, R_F discrete curl operators around faces and edges:



First scheme

$$\left\{\begin{array}{rl} M_2\partial_t[v]_F + M_2R_F[\omega]_E + G[p]_C & = 0 \\ -M_1(\nu^{-1})[\omega]_A + R_AM_2[v]_F & = 0 \\ D[v]_F & = 0 \end{array}\right.$$

can this be solved by pressure reduction?

1. compute $[\omega^-]_E$ from $[v^-]_F$ by solving $M_1(\nu^{-1})[\omega]_A = R_AM_2[v^-]_F = 0$
   - not block-diagonal
   - but only needed once per time-step
2. compute $[v^+]_F$ from $M_2\frac{[v^+]_F-[v^-]_F}{\Delta t} + M_2R_F[\omega^-]_E + G[p^+]_C = 0$
   $\rightarrow$ not possible locally! Instead, must solve the saddle-point

$$\begin{pmatrix} \frac{M_2}{\Delta t} & G \\ -D & 0 \end{pmatrix}\begin{pmatrix} [v^+]_F \\ [p^+]_C \end{pmatrix} = \begin{pmatrix} \frac{M_2}{\Delta t}[v^-]_F - M_2R_F[\omega^-]_E \\ 0 \end{pmatrix}$$

- incompressible flow : constant system $\rightarrow$ can use direct solver
- compressible/multiphase flow: variable matrix $\rightarrow$ must use iterative solver!

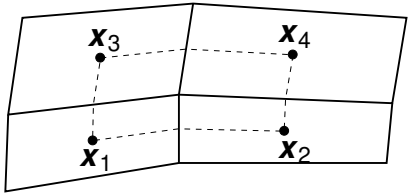
First scheme: conclusion

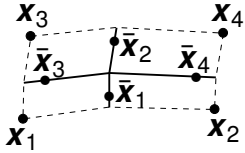
- many desirable properties: stability, symmetry (for Stokes),...
- but two incompatibilities with pressure reduction:
  - auxiliary variables to solve diffusion (vorticities $[\omega]_E$)
    $\rightarrow$ inconvenient, but manageable
  - mass matrix in momentum equation
    $\rightarrow$ forces a saddle-point system $\rightarrow$ pressure reduction impossible

$\Rightarrow$ the search continues!

Second scheme : avoiding pitfalls

- start with a classical cel-centered diffusion scheme
  $\rightarrow$ MPFA-O (Aavatsmark)





  - start from $[u]_c$ (value at cell)
  - for each (face, vertex) pair (f, v), introduce a variable u_{fv}
  - in each cell, $(u_c, u_{f_1 v}, u_{f_2 v})$ define a gradient $[\nabla u]_{cv}$
  - at each face around v, impose $\vec{n}_f.\Lambda[\nabla u]_{cv} = \vec{n}_f.[\nabla u]_{c'v}$
  - solve the local linear system on the (u_{fv}) around each vertex v
    $\rightarrow$ flux $F_f = -|f|\vec{n}_f.\Lambda\nabla u = F_f([u]_C)$!

- conditionally stable (but rather robust in practice)

Search

Second scheme: discretization

- mass equation $(\mathcal{M}_k) \rightarrow$ discretized in each cell c :
  - convective term : $|c|[\nabla.\alpha_k\rho_k\vec{v}_k]_c = \sum_{f\vee c}|f|[\alpha_k\rho_k]_f[v_k]_{cf}$ with $[\alpha_k\rho_k]_f$ chosen by a convection scheme (usually upwind)
  - other terms: local
- mass equation $(\mathcal{M}_k) \rightarrow$ discretized in each cell c :
  - convective term : same as in mass equation
  - diffusive term $[\nabla.(\alpha_k\lambda_k\nabla T_k)]_c$: computed using MPFA-O fluxes
  - other terms: local
- momentum equation $(\mathcal{Q}_k) \rightarrow$ discretized at faces (normal components):
  - pressure gradient $-\alpha_k\nabla p$: computed using MPFA-O fluxes
  - convective/diffusive terms: see next slide
  - all other terms: local

Second scheme: discretization

How to discretize the momentum convection/diffusion terms without altering the linear system structure?

$\rightarrow$ introduce cell velocities $[\vec{v}]_c$:

1. interpolate $[\vec{v}]_c$ from $[v]_f$:
   - at 1^{st} order $\rightarrow$ using "magical identity":

$$[\vec{v}_c] = \frac{1}{|c|} \sum_{f \vee c} |f| [v]_{cf} (\vec{x}_f - \vec{x}_c)$$

   - at 2^{nd} order (needed for diffusion) $\rightarrow$ possible with more neighbours
2. compute momentum convection/diffusion at cells:
   - $[\nabla.(\alpha_k \rho_k \vec{v}_k \otimes \vec{v}_k)]_c$ using a convection scheme on the $[\vec{v}]_c$
   - $[\nabla.\alpha_k \mu_k (\nabla \vec{v}_k +^t \nabla \vec{v}_k)]_c$ using MPFA-O fluxes
3. interpolate the needed face values by combining cell values :

$$[\nabla.(\alpha_k \rho_k \vec{v}_k \otimes \vec{v}_k)]_f = \mu \vec{n}_f.[\nabla.(\alpha_k \rho_k \vec{v}_k \otimes \vec{v}_k)]_{am(f)} + (1-\mu)\vec{n}_f.[\nabla.(\alpha_k \rho_k \vec{v}_k \otimes \vec{v}_k)]_{av(f)}$$

Search

Second scheme: properties

- no auxiliary variables!
  (except when using a prediction step $\rightarrow$ linear system in $([v]_f, [\vec{v}]_c)$
- no spurious oscillations despite using "collocated" momentum operators
  $\rightarrow$ the primary velocities are still the staggered $[v]_F$
- diagonal mass matrix in the momentum equations
  $\rightarrow$ pressure reduction possible

Drawbacks:

- conditional stability
  $\rightarrow$ alleviated by sacrificing precision for stability on deformed meshes
- high numerical cost on tetrahedra
  $\rightarrow$ stencil in each cell extends to cells sharing one of its vertices (often $\gtrsim 100$)

$\Rightarrow$ scheme implemented as PolyMAC_P0
But can the first scheme be fixed?

Search

Third scheme: back to HMM

HMM schemes have a second formulation (hybrid form) using face scalar unknowns instead of fluxes to solve $-\nabla.\Lambda\nabla u = s$

$\rightarrow$ using a SPD matrix $W_2^c(\Lambda)$ in each cell:

$$-[\Lambda\nabla u]_{cf} = \sum_{f'\vee c} W^c_{2ff'}(\Lambda)(u_{f'} - u_c)$$

$\rightarrow$ the u_f are determined by the equations $[\Lambda\nabla u]_{am(f)f} + [\Lambda\nabla u]_{av(f)f} = 0$, leading to the (SPD) linear system

$$\begin{cases} -\sum_{f\vee c} |f|[\Lambda\nabla u]_{cf} = |c|[s]_c & \forall c \\ [\Lambda\nabla u]_{am(f)f} + [\Lambda\nabla u]_{av(f)f} = 0 & \forall f \end{cases}$$

Third scheme: discretization

- mass equation $(\mathcal{M}_k)$: as usual
- energy equation $(\mathcal{E}_k)$: introduce face temperatures $[T_k]_f$ to compute the diffusive term $[\nabla.(\alpha_k \lambda_k \nabla T_k)]_c$
- momentum equation $(\mathcal{Q}_k) \rightarrow$ integrated at faces (normal components)
  - mass matrix is diagonal!
  - pressure gradient : introduce $[p]_f$ to compute $[\nabla p]_f$
    (system is closed using $[\nabla p]_{cf} + [\nabla p]_{c'f} = 0$)
  - convection term: computed at cells (using 1^{st}-order interpolation), then projected
  - momentum difusion: computed using vorticity variables
    using the identity $\nabla \wedge (\mu \nabla \wedge \vec{v}) = \nabla.(\mu^t \nabla \vec{v}) - \nabla.(\mu \nabla \vec{v})$

Third scheme : properties

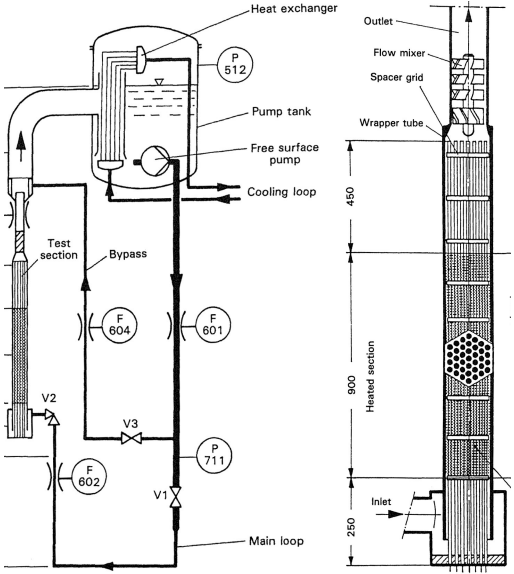
- stable on star-shaped meshes $\rightarrow$ like the original HMM scheme
- several auxiliary variables:
  - face temperatures $[T_k]_f$, vorticities $[\omega_k]_e$ $\rightarrow$ computed once per time step
  - face pressures $[p]_f$ $\rightarrow$ included in the reduced pressure system
- but pressure reduction is possible!

Main uses:

- very deformed meshes (but PolyMAC_P0 is hard to beat in practice...)
- meshes consisting mainly in tetrahedra

implemented as PolyMAC_P0P1nc (or "PolyMAC")

Applications

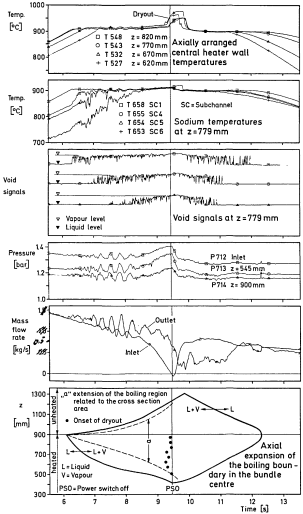


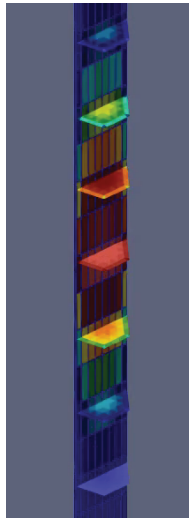
Two-phase sodium : KNS-37 L22 test

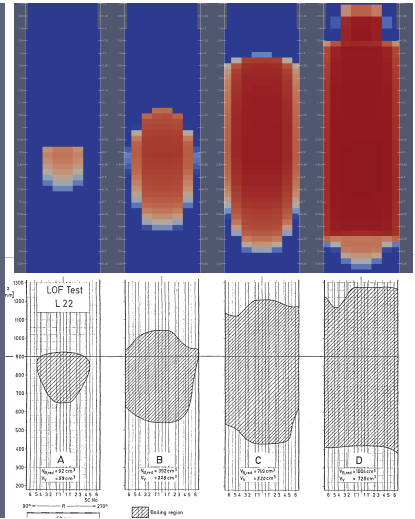
- the reference for sodium boiling!
- 37 pin, electrically-heated reactor element ($\sim$ 700 KW)
- loss of flow-type transient:
  - $t = 0$: pump trip ($t_{1/2} \sim 2.5$s)
  - $t = 6.3$s : local boiling (does not obstruct flow)
  - $t = 8.5$s : generalized boiling $\rightarrow$ blockage : flow redistribution
  - $t = 9.45$s : dry-out $\rightarrow$ electrical power trip

Applications

KNS-37 L22 : overall behavior, boiling zone

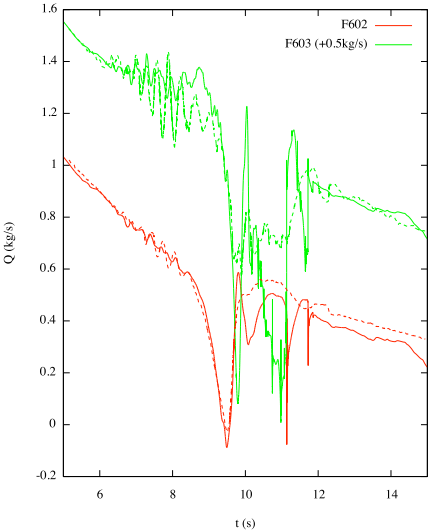


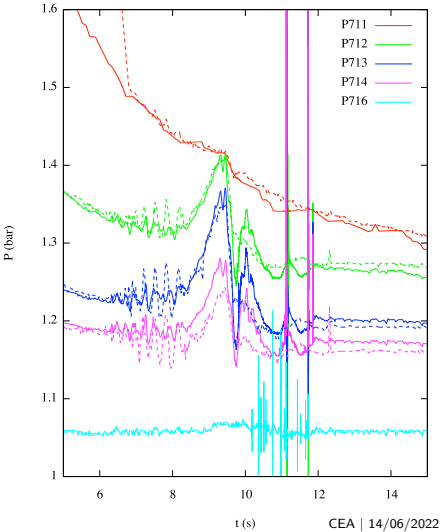




Applications

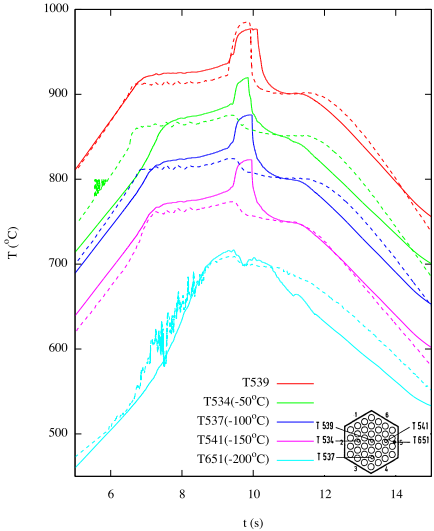
KNS-37 L22: flowrate, pressure

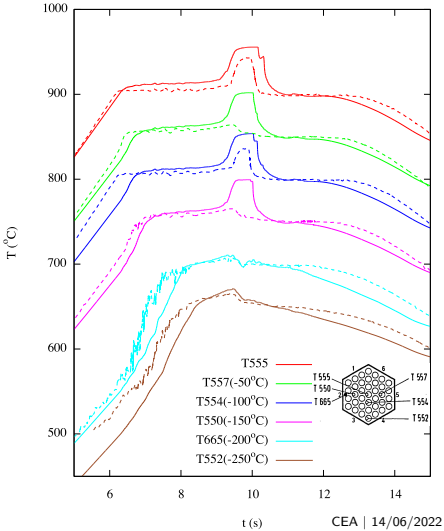




Applications

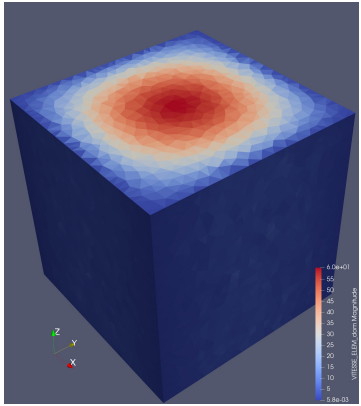
KNS-37 L22: pin temperature at 2/3rds + top of heated length





Applications

Navier-Stokes 3D benchmark

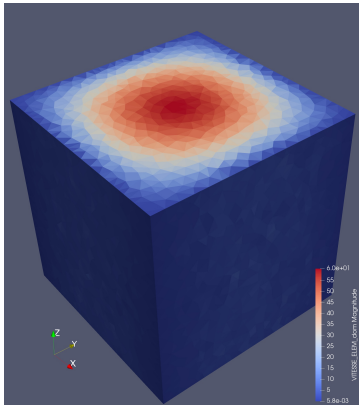


- proposed by M. Ndjinga for this symposium
- 3D manufactured solution (Poiseuille-like) for N-S:

$$\vec{v}_{ana} = x(1-x)y(1-y)\vec{e}_z$$

- transient simulation from $\vec{u} = 0$ at $CFL = 10^3$: need either
  - full $(\vec{v}, p)$ system $\rightarrow$ saddle-ppint:
    direct solvers (used here), augmented Lagrangian...
  - prediction-correction
    1. prediction $\rightarrow \vec{v}^*$ with $\nabla.\vec{v}^* \neq 0$:
       PolyMAC_P0 : system in $([v^*]_f, [\vec{v}^*]_c)$
       PolyMAC_P0P1nc : system in $([v^*]_f, [\omega^*]_e)$
       $\rightarrow$ saddle-point!
       but still solvable by iterative solvers (BCGS here)
    2. correction $\rightarrow$ elliptic system on p^+ to obtain $\nabla.\vec{v}^+ : 0$:
       PolyMAC_P0 : system in $([p^+]_c)$
       PolyMAC_P0P1nc : system in $([p^+]_c, [p^+]_f)$

Applications



Navier-Stokes 3D benchmark

Performance results (Apple M1, single-core):

	Poly_P0		Poly_P0P1nc	
	Full	Pred/corr	Full	Pred/corr
Hexa_4	52	0.55	64	0.65
Hexa_5	-	7.0	-	9.8
Tetra_3	1341	62.6	558	3.27
Tetra_6	-	827	-	118

- comparable convergence in time for both schemes
  $\rightarrow$ 10-11 time steps at CFL=1000 for both
- P0 faster on hexa, P0P1nc faster on tetra
- solving the full system via direct solvers does not scale...

Conclusion

- search for numerical schemes with "MAC-like" properties for multiphase flows
- strong constraints to allow the same pressure-reduction method as on Cartesian meshes:
  - block-diagonal structure in mass/energy equations $\rightarrow$ easy
  - but also in momentum equations $\rightarrow$ harder!
- two schemes implemented:
  - PolyMAC_P0 : based on MPFA-O
  - PolyMAC_P0P1nc : based on HMM
- when using CFL> 1, we need a direct solver to solve the $(\vec{v}, p)$ system
  $\rightarrow$ prediction/correction is still the best option
  (some schemes lead to a saddle-point in the correction step $\rightarrow$ KO)
  $\rightarrow$ this could be improved in the future!