Bringing PETSc to the Multi-Scale Simulation of Newtonian and non-Newtonian Free-Surface Flows

Juan Luis Prieto[1]

[1]Department of Energy Engineering, ETSII, Madrid

2nd International PETSc User Meeting 2016

Vienna (Austria), June 28th-30th

Outline

INDUSTRIALES
ETSII | UPM

Outline

INDUSTRIALES
ETSII | UPM

Outline

Outline

Outline

Motivation: Multi-phase simulation

Challenges and techniques

Importance of free-surface flows

Scientific, engineering and artistic applications:

- Combustion.
- Polymer extrusion.
- Bubbles.
- Waves.
- Image reconstruction.
- Shape recognition, etc.

Numerical difficulties

- Different densities/viscosities.
- Breaking up, merging.
- Area (volume) loss.

Available techniques

1. Lagrangian schemes: MAC, ALE, SPH.
2. Eulerian schemes: VOF, LS.
3. Hybrid schemes: CLSVOF, HPLS.

Motivation: Multi-phase simulation

Challenges and techniques

Importance of free-surface flows

Scientific, engineering and artistic applications:

- Combustion.
- Polymer extrusion.
- Bubbles.
- Waves.
- Image reconstruction.
- Shape recognition, etc.

Numerical difficulties

- Different densities/viscosities.
- Breaking up, merging.
- Area (volume) loss.

Available techniques

1. Lagrangian schemes: MAC, ALE, SPH.
2. Eulerian schemes: VOF, LS.
3. Hybrid schemes: CLSVOF, HPLS.

Motivation: Multi-phase simulation

Challenges and techniques

Importance of free-surface flows

Scientific, engineering and artistic applications:

- Combustion.
- Polymer extrusion.
- Bubbles.
- Waves.
- Image reconstruction.
- Shape recognition, etc.

Numerical difficulties

- Different densities/viscosities.
- Breaking up, merging.
- Area (volume) loss.

Available techniques

1. Lagrangian schemes: MAC, ALE, SPH.
2. Eulerian schemes: VOF, LS.
3. Hybrid schemes: CLSVOF, HPLS.

Motivation: Multi-phase simulation

Challenges and techniques

Importance of free-surface flows

Scientific, engineering and artistic applications:

- Combustion.
- Polymer extrusion.
- Bubbles.
- Waves.
- Image reconstruction.
- Shape recognition, etc.

Numerical difficulties

- Different densities/viscosities.
- Breaking up, merging.
- Area (volume) loss.

Available techniques

1. Lagrangian schemes: MAC, ALE, SPH.
2. Eulerian schemes: VOF, LS.
3. Hybrid schemes: CLSVOF, HPLS.

Motivation: Multi-phase simulation

Challenges and techniques

Importance of free-surface flows

Scientific, engineering and artistic applications:

- Combustion.
- Polymer extrusion.
- Bubbles.
- Waves.
- Image reconstruction.
- Shape recognition, etc.

Numerical difficulties

- Different densities/viscosities.
- Breaking up, merging.
- Area (volume) loss.

Available techniques

1. Lagrangian schemes: MAC, ALE, SPH.
2. Eulerian schemes: VOF, LS.
3. Hybrid schemes: CLSVOF, HPLS.

Non-Newtonian multi-phase flows

Example

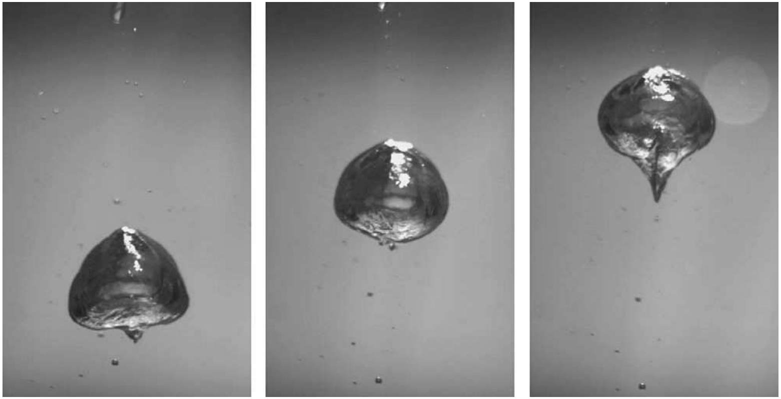
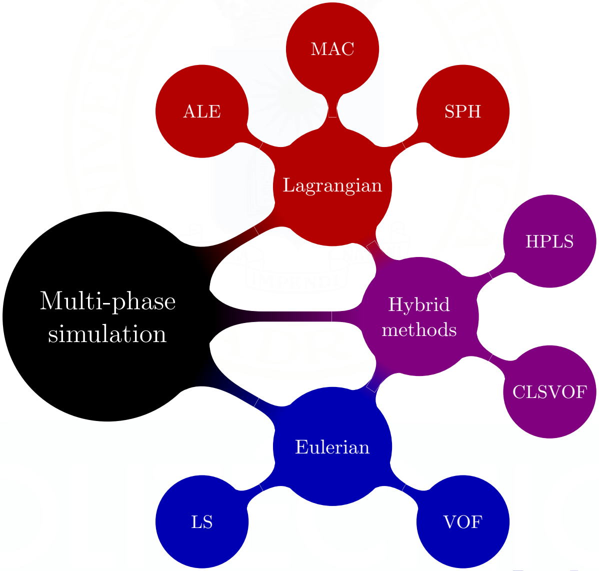


Figure: After the impact of a $d = 1.27cm$ steel sphere falling from $h0 = 70cm$, a large air bubble (about 2 cm^3) entrained by the sphere rises through the fluid. The images are separated by $\Delta t = 33ms$. [Reprinted from J. Non-Newtonian Fluid Mech, Vol. 135, B. Akers and A. Belmonte, 'Impact dynamics of a solid sphere falling into a viscoelastic micellar fluid', Pages 97-108, Copyright (2006), with permission from Elsevier].

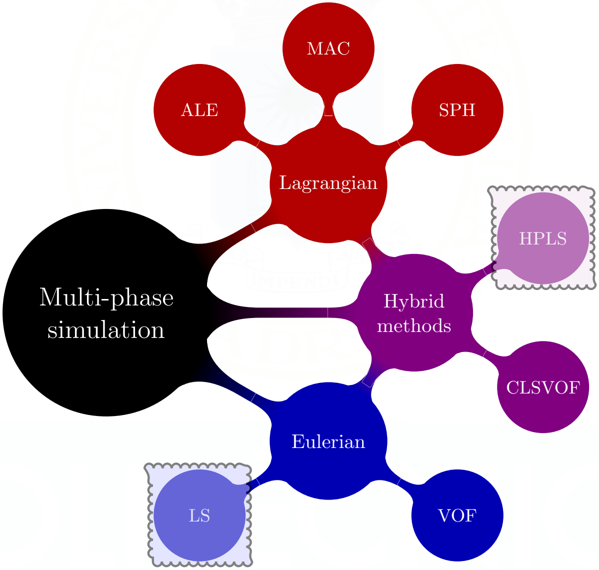
Multi-phase simulation

Available methods



Multi-phase simulation

Available methods



Semi-Lagrangian, Finite Element Level Set

Main features I: method of the characteristics

Method of the characteristics

Semi-Lagrangian formulation of the Navier-Stokes equations[1].

Makes use of

- Equations of the characteristic curves:

$$\frac{d\boldsymbol{X}(\boldsymbol{x}, t; \tau)}{d\tau} = \boldsymbol{v}\left(\boldsymbol{X}(\boldsymbol{x}, t; \tau), \tau\right); \qquad \boldsymbol{X}(\boldsymbol{x}, t; t) = \boldsymbol{x}.$$

$$\boldsymbol{X}(\boldsymbol{x}, t; s) = \boldsymbol{x} - \int_x^t \boldsymbol{v}\left(\boldsymbol{X}(\boldsymbol{x}, s; \tau), \tau\right) d\tau; \;\; \boldsymbol{X}^n \equiv \boldsymbol{X}\left(\boldsymbol{x}, t_{n+1}; t_n\right).$$

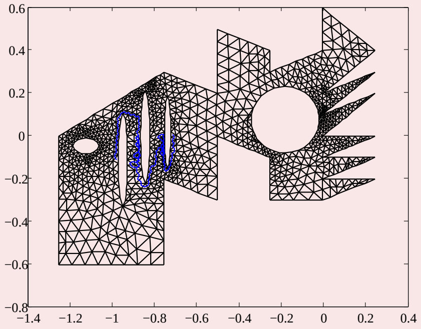
- Backward Difference Formula (BDF2) for temporal discretization of momentum equation.

[1] Allievi and Bermejo 2000, *Int. J. Numer. Meth. Fluids*.

Semi-Lagrangian, Finite Element Level Set

Main features II: improved search-and-locate algorithm

Multiply connected domains



Search and locate algorithm

- Valid for any unstructured bi-dimensional mesh[2].
- Combines finite element techniques with geometrical considerations[3].

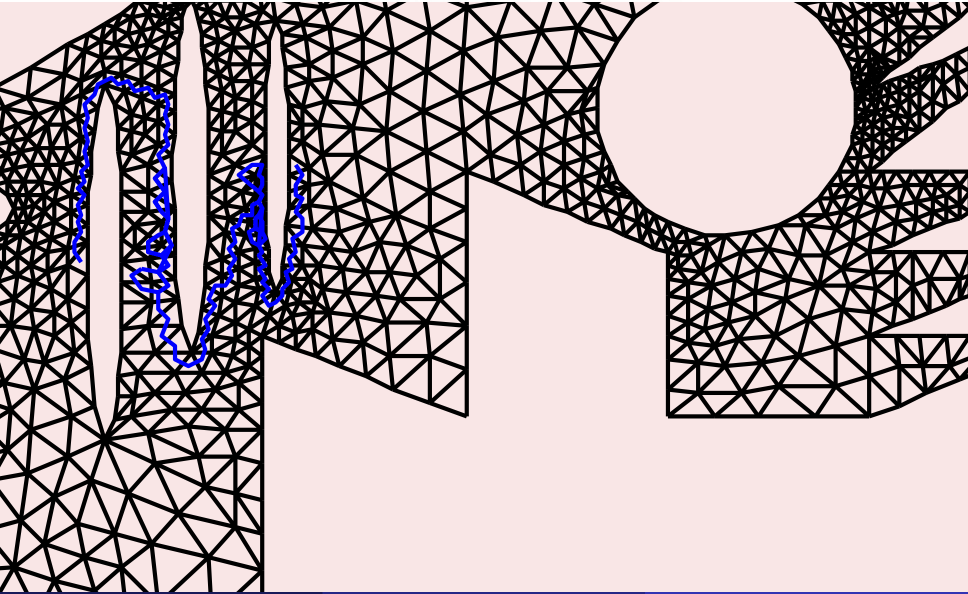
Continue...

[2] Allievi and Bermejo 1997, *J. Comput. Phys.*

[3] Prieto, Bermejo, and Laso 2010, *J. Non-Newtonian Fluid Mech.*

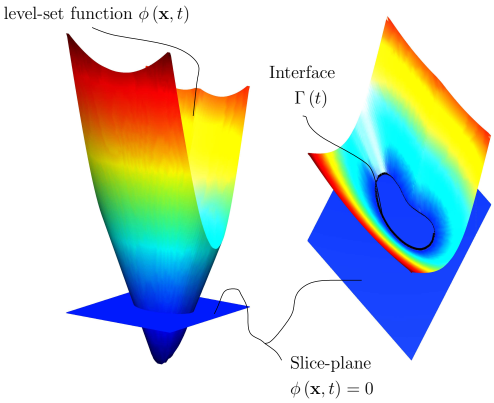
Semi-Lagrangian, Finite Element Level Set

Main features II: improved search-and-locate algorithm



Level-set method

General notes



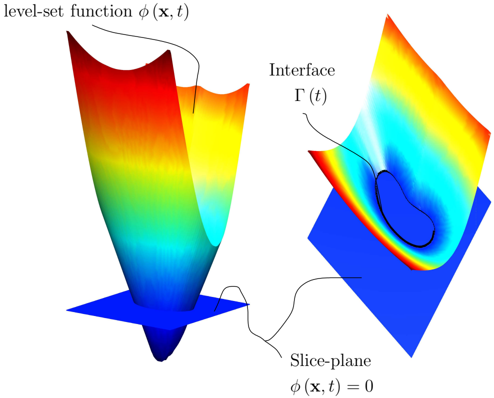
level-set function $\phi(\mathbf{x}, t)$
Interface $\Gamma(t)$
Slice-plane $\phi(\mathbf{x}, t) = 0$

Level-set approach

- Use of implicit function $\phi(\boldsymbol{x}(t), t)$.
- Interphase $\mathit{\Gamma}$ computed as zero iso-contour of ϕ.
- Signed-distance function (*Reinitialization*).

Level-set method

General notes


level-set function $\phi(\mathbf{x}, t)$
Interface $\Gamma(t)$
Slice-plane $\phi(\mathbf{x}, t) = 0$

Features and flaws

- Natural breaking-up and merging.
- Normal $\boldsymbol{n}$ available.
- Marker particles (hybrid approach).
- Extension to 3-d.
- Loss of area (volume).

Level-set method

Mathematical formulation

Definition and procedure

1. ϕ initialized as signed distance function ($\|\nabla\phi\| = 1$).
2. Iso-contours C: $\phi\left(\boldsymbol{x}\left(t\right), t\right) - C = 0$.
3. Evolution of ϕ with flow field: $\frac{D\phi}{Dt} = \frac{\partial\phi}{\partial t} + \boldsymbol{v}\cdot\nabla\phi = 0$.
4. Density, viscosity as function of ϕ: sharp integration across Γ_h.
5. Reinitialization procedure:
    - Prevents irregularities from developing at Γ_h.
    - Flow-of-time eikonal equation [4].
    - Solution inside a band (computationally efficient):

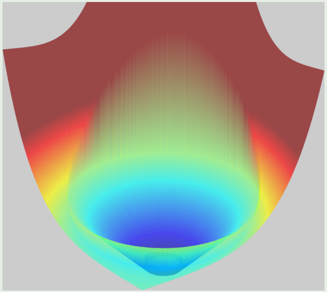
$$\begin{cases} \dfrac{D_{\boldsymbol{n}_u}u}{Dt} \equiv \dfrac{\partial u}{\partial t} + \boldsymbol{n}_u\cdot\nabla u = 0; \quad u\left(\boldsymbol{x}, 0\right) = u_0\left(\boldsymbol{x}\right) = \phi_0\left(\boldsymbol{x}\right); \\ \dfrac{D_{\boldsymbol{n}_v}v}{Dt} \equiv \dfrac{\partial v}{\partial t} + \boldsymbol{n}_v\cdot\nabla v = 0; \quad v\left(\boldsymbol{x}, 0\right) = v_0\left(\boldsymbol{x}\right) = -\phi_0\left(\boldsymbol{x}\right), \end{cases}$$

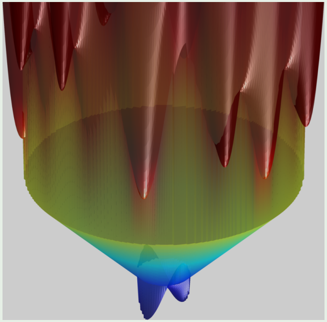
[4] Cheng and Tsai 2008, *J. Comput. Phys.*

Level-set method

Mathematical formulation

Reinitialization procedure: two examples of ϕ_0





Level set functions after the eikonal reinitialization algorithm:
$\phi_0(x, y) = \exp(x + y)\left(x^2 + y^2 - \frac{1}{4}\right)$ (left panel), and
$\phi_0(x, y) = [\sin(4\pi x)\sin(4\pi y) + 2]\left[\exp\left(x^2 + y^2 - \frac{1}{4}\right) - 1\right]$ (right panel).

Time step size $\tau = 10^{-2}$, number of time steps $N_\tau = 35$, and grid size $h = 10^{-2}$.

▸ More on the eikonal redistancing...

Particle Level-set method

Hybrid scheme

Concepts for marker particles

- Improves our previous work[4].
- Incorporated into our semi-Lagrangian scheme.
- Used in under-resolved (sub-mesh) regions.
- Local level-sets try to "correct" the global ϕ.
- Particles follow fluid trajectories: $\frac{d\boldsymbol{x}_p}{dt} = \boldsymbol{v}\left(\boldsymbol{x}_p, t\right)$.
- Radius $r_p = s_p \phi\left(\boldsymbol{x}_p\right)$, $r_{min} \leq r_p \leq r_{max}$ assigned to each particle.
- Particles initially placed in band around Γ; at outer region (*positive*, $s_p = 1$) and inner region (*negative*, $s_p = -1$)

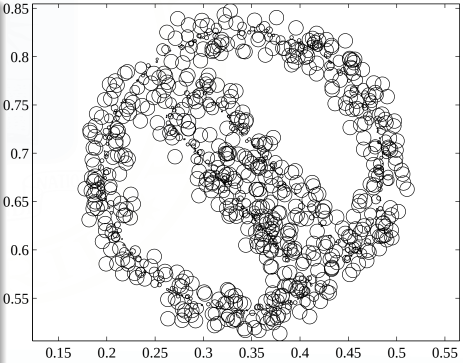
[4] Bermejo and Prieto 2013, *SIAM J. Sci. Comput.*

Marker particles and LS

Algorithm

Stages

- Error identification:
  Only *escaped* particles used to correct ϕ.
- Error quantification:
  A *local level set* is defined for each particle.
- Correction of ϕ:
  Auxiliary *positive* ϕ^+ and *negative* ϕ^- defined using positive and negative *escaped* particles[5].

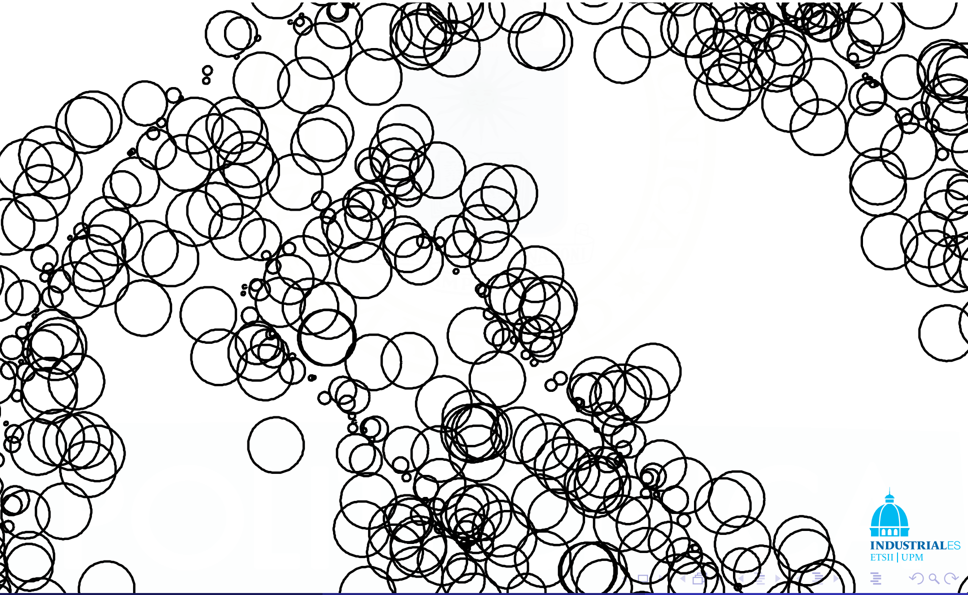


Continue...

[5] Enright et al. 2002, *J. Comput. Phys.*

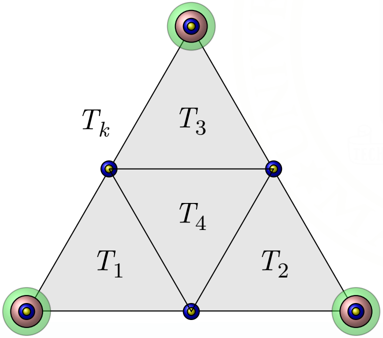
Marker particles and LS

Algorithm



INDUSTRIALES
ETSII | UPM

Finite element spaces
Full level set implementation



Notation

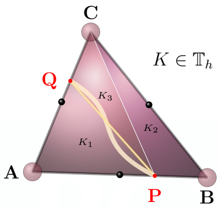
●	P1-iso-P2 Level set function ϕ
●	P2 Velocity $\boldsymbol{v}$
●	P1-disc Pressure p
●	P1 Polymer stress tensor $\boldsymbol{\tau}_p$

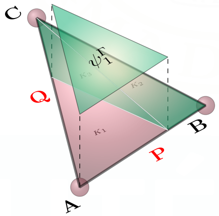
Finite element spaces

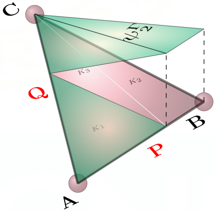
Discontinuous pressure space

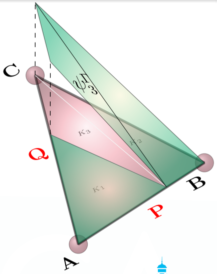
Why a discontinuous pressure space?

- ✓ To better capture pressure jumps:
  Surface tension, with Laplace-Beltrami operator to by-pass curvature κ.
- ✓ To reduce spurious currents (interface). ▸ More on Laplace-Beltrami...
- ✓ Not additional degrees of freedom:
  Locally enriched. Outside the interface, "old" pressure space[6].









[6] Ausas, Sousa, and Buscaglia 2010, *Comput. Methods. Appl. Mech. Engrg.*

Non-Newtonian multi-phase flows

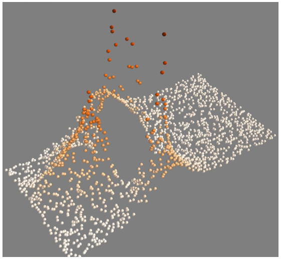
Micro-macro, multiscale approach

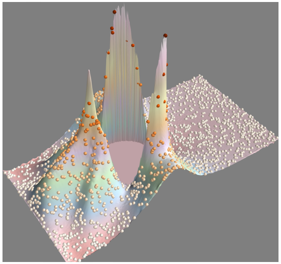
- “Polymer particles” scattered over the domain.
- Brownian dynamics simulation (stochastic approach).
- Two kinetic models: Hooke (Oldroyd-B) and FENE (‘Finitely Extensible Nonlinear Elastic’)[7]:

$$d\boldsymbol{Q} = \left(\boldsymbol{\kappa} \cdot \boldsymbol{Q} - \frac{1}{2De} \boldsymbol{Q} \right) dt + \frac{1}{\sqrt{De}} d\boldsymbol{W}, \text{Hooke;}$$

$$d\boldsymbol{Q} = \left(\boldsymbol{\kappa} \cdot \boldsymbol{Q} - \frac{1}{2De} \frac{\boldsymbol{Q}}{1 - \|\boldsymbol{Q}\|^2/b} \right) dt + \frac{1}{\sqrt{De}} d\boldsymbol{W}, \text{FENE.}$$

- Polymer stress tensor $\boldsymbol{\tau}_p$ (extra-stress tensor) in the momentum equation.
- Variance-reduced formulation (a la ‘Brownian Configution Fields’).
- Compactly Supported Radial Basis Function (CSRBF) reconstruction of $\boldsymbol{\tau}_p$[8].





[7] Öttinger 1996.

[8] Prieto 2016a, *J. Non-Newtonian Fluid Mech.*

INDUSTRIALES ETSII | UPM

Why did I start using PETSc?

Because the multi-phase simulations were running SLOW.

Why did I start using PETSc?

Because the multi-phase simulations were running SLOW.

- Actually, I was looking for a way to efficiently solve the Stokes problem resulting from the semi-Lagrangian discretization of the macroscopic equations in a FE setting.

Solution of Stokes problem

Dimensionless form

Momentum and continuity equations

$$
\begin{cases}
Re\rho\frac{D\boldsymbol{u}}{Dt}-\boldsymbol{\nabla}\cdot\left[\eta\left(\boldsymbol{\nabla}\boldsymbol{u}+\left(\boldsymbol{\nabla}\boldsymbol{u}\right)^{T}\right)\right]+\boldsymbol{\nabla}p=-\rho\boldsymbol{e}_z\frac{Re}{Fr^2}+\frac{c}{De}\boldsymbol{\nabla}\cdot\boldsymbol{\tau}_p+\frac{Re}{We}\kappa\delta_{\Gamma}(\phi)\boldsymbol{n}, \\
\boldsymbol{\nabla}\cdot\boldsymbol{u}=0; \\
\boldsymbol{u}\left(\boldsymbol{x},0\right)=\boldsymbol{u}_0\left(\boldsymbol{x}\right)\quad\forall\boldsymbol{x}\in D, \\
\boldsymbol{u}\left(\boldsymbol{x},t\right)=\boldsymbol{0}\quad\text{on}\quad\delta D_{\text{no-slip}}\subset\delta D,\forall t\in(0,T), \\
\boldsymbol{u}\left(\boldsymbol{x},t\right)\cdot\boldsymbol{n}=0\quad\text{and}\quad\boldsymbol{n}\cdot\boldsymbol{\tau}_s\cdot\boldsymbol{t}=0\quad\text{on}\quad\delta D_{\text{free-slip}}=\delta D\ \delta D_{\text{no-slip}},\forall t\in(0,T).
\end{cases}
$$

Dimensionless groups

- Reynolds $Re=\frac{\rho_s UL}{\eta_s}$.
- Froude $Fr^2=\frac{U^2}{gL}$.
- $We=\frac{\rho_s U^2 L}{\sigma}$ (σ the surface tension coefficient).
- Deborah $De=\frac{\lambda U}{L}$ (λ the relaxation time).
- Concentration (of the non-Newtonian fluid) $c=\frac{\lambda nk_B\Theta}{\eta_s}$. [a]

[a] Note that: $c=\frac{1-\beta}{\beta}\left(\frac{b+5}{5}\right)$, with $\beta=\frac{\eta_s}{\eta_s+\eta_p^0}$.

Solution of Stokes problem

Dimensionless form

Momentum and continuity equations: time-space discretization

$$\begin{cases} \frac{3Re}{2\Delta t}\left(\rho^*\left(\phi_h^n\right)\boldsymbol{u}_h^n,\varphi_h\right)+\left(\eta^*\left(\phi_h^n\right)\nabla\boldsymbol{u}_h^n,\nabla\varphi_h\right)-\left(p_h^n,\nabla\cdot\varphi_h\right)=\frac{2Re}{\Delta t}\left(\rho^*\left(\phi_h^n\right)\bar{\boldsymbol{u}}_h^{n-1},\varphi_h\right)- \\ -\frac{Re}{2\Delta t}\left(\rho^*\left(\phi_h^n\right)\bar{\boldsymbol{u}}_h^{n-2},\varphi_h\right)-\frac{Re}{Fr^2}\left(\rho^*\left(\phi_h^n\right)\boldsymbol{e}_z,\varphi_h\right)+ \\ +\frac{c}{De}\left(\nabla\cdot\tau_{ph}^n,\varphi_h\right)+\frac{Re}{We}\left(\kappa_h^n\delta_{\Gamma_{h/2}}(\phi_h^n)\boldsymbol{n}_h^n,\varphi_h\right),\forall\varphi_h\in\boldsymbol{V}_{h0}; \\ \left(\nabla\cdot\boldsymbol{u}_h^n,q_h\right)=0,\ \forall q_h\in Q_h; \qquad \text{with}:(a,b)\equiv\int_D ab\,dx \end{cases}$$

Efficient solution of saddle-point problem

✓ Extremely ill-conditioned system (as $1/h\uparrow,\rho_1/\rho_2\uparrow,\mu_1/\mu_2\uparrow$):

$$\begin{pmatrix} A & B \\ B^T & 0 \end{pmatrix}\begin{pmatrix} \boldsymbol{U} \\ \mathrm{P} \end{pmatrix}=\begin{pmatrix} \boldsymbol{F} \\ \boldsymbol{0} \end{pmatrix},$$

✓ Some possibilities:

1. Uzawa-Preconditioned Conjugate Gradient [Dean and Glowinski 1993].
2. Iterated penalty [Gunzburger 1989].
3. Schur-based block-preconditioned PETSc FieldSplit [Balay et al. 2015].

Solution of Stokes problem

Alternatives

PROS:

1. Uzawa-PCG:
   - Easy to implement.
   - Good convergence for well-conditioned problems.
2. Iterated penalty:
   - Robust (insensitive to condition number).
   - Few (2,3) iterations.
3. PETSc FieldSplit:
   - Excellent convergence.
   - High versatility.

CONS:

1. Uzawa-PCG:
   - Very poor convergence (useless) for ill-conditioned problems.
2. Iterated penalty:
   - High memory requirements.
   - Non-zero sparsity of BB^T (slow, sparse matrix-matrix product)
   - Very ill-conditioned $(A + 1/\varepsilon BB^T)$ to update $\boldsymbol{U}^k$.
3. PETSc FieldSplit:
   - Implementation into existing code (PCFieldSplit interface).

Solution of Stokes problem

Alternatives

PROS:

1. Uzawa-PCG:
   - Easy to implement.
   - Good convergence for well-conditioned problems.
2. Iterated penalty:
   - Robust (insensitive to condition number).
   - Few (2,3) iterations.
3. PETSc FieldSplit:
   - Excellent convergence.
   - High versatility.

CONS:

1. Uzawa-PCG:
   - Very poor convergence (useless) for ill-conditioned problems.
2. Iterated penalty:
   - High memory requirements.
   - Non-zero sparsity of BB^T (slow, sparse matrix-matrix product)
   - Very ill-conditioned $(A + 1/\varepsilon BB^T)$ to update $\boldsymbol{U}^k$.
3. PETSc FieldSplit:
   - Implementation into existing code (PCFieldSplit interface).

Efficient solution of saddle-point problem

Comparison of methods

Table: Comparison of the average number of iterations, time and memory requirements for the three proposed methods of solving the saddle-point problem in a rising bubble simulation.

$\frac{\rho_1}{\rho_2}$	$1/h$	Uzawa			Iterated penalty			Field-split		
		Iters	Time(s)	Mem(%)	Iters	Time(s)	Mem(%)	Iters	Time(s)	Mem(%)
10										
	40	54.7	0.6	0.8	2	0.25	1.1	8	0.22	0.9
	80	57	2.8	2.4	2	1.5	4.1	10	0.93	2.6
	160	54	11.9	8.8	2	9.9	17.6	12	4.2	10.5
	320	52.3	52.5	32.1	2	71.3	78.2	15	21.6	41.5
10^2										
	40	141	1.4	0.8	2	0.25	1.0	9	0.24	0.9
	80	151.7	6.7	2.3	2	1.5	4.1	11	0.97	2.7
	160	157	30.5	8.6	2	9.9	17.5	13	4.5	10.2
	320	156	130.9	35.1	2	71.4	80.5	15.3	21.9	42.1
10^3										
	40	203.7	1.98	0.8	2	0.25	1.1	10.7	0.27	0.8
	80	242.2	10.5	2.3	2	1.5	4.1	12	1.0	2.6
	160	302.3	56.8	8.8	2	9.9	17.6	14.7	4.9	10.1
	320	333	266.1	35.2	2	73.8	80.9	18	24.4	41.8

Simulations run in a 4-core, i7-3770k@4.2 GHz, with 32 GB DDR3-RAM@1666 MHz.

Efficient solution of saddle-point problem

PCFieldSplit: auxiliary structure

Structure defined for Stokes-like problem

```
/* Structure with all the information required to solve a saddle-point problem */
typedef struct
{
Mat K; /*MatNest matrix with 4 sub-matrices: K = [K00 K01; K10 K11]*/
Mat K_ij[4]; /*each of the four sub-matrices of block matrix "K"*/
Mat Pmat; /*MatNest precond-matrix: Pmat = [Pmat00 Pmat01; Pmat10 Pmat11]; */
Mat Pmat_ij[4]; /*each of the four sub-matrices of precond-matrix "Pmat"*/
Vec x; /*solution of system*/
Vec b; /*right hand side vector*/
KSP ksp; /*solver context*/
MatNullSpace nullsp; /*Nullspace for pressure (in Stokes solved by PCFieldSplit)*/
PC pc; /*preconditioner context*/
IS isg[2]; /* index sets (rows) of splits "0" and "1" (e.g. velocity, pressure)*/
}
PETSC_Saddle_point_system;
```

Efficient solution of saddle-point problem

PCFieldSplit: useful functions

```
/*Construct index sets, vectors for Stokes System*/
ierr = fun_petsc_construct_index_sets_PETSC_Stokes_system ( &Stokes_Problem_system);
 /*Construct KSP and PC contexts for Stokes System*/
ierr = fun_petsc_construct_KSP_and_PC_PETSC_Stokes_system ( &Stokes_Problem_system);
/*Construct RHS side for Stokes System*/
ierr = fun_petsc_construct_rhs_and_solution_vectors_PETSC_Stokes_system ( &Stokes_Problem_system);
...
/*Perform splitting of fields for preconditioner (after the type of PC has been setup*/
ierr = fun_petsc_construct_field_splits_PC_PETSC_Stokes_system ( &Stokes_Problem_system); CHKERRQ(ierr);
```

Example function to build block matrix $K = [AB; B^T 0]$

```
PetscErrorCode fun_petsc_construct_block_matrix_PETSC_Stokes_system (...)
{
  ...
  /* Build first sub-matrix "K00" */
  ierr = MatDuplicate(K00,MAT_COPY_VALUES,& (Stokes_Problem_system->K_ij[0]));CHKERRQ(ierr);
  ierr = MatAssemblyBegin(Stokes_Problem_system->K_ij[0],MAT_FINAL_ASSEMBLY); CHKERRQ(ierr);
  ierr = MatAssemblyEnd(Stokes_Problem_system->K_ij[0],MAT_FINAL_ASSEMBLY); CHKERRQ(ierr);
  ...
  /* Build complete, block matrix "K" from sub-matrices "K_ij" as a nested matrix */
  ierr = MatCreateNest(PETSC_COMM_WORLD, 2, NULL, 2, NULL, \
                Stokes_Problem_system->K_ij, & (Stokes_Problem_system->K)); CHKERRQ(ierr);
  ierr = MatAssemblyBegin(Stokes_Problem_system->K,MAT_FINAL_ASSEMBLY); CHKERRQ(ierr);
  ierr = MatAssemblyEnd(Stokes_Problem_system->K,MAT_FINAL_ASSEMBLY); CHKERRQ(ierr);
  ...
}
```

Efficient solution with PCFieldSplit

Definition and useful options I

LDU factorization [Elman et al. 2008, *J. Comput. Phys.*] of block matrix:

$$\begin{pmatrix} A & B \\ B^T & 0 \end{pmatrix} = \begin{pmatrix} I & 0 \\ B^T A^{-1} & I \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} I & A^{-1}B \\ 0 & I \end{pmatrix}, \quad S \equiv -B^T A^{-1} B.$$

```
/* KSP00=preonly (cholesky,CHOLMOD); KSP11:preonly (lsc); ksp_lsc: cg (ml(+asm)). */
char  options_pc_stokes[] = "-stokes_ksp_rtol 5.e-9 -stokes_ksp_diagonal_scale \
-stokes_ksp_type fgmres -stokes_pc_type fieldsplit -stokes_pc_fieldsplit_type schur \
-stokes_pc_fieldsplit_schur_fact_type upper -stokes_pc_fieldsplit_detect_saddle_point
-stokes_fieldsplit_0_pc_type cholesky \
-stokes_fieldsplit_0_pc_factor_mat_solver_package cholmod \
-stokes_fieldsplit_0_ksp_type preonly -stokes_fieldsplit_1_pc_type lsc \
-stokes_fieldsplit_1_lsc_pc_type ml \
-stokes_fieldsplit_1_lsc_mg_coarse_pc_factor_shift_type NONZERO \
-stokes_fieldsplit_1_lsc_mg_levels_1_pc_type asm \
-stokes_fieldsplit_1_lsc_mg_levels_2_pc_type asm \
-stokes_fieldsplit_1_lsc_mg_levels_3_pc_type asm \
-stokes_fieldsplit_1_lsc_mg_levels_4_pc_type asm \
-stokes_fieldsplit_1_lsc_mg_levels_5_pc_type asm \
-stokes_fieldsplit_1_lsc_ksp_max_it 3 -stokes_fieldsplit_1_lsc_ksp_type cg \
-stokes_fieldsplit_1_lsc_ksp_constant_null_space  \
-stokes_fieldsplit_1_ksp_type preonly";
```

Efficient solution with PCFieldSplit

Definition and useful options II

```
/* ...KSP11:preonly (lsc), ksp_lsc: preonly (lu,UMFPACK). */
char  options_pc_stokes[] = "-stokes_ksp_monitor_true_residual \
-stokes_ksp_final_residual \
-stokes_ksp_monitor -stokes_ksp_converged_reason -stokes_ksp_rtol 5.e-9 \
-stokes_ksp_view -stokes_ksp_type gcr \
-stokes_ksp_initial_guess_nonzero false \
-stokes_pc_type fieldsplit -stokes_pc_fieldsplit_schur_precondition self \
-stokes_pc_fieldsplit_type schur \
-stokes_pc_fieldsplit_schur_fact_type upper \
-stokes_pc_fieldsplit_detect_saddle_point \
-stokes_fieldsplit_0_pc_type cholesky \
-stokes_fieldsplit_0_pc_factor_mat_solver_package cholmod \
-stokes_fieldsplit_0_ksp_type preonly -stokes_fieldsplit_1_pc_type lsc \
-stokes_fieldsplit_1_lsc_pc_type lu \
-stokes_fieldsplit_1_lsc_ksp_type preonly \
-stokes_fieldsplit_1_lsc_pc_factor_mat_solver_package umfpack  \
-stokes_fieldsplit_1_lsc_ksp_constant_null_space  \
-stokes_fieldsplit_1_ksp_type preonly";
```

INDUSTRIALES
ETSII | UPM

Efficient solution with PCFieldSplit

Definition and useful options III

```
/* ... KSP11:preonly (lsc); ksp_lsc: fgmres (asm). */
char  options_pc_stokes[] = "-stokes_ksp_converged_reason \
-stokes_ksp_rtol 5.e-9 -stokes_ksp_type gcr \
-stokes_pc_type fieldsplit -stokes_pc_fieldsplit_schur_precondition self \
-stokes_pc_fieldsplit_type schur \
-stokes_pc_fieldsplit_schur_fact_type upper \
-stokes_pc_fieldsplit_detect_saddle_point \
-stokes_fieldsplit_0_pc_type cholesky \
-stokes_fieldsplit_0_pc_factor_mat_solver_package cholmod \
-stokes_fieldsplit_0_ksp_type preonly \
-stokes_fieldsplit_1_pc_type lsc -stokes_fieldsplit_1_lsc_pc_type asm \
-stokes_fieldsplit_1_lsc_ksp_type fgmres  \
-stokes_fieldsplit_1_lsc_ksp_constant_null_space \
-stokes_fieldsplit_1_ksp_type preonly -stokes_fieldsplit_1_ksp_monitor";
```

▸ Conf.file ▸ Makefile ▸ Eclipse

INDUSTRIALES
ETSII | UPM

Efficient solution of linear systems with PETSc

Some Remarks

Improvement over Uzawa-PCG without PETSc

1. Moderate density and viscosity ratios (10). Grid size $1/h = 160$:

Efficient solution of linear systems with PETSc

Some Remarks

Improvement over Uzawa-PCG without PETSc

1. Moderate density and viscosity ratios (10). Grid size $1/h = 160$:
   - From 23 hours to 1.5 hours ($\approx 15\times$).

Efficient solution of linear systems with PETSc

Some Remarks

Improvement over Uzawa-PCG without PETSc

1. Moderate density and viscosity ratios (10). Grid size $1/h = 160$:
   - From 23 hours to 1.5 hours ($\approx 15\times$).
2. High density and viscosity ratios (10^3). Grid size $1/h = 160$:

Efficient solution of linear systems with PETSc

Some Remarks

Improvement over Uzawa-PCG without PETSc

1. Moderate density and viscosity ratios (10). Grid size $1/h = 160$:
   - From 23 hours to 1.5 hours ($\approx 15\times$).
2. High density and viscosity ratios (10^3). Grid size $1/h = 160$:
   - From 8.3 days to 2.5 hours ($\approx 80\times$).

Efficient solution of linear systems with PETSc

Some Remarks

Improvement over Uzawa-PCG without PETSc

1. Moderate density and viscosity ratios (10). Grid size $1/h = 160$:
   - From 23 hours to 1.5 hours ($\approx 15\times$).
2. High density and viscosity ratios (10^3). Grid size $1/h = 160$:
   - From 8.3 days to 2.5 hours ($\approx 80\times$).
3. Smaller grid size was unfeasible. ▸ More data

Efficient solution of linear systems with PETSc

Some Remarks

Improvement over Uzawa-PCG without PETSc

1. Moderate density and viscosity ratios (10). Grid size $1/h = 160$:
   - From 23 hours to 1.5 hours ($\approx 15\times$).
2. High density and viscosity ratios (10^3). Grid size $1/h = 160$:
   - From 8.3 days to 2.5 hours ($\approx 80\times$).
3. Smaller grid size was unfeasible. ▸ More data

PETSc also used in:

- Filtering: $(M + \varepsilon R)\hat{\phi} = M\phi$.
- Computation of $\kappa = \nabla \boldsymbol{u}$.
- Solution of saddle-point problem for CSRBFs: S is $l \times l, l = \{6, 10\}$ in $\{2D, 3D\}$.

$$\begin{pmatrix} A & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda} \\ \boldsymbol{c} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{0} \end{pmatrix} \Leftrightarrow \begin{cases} (-S)\,\boldsymbol{c} = (P^T A^{-1})\,\boldsymbol{f}; \\ A\boldsymbol{\lambda} = \boldsymbol{f} - P\boldsymbol{c}. \end{cases}$$

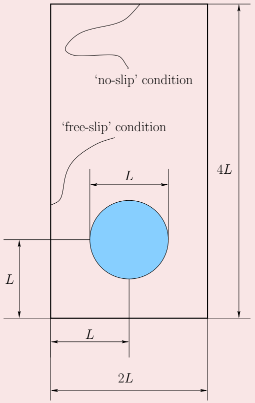
Complex Newtonian multi-phase flows

Benchmark problem: rising bubble

Notes on multi-phase complex flows

1. Demanding even in bi-dimensional problems.

Geometry



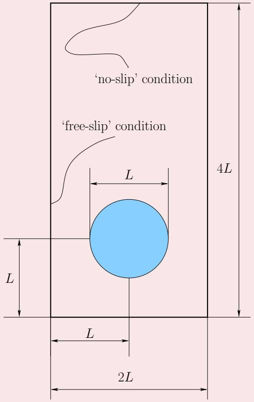
Complex Newtonian multi-phase flows

Benchmark problem: rising bubble

Notes on multi-phase complex flows

1. Demanding even in bi-dimensional problems.
2. No analytical solutions possible.

Geometry



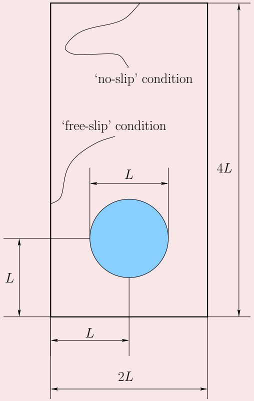
Complex Newtonian multi-phase flows

Benchmark problem: rising bubble

Notes on multi-phase complex flows

1. Demanding even in bi-dimensional problems.
2. No analytical solutions possible.
3. Numerical methods provide (slightly) different solutions at different regimes (⇒) Measurement of relevant magnitudes

Geometry



Complex Newtonian multi-phase flows

Benchmark problem: rising bubble

Parameters

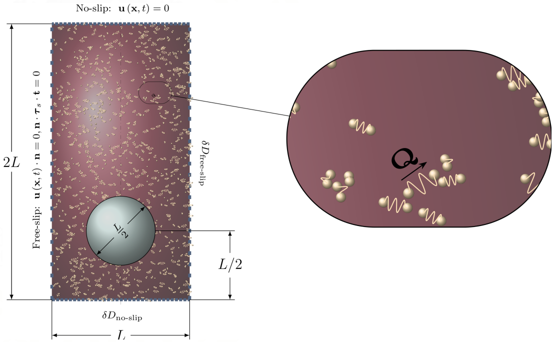
- $h_M = 1/50, Re = 35, Fr = 1.$
- Left: $We = 10, \frac{\rho_2}{\rho_1} = 10^{-1}, \frac{\mu_2}{\mu_1} = 10^{-1}.$
- Right: $We = 125, \frac{\rho_2}{\rho_1} = 10^{-3}, \frac{\mu_2}{\mu_1} = 10^{-2}.$

Non-Newtonian multi-phase flows

Buoyancy-driven bubbles

FENE fluid

1. Non-Newtonian fluid in the outer region.
2. Transition from oblate to prolate state. Eventual cusp-like tail.[9]



No-slip: $\mathbf{u}(\mathbf{x},t)=0$
Free-slip: $\mathbf{u}(\mathbf{x},t)\cdot\mathbf{n}=0, \mathbf{n}\cdot\boldsymbol{\tau}_s\cdot\mathbf{t}=0$
$2L$
$\delta D_{\text{free-slip}}$
$L/2$
$L/2$
$\delta D_{\text{no-slip}}$
L

SLEIPNNIR method

- '**S**emi-**L**agrangian **E**nsemble **I**mplementation of **P**article level sets for **N**ewtonian and non-**N**ewtonian **I**nterfacial **R**heology'.[10]

[9] Prieto 2015, *J. Non-Newtonian Fluid Mech.*

[10] Prieto 2016b, *Comput. Methods Appl. Mech. Engrg.*

Benchmark problem - shapes of FENE rising bubble

Cases: $c = 1, De = 1$ (blue); $c = 3, De = 1$ (green); $c = 5, De = 3$ (yellow); $c = 9, De = 5$ (red)

$$\frac{\rho_2}{\rho_1} = 10^{-1} = \frac{\mu_2}{\mu_1}.$$

$$Re = 35,\ We = 10,\ Fr = 1$$

$$\frac{\rho_2}{\rho_1} = 10^{-3};\ \frac{\mu_2}{\mu_1} = 10^{-2}.$$

$$Re = 35,\ We = 125,\ Fr = 1.$$

Benchmark problem - streamlines of FENE rising bubble

Low density and viscosity ratios: $\frac{\rho_2}{\rho_1} = 10^{-1}$; $\frac{\mu_2}{\mu_1} = 10^{-1}$

$c = 1, De = 1$ $c = 3, De = 1$ $c = 5, De = 3$ $c = 9, De = 5$

Benchmark problem - streamlines of FENE rising bubble

High density and viscosity ratios: $\frac{\rho_2}{\rho_1} = 10^{-3}$; $\frac{\mu_2}{\mu_1} = 10^{-2}$

$c = 1, De = 1$ $c = 3, De = 1$ $c = 5, De = 3$ $c = 9, De = 5$

Benchmark problem - isocontours of $\boldsymbol{\tau}_p$

Isocontours of shear-component τ_{p12} and normal stress difference $\tau_{p11} - \boldsymbol{\tau}_{p22}$

τ_{p12}: $c = 5, De = 3$

τ_{p12}: $c = 9, De = 5$

$\tau_{p11} - \tau_{p22}$: $c = 5, De = 3$

$\tau_{p11} - \tau_{p22}$: $c = 9, De = 5$

Conclusions

- A multiscale semi-Lagrangian, particle level-set, micro-macro method for the simulation of Newtonian and non-Newtonian free surface flows has been developed.

Conclusions

- A multiscale semi-Lagrangian, particle level-set, micro-macro method for the simulation of Newtonian and non-Newtonian free surface flows has been developed.
- Multi-phase Newtonian and non-Newtonian flows can be investigated. Experimentally observed effects are reproduced.

Conclusions

- A multiscale semi-Lagrangian, particle level-set, micro-macro method for the simulation of Newtonian and non-Newtonian free surface flows has been developed.
- Multi-phase Newtonian and non-Newtonian flows can be investigated. Experimentally observed effects are reproduced.
- PETSc makes it possible: the code runs efficiently in a commodity PC.

Future work

1. Addition of isotropic and anisotropic adaptivity. ▸ Mesh adapt
2. Continuation LC investigations: Doi-Hess model, defects, biological sensors. ▸ Current LC bubble...
3. Extension to three-dimensional problems.
4. MPI Parallelization of the code for distributed-memory machines...
5. And/or implementation in many core (e.g. Intel® “Knights Landing”) architectures?

THANK YOU

Main features III:

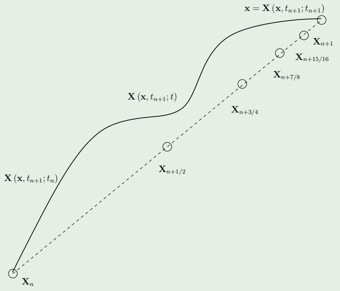
Computation of trajectories and feet of characteristics

Time-adaptive integration

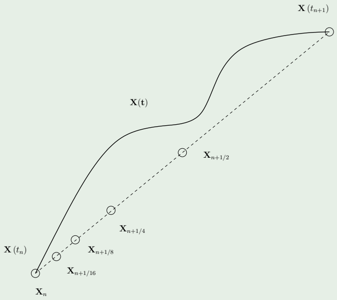
Applied to trajectories and feet of characteristic curves.

Basic scheme

- Adaptive version of the process:



$\mathbf{x} = \mathbf{X}(\mathbf{x}, t_{n+1}; t_{n+1})$
$\mathbf{X}_{n+1}$
$\mathbf{X}_{n+15/16}$
$\mathbf{X}_{n+7/8}$
$\mathbf{X}(\mathbf{x}, t_{n+1}; t)$
$\mathbf{X}_{n+3/4}$
$\mathbf{X}_{n+1/2}$
$\mathbf{X}(\mathbf{x}, t_{n+1}; t_n)$
$\mathbf{X}_n$



- Trajectories $\boldsymbol{X}(\boldsymbol{x}_i, t_{n+1}; t)$ do not leave the domain.

Main features III:

Computation of trajectories and feet of characteristics (ii)

Time-adaptive integration

Applied to trajectories and feet of characteristic curves.

Basic scheme

- Mid-point rule for Eq.6:

$$\boldsymbol{X}\left(\boldsymbol{x}, t_{n+1}; t_l\right) = \boldsymbol{x}_i - \Delta t \boldsymbol{v}_h\left(\boldsymbol{X}\left(\boldsymbol{x}_i, t_{n+1}; t_l + \vartheta_{nl}\right), t_l + \vartheta_{nl}\right),$$

  with $\vartheta_{nl} \equiv \frac{(n+1-l)\Delta t}{2}$.

- Extrapolation formula for velocity:

$$\boldsymbol{v}_h\left(\cdot, t_l + \vartheta_{nl}\right) = \begin{cases} \frac{3}{2}\boldsymbol{v}_h(\cdot, t_n) - \frac{1}{2}\boldsymbol{v}_h(\cdot, t_{n-1}), \ l = n, \\ \boldsymbol{v}_h(\cdot, t_n), \ l = n-1. \end{cases}$$

Main features III:

Computation of trajectories and feet of characteristics (iii)

Basic scheme (cont.)

- Fixed point iterative algorithm for:

$$\boldsymbol{\varepsilon}(\boldsymbol{x}_i, t_{n+1}; t_l) = \Delta t \boldsymbol{v}_h \left(\boldsymbol{x}_i - \frac{1}{2}\boldsymbol{\varepsilon}(\boldsymbol{x}_i, t_{n+1}; t_l), t_l + \vartheta_{nl} \right).$$

with $\boldsymbol{\varepsilon}(\boldsymbol{x}_i, t_{n+1}; t_l) \equiv \boldsymbol{x}_i - \boldsymbol{X}(\boldsymbol{x}_i, t_{n+1}; t_l)$.

INDUSTRIALES
ETSII | UPM

Eikonal redistancing

Reinitialization algorithm and convergence rates

Algorithm 1: Reinitialization scheme by time-dependent eikonal equation.

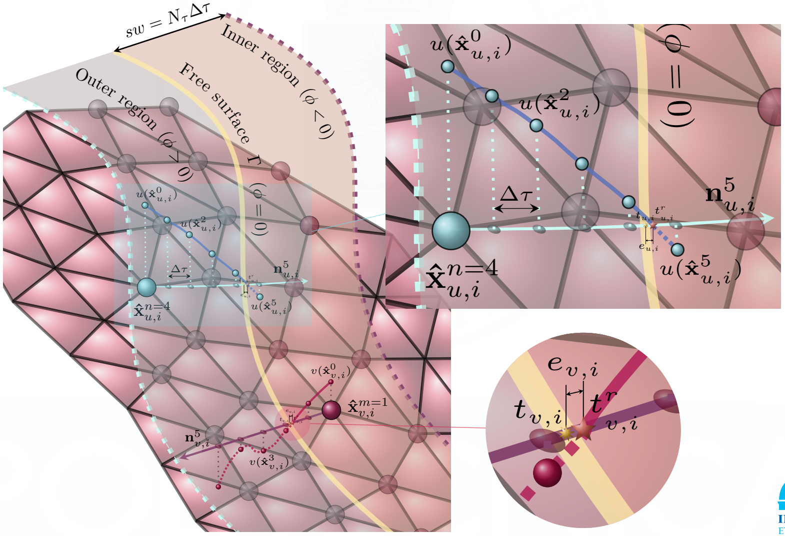
Data: Scalar fields u, v; level set function ϕ_0; interface Γ; subdomains $D_1\,(\boldsymbol{x} \in D, \phi_0(\boldsymbol{x}) > 0)$ and $D_2\,(\boldsymbol{x} \in D, \phi_0(\boldsymbol{x}) < 0)$; time step size $\Delta\tau$; number of time steps N_τ; points $\left\{\boldsymbol{x}_{\Sigma_\Gamma}\right\}$ inside band Σ_Γ^{sw} of semi-width $sw = T_{end} = N_\tau \Delta\tau$ around Γ.

Result: Signed distance function $d(\boldsymbol{x})$, $\forall \boldsymbol{x} \in \left\{\boldsymbol{x}_{\Sigma_\Gamma}\right\}$.

```
1  assign $u^0(\boldsymbol{x}) = \phi_0, v^0(\boldsymbol{x}) = -\phi_0$.
2  for time step $n = 1 : N_\tau$ do
3      build normals $\boldsymbol{n}_u^n, \boldsymbol{n}_v^n$ of $u^n, v^n$, for $\boldsymbol{x} \in \Sigma_\Gamma^{sw}$, by the second-order method.
4      compute feet of characteristic curves $\boldsymbol{X}_{\boldsymbol{n}_u}^n, \boldsymbol{X}_{\boldsymbol{n}_v}^n$ of $\frac{D_{\boldsymbol{n}_u}^n}{Dt}, \frac{D_{\boldsymbol{n}_v}^n}{Dt}$.
5      set $u^{n*}, v^{n*}$ as the quadratically interpolated solution of $u^n, v^n$ at $\boldsymbol{X}_{\boldsymbol{n}_u}^n, \boldsymbol{X}_{\boldsymbol{n}_v}^n$.
6      update $u^{n+1} = u^{n*}, v^{n+1} = v^{n*}$.
7      identify and store mesh points $\hat{\boldsymbol{x}}_{u,i}^n$, $\hat{\boldsymbol{x}}_{v,i}^n$ with change of sign in $\left[u^n(\boldsymbol{x}), u^{n+1}(\boldsymbol{x})\right)$, $\left[v^n(\boldsymbol{x}), v^{n+1}(\boldsymbol{x})\right)$,
       respectively.
8  end
9  for mesh points with change of sign $\hat{\boldsymbol{x}}_{u,i}^n$, $\hat{\boldsymbol{x}}_{v,i}^n$ do
10     build Akima spline for set of values $\{n\Delta\tau, u^n(\boldsymbol{x})\}$, $\{n\Delta\tau, v^n(\boldsymbol{x})\}$.
11     compute $\boldsymbol{x}_u^r$, $\boldsymbol{x}_v^r$ roots of Akima splines by iterative method, and
12     set $d(\hat{\boldsymbol{x}}_{u,i}^n) = \boldsymbol{x}_u^r$, $d(\hat{\boldsymbol{x}}_{v,i}^n) = -\boldsymbol{x}_v^r$.
13 end
```

Eikonal redistancing

Reinitialization algorithm and convergence rates



$sw = N_\tau \Delta\tau$
Inner region ($\phi < 0$)
Free surface Γ ($\phi = 0$)
Outer region ($\phi > 0$)
$u(\hat{\mathbf{x}}^0_{u,i})$
$u(\hat{\mathbf{x}}^2_{u,i})$
$\Delta\tau$
$\mathbf{n}^5_{u,i}$
$t_{u,i}$
$t^r_{u,i}$
$e_{u,i}$
$u(\hat{\mathbf{x}}^5_{u,i})$
$\hat{\mathbf{x}}^{n=4}_{u,i}$
$(\phi = 0)$
$v(\hat{\mathbf{x}}^0_{v,i})$
$\hat{\mathbf{x}}^{m=1}_{v,i}$
$\mathbf{n}^5_{v,i}$
$v(\hat{\mathbf{x}}^3_{v,i})$
$e_{v,i}$
$t_{v,i}$
$t^r_{v,i}$

Eikonal redistancing

Reinitialization algorithm and convergence rates

Table: Convergence rates p when using Algorithm 1 for the reinitialization of a level set function $\phi_0(x, y) = \exp(x + y)\left(x^2 + y^2 - \frac{1}{4}\right)$ and different mesh refinement, interpolant and approximation type.

$1/h$	p_l^{linear}	p_q^{linear}	p_l^{cubic}	p_q^{cubic}	p_l^{Akima}	p_q^{Akima}
100	1.00	1.82	1.01	1.00	1.01	1.78
200	1.05	1.86	1.04	1.00	1.04	1.95
400	1.00	1.92	1.00	1.00	1.00	1.97
800	1.01	1.99	1.01	1.00	1.01	1.97

Table: Convergence rates p when using Algorithm 1 for the reinitialization of a level set function $\phi_0(x, y) = [\sin(4\pi x)\sin(4\pi y) + 2]\left[\exp\left(x^2 + y^2 - \frac{1}{4}\right) - 1\right]$ in a band Σ_Γ^{sw} of semi-width $sw = 0.301636$.

$1/h$	p_l^{linear}	p_q^{linear}	p_l^{cubic}	p_q^{cubic}	p_l^{Akima}	p_q^{Akima}
100	0.56	1.78	0.54	1.29	0.54	1.69
200	0.79	1.43	0.79	1.03	0.79	1.44
400	0.87	1.64	0.87	1.02	0.87	1.70
800	0.91	1.89	0.91	1.00	0.91	1.89

Surface tension implementation

Laplace-Beltrami operator

Properties and notation

- Identity function: $\boldsymbol{id}$; Laplace-Beltrami operator (or surface Laplacian): $\underline{\boldsymbol{\Delta}}$; surface gradient: $\underline{\boldsymbol{\nabla}}$; tangential projection tensor: $\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{n}_h \otimes \boldsymbol{n}_h$.
- Equalities: $\int_\Gamma \kappa \boldsymbol{n} \cdot \varphi \, d\Gamma = \int_\Gamma \underline{\boldsymbol{\Delta}} \boldsymbol{id}_\Gamma \cdot \varphi \, d\Gamma = -\int_\Gamma \underline{\boldsymbol{\nabla}} \boldsymbol{id}_\Gamma \cdot \underline{\boldsymbol{\nabla}} \varphi \, d\Gamma = -\int_\Gamma \boldsymbol{P} : \boldsymbol{\nabla} \varphi \, d\Gamma$.
- Explicit scheme:

$$\boldsymbol{f}^{*n}_{\sigma h} = -\frac{Re}{We} \sum_{i=0}^{N_{\Gamma_{h/2}}} \int_{\Gamma^i_{h/2}} \left(\boldsymbol{I} - \overset{\Gamma}{\boldsymbol{n}}{}^n_i \otimes \overset{\Gamma}{\boldsymbol{n}}{}^n_i \right) : \boldsymbol{\nabla} \varphi_h \, d\Gamma^i_{h/2}.$$

- Semi-implicit scheme:

$$\boldsymbol{f}^{*n}_{\sigma h} = -\frac{Re}{We} \sum_{i=0}^{N_{\Gamma_{h/2}}} \int_{\Gamma^i_{h/2}} \left[\left(\boldsymbol{I} - \overset{\Gamma}{\boldsymbol{n}}{}^n_i \otimes \overset{\Gamma}{\boldsymbol{n}}{}^n_i \right) + \Delta t \boldsymbol{\nabla} \boldsymbol{u}^{n+1} \cdot \left(\boldsymbol{I} - \overset{\Gamma}{\boldsymbol{n}}{}^n_i \otimes \overset{\Gamma}{\boldsymbol{n}}{}^n_i \right) \right] : \boldsymbol{\nabla} \varphi_h \, d\Gamma^i_{h/2}.$$

INDUSTRIALES ETSII | UPM

Solution of linear systems with PETSc

Some additional data

"Large" simulation on Xeon E5-2670 v3 (12 core, 30M Cache, 2.30 GHz), 128 GB RAM. $1/h = 640$, $\rho_1/\rho_2 = 10^3$.

```
...
  Mat Object:              (stokes_fieldsplit_0_)             1 MPI processes
    type: seqaij
    rows=6561282, cols=6561282
    total: nonzeros=1.50794e+08, allocated nonzeros=1.50794e+08
...
  Mat Object:   1 MPI processes
    type: nest
    rows=7382403, cols=7382403
      Matrix object:
        type=nest, rows=2, cols=2
        MatNest structure:
        (0,0) : type=seqaij, rows=6561282, cols=6561282
        (0,1) : type=seqaij, rows=6561282, cols=821121
        (1,0) : type=seqaij, rows=821121, cols=6561282
        (1,1) : type=seqsbaij, rows=821121, cols=821121
```

KSP	PC (K00,K11)	Iters (1st time step)	Memory (GB)	Time (s) (1st time step)
gcr (30 rest)	(CHOLMOD,UMFPACK)	50	59.1	233.5
fgmres (200 rest)	(CHOLMOD,UMFPACK)	50	59.6	252.5
gcr (15 rest)	(CHOLMOD,UMFPACK)	53	57.5	238.6
gcr (15 rest)	cg(hypre), gcr(hypre))	59	53	1797.5

Algorithm for CSRBF implementation

Computation of τ_p

Algorithm 2: Solution procedure for the reconstruction of $\boldsymbol{\tau}_p$ by CSRBFs.

Data: Number of right-hand sides N_f; system matrices A, P of problem (24); matrix of right-hand sides $\tilde{\boldsymbol{f}} \equiv \left(\boldsymbol{f}_1 | \dots | \boldsymbol{f}_{N_f}\right)$.

Result: Solution matrices $\tilde{\boldsymbol{\lambda}} \equiv \left(\boldsymbol{\lambda}_1 | \dots | \boldsymbol{\lambda}_{N_f}\right)$ and $\tilde{\boldsymbol{c}} \equiv \left(\boldsymbol{c}_1 | \dots | \boldsymbol{c}_{N_f}\right)$.

```
1 solve $A\tilde{X} = \tilde{\boldsymbol{f}}$, using CHOLMOD;
2 compute $\tilde{Y} = P^T \tilde{X}$, using cblas_dgemm;
3 solve $AX = P$, using CHOLMOD;
4 compute $Y = P^T X$, using cblas\_dgemm;
5 solve $Y\tilde{\boldsymbol{c}} = \tilde{Y}$, using LAPACKE\_dgesv;
6 compute $\hat{\boldsymbol{f}} = \tilde{\boldsymbol{f}} - P\tilde{\boldsymbol{c}}$, using cblas\_dgemm;
7 solve $A\tilde{\boldsymbol{\lambda}} = \hat{\boldsymbol{f}}$, using CHOLMOD;
```

$$\begin{pmatrix} A & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda} \\ \boldsymbol{c} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{0} \end{pmatrix}, \text{ with } \begin{cases} A \equiv \{a_{ij}\} = \hat{\phi}_\chi\left(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|\right), 1 \le i, j, \le N_{ens}; \\ P \equiv \{p_{ij}\} = \hat{p}_j\left(\boldsymbol{x}_i\right), 1 \le i \le N_{ens}, 1 \le j \le l; \\ \boldsymbol{\lambda} \equiv \{\lambda_i\}, \boldsymbol{f} \equiv \{f_i\}, 1 \le i \le N_{ens}; \boldsymbol{c} \equiv \{c_i\}, 1 \le i \le l. \end{cases}$$

PETSc Additional material I

Configuration

Example conf. for `nauglamir_current_petsc_optimized_static_OpenBLAS` in PETSc 3.6.3.

```
./configure -with-blas-lib=[libopenblas.so,libpthread.so]
 ↪    -with-lapack-lib=libopenblas.so -with-cc=/usr/local/bin/gcc
 ↪    -with-cox-dialect=C++11 -with-fc=/usr/local/bin/gfortran COPTFLAGS='-O3
 ↪    -march=native -mtune=native' CXXOPTFLAGS='-O3 -march=native -mtune=native'
 ↪    FOPTFLAGS='-O3 -march=native -mtune=native'
-with-suitesparse -with-suitesparse-dir=/usr/local -download-elemental
 ↪    -download-superlu -download-superlu-mt -download-metis   -download-ptscotch
 ↪    -download-blacs -download-parmetis=1 -download-ml -download-fiat
 ↪    -download-triangle -download-generator -download-scientificpython
 ↪    -download-mpich -known-mpi-shared-libraries  -with-shared-libraries=0
 ↪    -download-scalapack     -download-mumps -with-openmp -with-pthreadclasses
 ↪    -download-superlu_dist -download-hypre  -download-pastix -with-debugging=0
```

PETSc Additional material II

Makefile

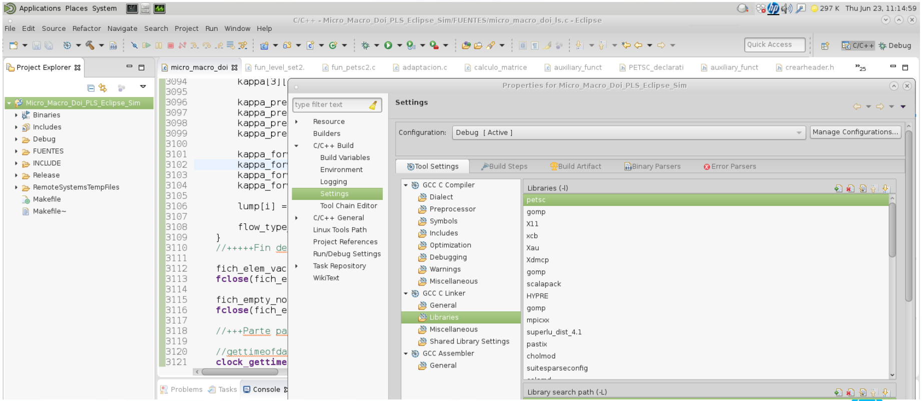
Makefile for nauglamir_current_petsc_optimized_static_OpenBLAS in PETSc 3.6.3.

```
COMPILER   = gcc -fopenmp  -Wall -O3 -std=gnu99 -march=native
  ↪    -I/opt/PETSc_library/petsc/nauglamir_current_petsc_optimized_static_OpenBLAS/include
  ↪    -I/opt/PETSc_library/petsc/include

LIBS       = -static -L/opt/PETSc_library/petsc/nauglamir_current_petsc_optimized_static_OpenBLAS/lib -lpetsc
  ↪    -lgomp -lX11 -lxcb -lXau -lXdmcp -lpthread -lcmumps -ldmumps -lmpi -lmpifort -lsmumps -lzmumps
  ↪    -lmumps_common -lmpifort -lpord -lscalapack -lHYPRE -lstdc++ -lsuperlu_dist_4.1 -lsuperlu_4.3 -lpastix
  ↪    -lml -lstdc++ -lumfpack -lEl -lmpi -lpmrrr -lopenblas -lpthread -lcholmod -lcamd -lccolamd -lcolamd
  ↪    -lamd -lklu -lbtf -lsuitesparseconfig  -lumfpack -lamd   -ltriangle -lptesmumps -lscotch -lptscotch
  ↪    -lptscotcherr  -lparmetis   -lmetis -lquadmath -lz -ldl -lmpi -ldl -lgsl -lgslcblas -lopenblas
  ↪    -lpthread\
-lm -lgfortran -lquadmath -lrt -lpthread -lstdc++

EXECUTABLE = mm_test
OBJECT     = micro_macro_doi_ls.o aritmeticas.o calculo_matrices.o calculo_matrices2.o caracteristicas.o
  ↪    crear.o doi_hess_fun.o funciones.o fun_micro_macro.o fun_micro_macro2.o fun_micro_macro3.o
  ↪    fun_level_set.o fun_level_set2.o fun_level_set3.o fun_stokes.o manipular_malla.o rand_knuth.o
  ↪    rand_knuth_int.o resolver_sistemas.o fun_petsc.o fun_petsc2.o fun_petsc3.o kdtree.o
$(EXECUTABLE): $(OBJECT)
        $(COMPILER) -o $(EXECUTABLE) $(OBJECT) $(LIBS)

%.o:    %.c
        $(COMPILER) -o $*.o -c $*.c
```

PETSc -Additional material III

Eclipse

Eclipse configuration for `nauglamir_current_petsc_optimized_static_OpenBLAS` in PETSc 3.6.3.



C/C++ - Micro_Macro_Doi_PLS_Eclipse_Sim/FUENTES/micro_macro_doi_ls.c - Eclipse

Project Explorer: Micro_Macro_Doi_PLS_Eclipse_Sim — Binaries, Includes, Debug, FUENTES, INCLUDE, Release, RemoteSystemsTempFiles, Makefile, Makefile~

```
        kappa[3][
        kappa_pre
        kappa_pre
        kappa_pre
        kappa_pre

        kappa_for
        kappa_for
        kappa_for
        kappa_for

        lump[i] =

        flow_type
    }
    //+++++Fin de

    fich_elem_vac
    fclose(fich_e

    fich_empty_no
    fclose(fich_e

    //+++Parte pa

    //gettimeofda
    clock_gettime
```

Properties for Micro_Macro_Doi_PLS_Eclipse_Sim

Settings

Configuration: Debug [Active] — Manage Configurations...

Tool Settings | Build Steps | Build Artifact | Binary Parsers | Error Parsers

GCC C Linker → Libraries

Libraries (-l):

- petsc
- gomp
- X11
- xcb
- Xau
- Xdmcp
- gomp
- scalapack
- HYPRE
- gomp
- mpicxx
- superlu_dist_4.1
- pastix
- cholmod
- suitesparseconfig

Library search path (-L)

Imposed flows

Benchmark problems: slotted cylinder

Zalesak's slotted cylinder

- Benchmark problem for diffusive effects (corners, slot).
- Imposed velocity field: $\boldsymbol{v} \equiv (u, v) = (0.5 - y, x - 0.5)$.
- Mesh with $NE = 5248$ elements ($h_M \approx 10^{-2}$).

Imposed flows

Benchmark problems: single vortex

Single vortex flow

- Benchmark for stretching and formation of thin filaments.
- Particles do improve resolution, using a finer mesh.
- $\boldsymbol{v} = \left(-\sin^2(\pi x)\sin(2\pi y), \sin^2(\pi y)\sin(2\pi x)\right)$.

N^{mk}	$s/iter$
-	1.91
$1.5 \cdot 10^4$	2.07
$1.5 \cdot 10^5$	3.43
$1.5 \cdot 10^6$	10.20

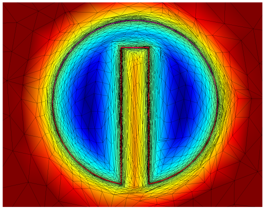
Example with $\Delta t = 5 \cdot 10^{-3}$, $N^{mk} = 1.5 \cdot 10^6$.

Adaptive Mesh Refinement

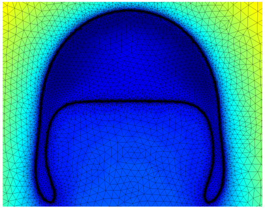
First steps

- ✓ Promising preliminary results (in collaboration: [Carpio and Prieto 2014, *Comput. Methods Appl. Mech. Engrg.* Carpio, Prieto, and Vera 2016, *J. Comput. Phys.*]).
- Structures must be created and destroyed at each time step (sparsity changes).
- Ideas for optimization? (storage, time).
- Ideas for parallel (an)isotropic mesh generator? (Bottleneck with serial BAMG / MMG3D; Gmsh?).
- Alternatives to CHOLMOD? (STRUMPACK?). (MUMPS,HYPRE,ML: slower).

Zalesak (anisotropic adaptation): mesh and shape after 50 loops



Surface tension-free rising bubble (isotropic adaptation)



Benchmark problem

LC bubble in Newtonian fluid: effects of protein on idofs

External agent (e.g. proteins) distorts configurations

- Protein interaction: changes $\boldsymbol{u} \Rightarrow \boldsymbol{\tau}_{LC} \Rightarrow \mathit{\Gamma}$ $(c = 50, Pe = 3)$.
- Left: director for $Re = 10$, $We = 35$. Right: birefringence for $Re = 35$, $We = 125$.

Allievi, A. and R. Bermejo (1997). “A Generalized Particle Search-Locate Algorithm for Arbitrary Grids”. In: *J. Comput. Phys.* 132, pp. 157–166.

— (2000). “Finite element modified method of characteristics for the Navier-Stokes equations”. In: *Int. J. Numer. Meth. Fluids* 32, pp. 439–464.

Ausas, R. F., F. S. Sousa, and G. C. Buscaglia (2010). “An improved finite element space for discontinuous pressures”. In: *Comput. Methods. Appl. Mech. Engrg.* 199, pp. 1019–1031.

Balay, Satish et al. (2015). *PETSc Web page*. http://www.mcs.anl.gov/petsc.

Bermejo, R. and J. L. Prieto (2013). “A Semi-Lagrangian Particle Level Set Finite Element Method for Interface Problems”. In: *SIAM J. Sci. Comput.* 35.4, A1815–A1846.

Carpio, J. and J. L. Prieto (2014). “An anisotropic, fully adaptive algorithm for the solution of convection dominated equations with semi-Lagrangian schemes”. In: *Comput. Methods Appl. Mech. Engrg.* 273, pp. 77–99.

Carpio, J., J. L. Prieto, and M. Vera (2016). “A local anisotropic adaptive algorithm for the solution of low-Mach transient combustion problems”. In: *J. Comput. Phys.* 306, pp. 19–42.

Cheng, L. T. and Y. H. Tsai (2008). “Redistancing by flow of time dependent eikonal equation”. In: *J. Comput. Phys.* 227, pp. 4002–4017.

Dean, E. J. and R. Glowinski (1993). “On some finite element methods for the numerical simulation of incompressible viscous flow”. In: *Incompressible Computational Fluid Dynamics*. Ed. by M. D. Gunzburger and R. A. Nicolaides. New York: Cambridge University Press, pp. 109–150.

Elman, H. et al. (2008). “A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier-Stokes equations”. In: *J. Comput. Phys.* 227, pp. 1790–1808.

Enright, D. et al. (2002). “A Hybrid Particle Level Set Method for Improved Interface Capturing”. In: *J. Comput. Phys.* 183.1, pp. 83–116.

Gunzburger, Max D. (1989). *Finite Element Methods for Viscous Incompressible Flows.* Academic Press. isbn: 978-0-12-307350-1.

Öttinger, H. C. (1996). *Stochastic Processes in Polymeric Fluids: Tools and Examples for Developing Simulation Algorithms.* Berlin: Springer.

Prieto, J. L. (2015). “Stochastic particle level set simulations of buoyancy-driven droplets in non-Newtonian fluids”. In: *J. Non-Newtonian Fluid Mech.* 226, pp. 16–31.

— (2016a). “An RBF-reconstructed, polymer stress tensor for stochastic, particle-based simulations of non-Newtonian, multiphase flows”. In: *J. Non-Newtonian Fluid Mech.* 227, pp. 90–99.

— (2016b). “SLEIPNNIR: A multiscale, particle level set method for Newtonian and non-Newtonian interface flows”. In: *Comput. Methods Appl. Mech. Engrg.* 307, pp. 164–192.

Prieto, J. L., R. Bermejo, and M. Laso (2010). “A semi-Lagrangian micro-macro method for viscoelastic flow calculations”. In: *J. Non-Newtonian Fluid Mech.* 165, pp. 120–135.