

>>> Level-set frameworks

Name: Thibaut METIVET

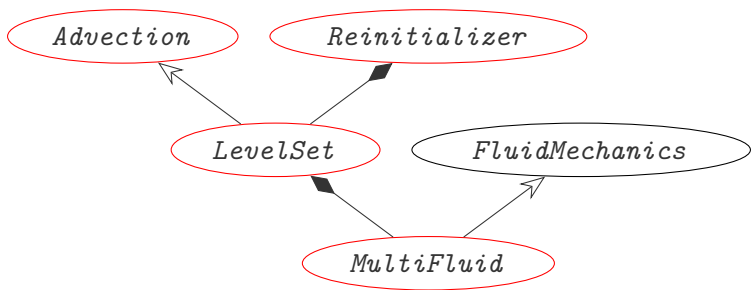
Date: 14/09/2017

Advection framework

Levelset framework

Multifluid framework

>>> Frameworks



Advection framework

>>> Mathematical problem

$$\partial_t \phi + \mathbf{u} \cdot \nabla \phi - \nabla \cdot (D \nabla \phi) + R \phi = 0$$

\mathbf{u} : advection velocity

D : diffusion coefficient

R : reaction coefficient

Variational formulation

$$\forall \psi \in H^1(\Omega),$$

$$\int_{\Omega} (\partial_t \phi + \mathbf{u} \cdot \nabla \phi) \psi + \int_{\Omega} D \nabla \phi \cdot \nabla \psi + \int_{\Omega} R \phi \psi = 0$$

>>> Discretization

Time

Backward differentiation formula scheme

$$\partial_t \phi \rightarrow \sum_i^n \frac{\alpha_i}{\delta t} \phi^{n+1-i}$$

Space

Lagrange P_n finite-elements

→ need stabilization (GLS, SUPG, CIP)

Levelset framework

>>> Level-set

$\phi(\mathbf{x})$ signed distance function to interface: $|\nabla\phi| = 1$

Evolution

Pure advection: $\partial_t\phi + \mathbf{u} \cdot \nabla\phi = 0$

Toolbox

feelpplib_toolbox_levelset_[2d/3d]

Initial value

Only $\phi = 0$ iso-level relevant e.g.

* circle: $\phi = \sqrt{(x - x_c)^2 + (y - y_c)^2} - r$

* ellipse: $\phi = \sqrt{\left(\frac{x-x_c}{a}\right)^2 + \left(\frac{y-y_c}{b}\right)^2} - 1$ (redistantiate)

→ set in JSON file

>>> JSON file

```
{
  "Name": "LS",
  "Model": "Advection",
  "Parameters":
  {
    "x0": "0.", "y0": "0.", "a0": "1.", "b0": "2."
  },
  "InitialConditions":
  {
    "[levelset_prefix]":
    {
      "Dirichlet":
      {
        "":
        {
          "expr": "sqrt( (x-x0)*(x-x0)/(a0*a0) + (y-y0)*(y-y0)/(b0*b0) ) - 1",
            "x": "x0", "y": "y0", "a0": "a0", "b0": "b0"
        }
      },
      "shapes":
      {
        "sphere1":
        {
          "shape": "sphere",
          "xc": "0",
          "yc": "0",
          "zc": "0",
          "radius": "0.5"
        }
      }
    }
  }
}
```

>>> Geometrical quantities

Interface

$$\delta_{\phi}(\mathbf{x}) = \begin{cases} 0 & \text{if } \phi(\mathbf{x}) < -\varepsilon \\ \frac{1}{2\varepsilon} \left[1 + \cos\left(\frac{\pi\phi}{\varepsilon}\right) \right] & \text{if } -\varepsilon \leq \phi(\mathbf{x}) \leq \varepsilon \\ 0 & \text{if } \phi(\mathbf{x}) > \varepsilon \end{cases}$$

`element_levelset_ptrtype const& dirac() const;`

Inner and outer

$$H_{\phi}(\mathbf{x}) = \begin{cases} 0 & \text{if } \phi(\mathbf{x}) < -\varepsilon \\ \frac{1}{2} \left[1 + \frac{\phi}{\varepsilon} + \frac{1}{\pi} \sin\left(\frac{\pi\phi}{\varepsilon}\right) \right] & \text{if } -\varepsilon \leq \phi(\mathbf{x}) \leq \varepsilon \\ 1 & \text{if } \phi(\mathbf{x}) > \varepsilon \end{cases}$$

`element_levelset_ptrtype const& heaviside() const;`

>>> Geometrical quantities: related options

$$\int_{\Gamma} f(\mathbf{x}) \rightarrow \int_{\Omega} f(\mathbf{x}) \delta_{\phi}(\mathbf{x}) |\nabla\phi| \quad \text{or} \quad \int_{\Omega} f(\mathbf{x}) \delta_{\frac{\phi}{|\nabla\phi|}}(\mathbf{x})$$
$$\int_{\{\phi(\mathbf{x}) \geq 0\}} f(\mathbf{x}) \rightarrow \int_{\Omega} f(\mathbf{x}) H_{\phi}(\mathbf{x})$$

Options

- [prefix].thickness-interface **double**: value of ε
- [prefix].use-regularized-phi **bool**: use $\frac{\phi}{|\nabla\phi|}$
- [prefix].h-d-nodal-proj **bool**: use nodal projection (or L_2 projection)

>>> Normal and curvature

Normal

$$\mathbf{N} = \frac{\nabla\phi}{|\nabla\phi|}$$

```
element_levelset_vectorial_ptrtype const& normal() const;
```

```
--[prefix].smooth-gradient bool: smooth  $\nabla\phi$ 
```

Curvature

$$\kappa = \nabla \cdot \mathbf{N}$$

```
element_levelset_ptrtype const& curvature() const;
```

```
--[prefix].smooth-curvature bool: smooth  $\kappa$ 
```

>>> Computing derivatives: projector

L_2 projector

$$\Pi_{L_2}[f] = \hat{f} \in \mathbb{V} \subset L_2 \cap \mathcal{C}^0 \text{ s.t.}$$

$$\int_{\Omega} \hat{f} v = \int_{\Omega} f v, \quad \forall v \in \mathbb{V}$$

e.g. $\mathbf{N} = \Pi_{L_2} \left[\frac{\nabla \phi}{|\nabla \phi|} \right]$

Need to have

(polynomial order of \mathbb{V}) \geq (derivative order of f)

Options

Standard **backend** options with prefixes

--[levelset_prefix].projector-l2.[backend_option]

--[levelset_prefix].projector-l2-vec.[backend_option]

>>> Computing derivatives: smoother

Smoother

For low-order discretization spaces, high-order derivatives become **inaccurate**

⇒ need to smooth out numerical errors

$\Pi_{sm}[f] = \tilde{f} \in \mathbb{W} \subset H^1 \cap C^0$ s.t.

$$\int_{\Omega} \tilde{f} w + \int_{\Omega} \eta \nabla \tilde{f} \cdot \nabla w - \int_{\partial\Omega} \eta \nabla \tilde{f} \cdot \mathbf{n} w = \int_{\Omega} f w, \quad \forall w \in \mathbb{W}$$

→ adds diffusion errors: need to tune η

Options

--[levelset_prefix].smoother.[backend_option]

--[levelset_prefix].smoother.smooth-coeff **double**: tune η

Same for smoother-vec.

>>> Redistanciation

After some iterations, need to restore $|\nabla\phi| = 1$
 \Rightarrow fast-marching or Hamilton-Jacobi method

>>> Redistanciation

After some iterations, need to restore $|\nabla\phi| = 1$

\Rightarrow fast-marching or Hamilton-Jacobi method



- + Fast $\mathcal{O}(N \log(N))$
- + "Exact"
- Limited to P_1 space
- Not fully parallel



Solve $\partial_\tau\phi = \text{sign}(\phi)(1 - |\nabla\phi|)$

- + Any P_N
- + Scales in parallel
- Can need many iterations
- No stop criterion
- Interface not anchored

>>> Redistanciation options

--[prefix].reinit-method **fm/hj**: select the redistanciation method

Fast-marching

--[prefix].fm-init-first-elts-strategy **int**: choose strategy to initialize the first values

0: do nothing

1: use $\frac{\phi}{|\nabla\phi|}$ on interface elements

2: use a few steps of Hamilton-Jacobi

Hamilton-Jacobi

--[prefix].reinit-hj.tol **double**: tolerance on relative rate of change for convergence

--[prefix].reinit-hj.max-iter **int**: max number of HJ iterations

>>> Physics enhancements

Stretch

If $\nabla \cdot \mathbf{u} = 0$, $s = |\nabla\phi|$ accounts for the changes in surface measure

>>> Physics enhancements

Stretch

If $\nabla \cdot \mathbf{u} = 0$, $s = |\nabla \phi|$ accounts for the changes in surface measure

But inaccurate + redistanciation $\Rightarrow s = 1$

\rightarrow advect as an independent field

$$\partial_t s + \mathbf{u} \cdot \nabla s + (\mathbf{N} \otimes \mathbf{N}) : D(\mathbf{u}) s = 0$$

--[prefix].use-stretch-augmented **bool**

--[prefix].reinit-stretch-augmented **bool**: reinitialize s to 1 but without changing its value near the interface

>>> Physics enhancements

Stretch

If $\nabla \cdot \mathbf{u} = 0$, $s = |\nabla\phi|$ accounts for the changes in surface measure

But inaccurate + redistanciation $\Rightarrow s = 1$

\rightarrow advect as an independent field

$$\partial_t s + \mathbf{u} \cdot \nabla s + (\mathbf{N} \otimes \mathbf{N}) : D(\mathbf{u}) s = 0$$

--[prefix].use-stretch-augmented **bool**

--[prefix].reinit-stretch-augmented **bool**: reinitialize s to 1 but without changing its value near the interface

Membrane mechanics

--[prefix].use-cauchy-augmented **bool**

Multifluid framework

Purpose

- * Physical parameters and forces
- * Couple level-set and fluid
 - Solve Navier-Stokes equation $\rightarrow \mathbf{u}^{(n+1)}, p^{(n+1)}$
 - Advect level set with $\mathbf{u}^{(n+1)} \rightarrow \phi^{(n+1)}$
 - Update fluid parameters and forces
 $\rightarrow \rho_\phi^{(n+1)}, \mu_\phi^{(n+1)}, \mathbf{F}_\phi^{(n+1)}$
- * Manage redistantiation

Toolbox

`feelp_toolbox_multifluid_[2d/3d]`

>>> Problem setup and physical parameters

Generic options

```
--multifluid.nfluids int: number of fluids considered ( $\geq 2$ )  
--multifluid.[mesh options]  
  
--multifluid.fluid.[fluid option]: fluid solver options  
--multifluid.levelset[i].[levelset option]: levelset options
```

Physical parameters

```
--multifluid.fluid.[rho/mu]: outer fluid parameters  
--multifluid.levelset[i].[rho/mu]: levelset i inner fluid  
parameters
```

>>> Forces

Based on

```
template<class LevelSetType >
class InterfaceForcesModel
{
    [...]
    virtual void build( std::string const& prefix,
                       levelset_ptrtype const& ls );
    void updateInterfaceForces( element_ptrtype & F,
                               bool overwrite = false) const;
    [...]
private:
    virtual void updateInterfaceForcesImpl( element_ptrtype & F )
        const =0;
    [...]
};
```

and registered in MultiFluid force factory.

>>> Forces

Available forces

--multifluid.levelset [i].interface-forces-model **string**: enable given force model

- * helfrich
- * inextensibility-force
- * linear-elastic-force
- * skalak-force

A simple demo...