FreeVol++ : 2D finite volume solver inside FreeFem++

Georges SADAKA[1], Frédéric HECHT[2], Nicolas SEGUIN[3], Denys DUTYKH[4] and Pierre JOLIVET[5]

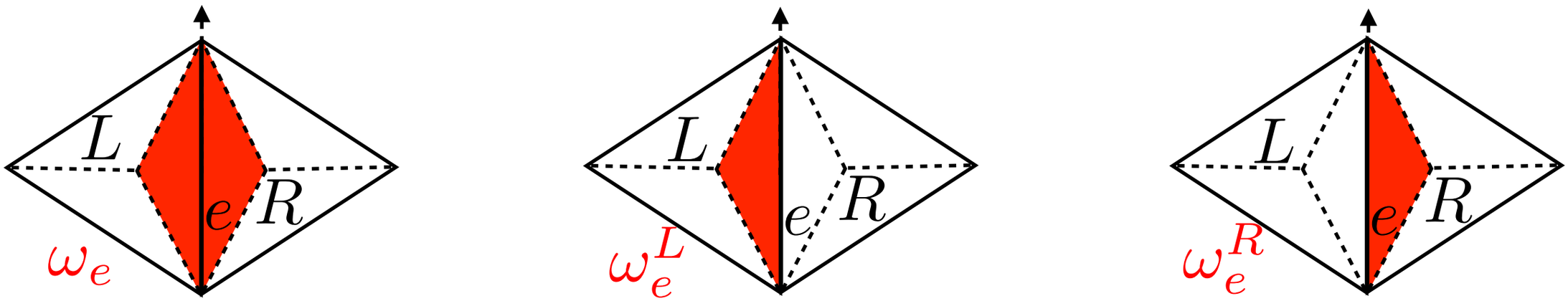
[1]LMRS, Univ. de Rouen, [2]LJLL, UPMC, [3]IRMAR, Univ. De Rennes,
[4]LAMA, CNRS, Chambéry, [5]IRIT-ENSEEIHT, CNRS, Toulouse

The 12th tutorial and Workshop on FreeFem++ 10-11/12/20

Outline

Intoduction

We let Ω the domain, $\mathcal{T}_h$ triangulation of Ω and $\mathcal{E}_h$ edges of $\mathcal{T}_h$, $\omega_e = \{x \in \Omega / \forall e' \in \mathcal{E}_h, d(x,e) \leq d(x,e')\}$, ω_e^L, ω_e^R is the partition of ω_e in two by the edge e, respecting the global orientation of the edge, we remark that the set ω_e^R can be empty if the edge on the boundary.



Let be the $\mathbf{P0}(\omega)$ constant function on domain ω and we define :

$V0_h = \left\{u_h \in L^2(\bar{\Omega}) | \forall T \in \mathcal{T}_h,\, u_{h|T} \in \mathbf{P0}(T)\right\}$ `fespace V0h(Th,P0);`

$V0E_h = \left\{u_h \in L^2(\bar{\Omega}) | \forall e \in \mathcal{E}_h,\, u_{h|\omega_e} \in \mathbf{P0}(\omega_e)\right\}$ `fespace V0Eh(Th,P0Edge);`

$V0Edc_h = \left\{u_h \in L^2(\bar{\Omega}) | \forall e \in \mathcal{E}_h,\, u_{h|\omega_e^L} \in \mathbf{P0}(\omega_e^L) \text{ and } u_{h|\omega_e^R} \in \mathbf{P0}(\omega_e^R)\right\}$

```
load "Element_PkEdge" fespace V0Edch(Th,P0edgedc);
```

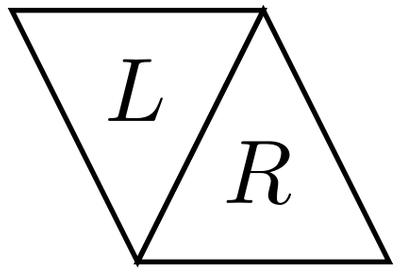
⋆ `area` gives the area of the current triangle.

⋆ `qforder` the order of the Gauss formula, here will be equal to 1.

⋆ `lenEdge` gives the length of the current edge of the current triangle.

⋆ `edgeOrientation` : $\varepsilon_{edge} = \{-1, 1\}$ orients the current edge of the current triangle.

⋆ `nTonEdge` $= 1$ if the current triangle has a boundary edge and $= 2$ otherwise.



⋆ `InternalEdge` for the internal edge.

⋆ `BoundaryEdge` for the boundary edge.

⋆ `otherside` give the otherside R adjacent triangle of the current triangle L on the current edge.

⋆ `intalledges` gives the integral on all edges of all triangles so all internal edges are see two times.

⋆ `jump` $[u] = u_R - u_L$ is the jump of u across an edge.

Intoduction

$\forall u, v \in V0_h$

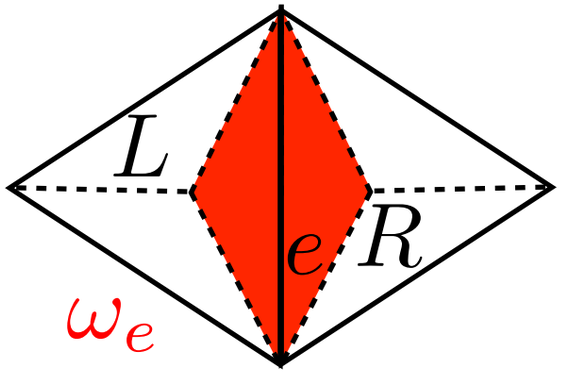
`intalledges(Th,qforder=1)(u*v)` $= \sum_T \int_{\partial T} u \cdot v = \sum_T |\partial T| u,$

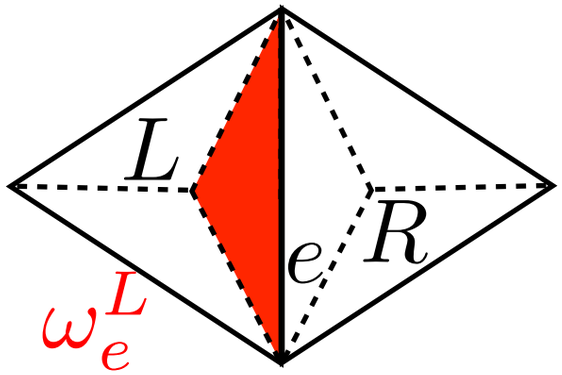
`intalledges(Th,qforder=1)(u*v*area/lenEdge)` $= \sum_T \int_{\partial T} \frac{|T| \cdot u \cdot v}{|\partial T|} = \sum_T |T| u,$

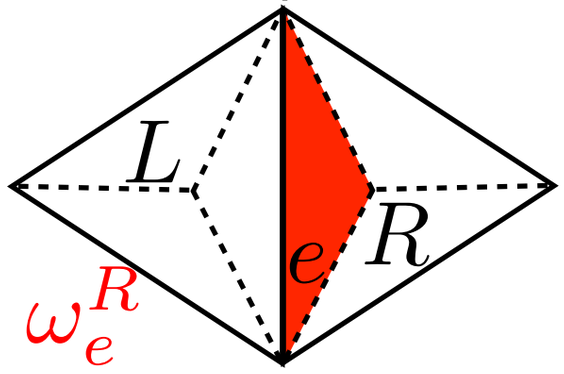
$\forall u \in V0_h, \forall v \in V0E_h$

`intalledges(Th,qforder=1)(edgeOrientation*u*v/2)` $= \sum_T \int_{\partial T} \varepsilon_{edge} u \cdot v,$

`intalledges(Th,qforder=1)(edgeOrientation*jump(u)*v/2)` $= \sum_T \int_{\partial T} \varepsilon_{edge}(u_R - u_L) \cdot v$







$\forall u \in V0_h, \forall v \in V0Edc_h$

`intalledges(Th,qforder=1)(edgeOrientation*u*v)` $= \sum_T \int_{\partial T} \varepsilon_{edge} u \cdot v,$

`intalledges(Th,qforder=1)(edgeOrientation*jump(u)*v)` $= \sum_T \int_{\partial T} \varepsilon_{edge}(u_R - u_L) \cdot v$

Consider a simple 2D advection problem defined by :

$$\begin{cases} \partial_t u + \vec{\mathbf{V}} \cdot \vec{\nabla} u = 0, \text{ where } \vec{\mathbf{V}} = [V_1, V_2], \nabla \cdot \vec{\mathbf{V}} = 0 \\ u(0, x, y) = u_0(x, y) \end{cases}$$

$\forall T \in \mathcal{T}_h \subset \Omega$, $\forall u, V_1, V_2 \in V0_h$ this system is equivalent to :

$$\partial_t \sum_T \int_T u + \sum_T \int_T \vec{\mathbf{V}} \cdot \vec{\nabla} u = 0$$

after I.P.P. and using $\int_T u = |T| u$:

$$\partial_t \sum_T u + \frac{1}{|T|} \underbrace{\sum_T \int_{\partial T} \varepsilon_{edge} \left(\vec{\mathbf{V}} \cdot \vec{n}_{out} \right) u}_{\phi_{num}(u)} = 0,$$

find $u_h \in V0_h$ such that $\forall v_h \in V0Edc_h$ we have :

$$\partial_t u_h + \mathcal{A} \underbrace{\sum_T \int_{\partial T} \varepsilon_{edge} \left(\vec{\mathbf{V}} \cdot \vec{n}_{out} \right) u_h \cdot v_h}_{\phi_{num}(u_h, v_h)} = 0,$$

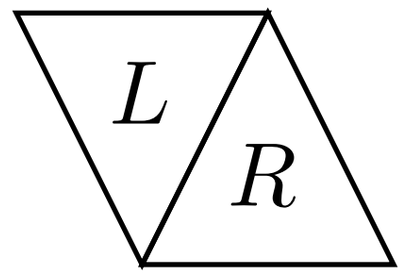
$$\mathcal{A}_{i,j} = \sum_T \int_{\partial T} \varepsilon_{edge} \frac{\varphi_i \psi_j}{|T||\partial T|}, \forall \varphi_i \in V0Edc_h, \forall \psi_j \in V0_h.$$

```
load "Element_PkEdge"
fespace V0Edch(Th,P0edgedc);
fespace V0h(Th,P0);
varf vA(phii,psij)=intalledges(Th,qforder=1)(edgeOrientation*phii*psij/lenEdge/area);
matrix VA = vA(V0Edch,V0h);
```

Upwind flux : $\forall u_h \in V0_h, \forall v_h \in V0Edc_h, \phi_{num}(u_h, v_h) =$

$$\sum_T \int_{\partial T} \varepsilon_{edge} \begin{cases} \left(\vec{\mathbf{V}}_{|L} \cdot \vec{n}_{L\to R}\right) u_{h|L} v_h & \text{if } \vec{\mathbf{V}}_{|L} \cdot \vec{n}_{L\to R} > 0 \\ \left(\vec{\mathbf{V}}_{|R} \cdot \vec{n}_{L\to R}\right) u_{h|R} v_h & \text{otherwise} \end{cases},$$

$$\text{B.C. : } \sum_{T_\Gamma} \int_{\partial T_\Gamma} \begin{cases} \left(\vec{\mathbf{V}} \cdot \vec{n}_{out}\right) u_h \cdot v_h & \text{if } \vec{\mathbf{V}} \cdot \vec{n}_{out} > 0 \\ 0 & \text{otherwise} \end{cases}$$




```
macro u0()(otherside(u))//
macro V10()(otherside(V1))//
macro V20()(otherside(V2))//
func NN=[N.x,N.y];
macro Upwind()([V1,V2]'*NN > 0. ? [V1,V2]'*NN*u : [V10,V20]'*NN*u0)//
macro UpwindBC()([V1,V2]'*NN > 0. ? [V1,V2]'*NN*u : 0.)//
varf vphiUpwind(unused,vh) = intalledges(Th,qforder=1)
(edgeOrientation*(InternalEdge*Upwind+BoundaryEdge*UpwindBC)*vh);
V0Edch phi;
phi[] = vphiUpwind(0,V0Edch);
```

CFL (C. Berthon *et* al. 14)

$$\delta t \frac{\mathcal{P}_T}{|T|} \max(|\lambda_L|, |\lambda_R|) \leq 1, \forall T$$

where $\mathcal{P}_T$ is the perimeter of the triangle T, $\forall T, \forall u_h, v_h \in V0_h$:

$$\delta t \max\left(\sum_T \int_{\partial T} \frac{\max(|\vec{\mathbf{V}}_L \cdot \vec{n}_{L\to R}|, |\vec{\mathbf{V}}_R \cdot \vec{n}_{L\to R}|)}{|T|} v_h \right) \leq 1$$

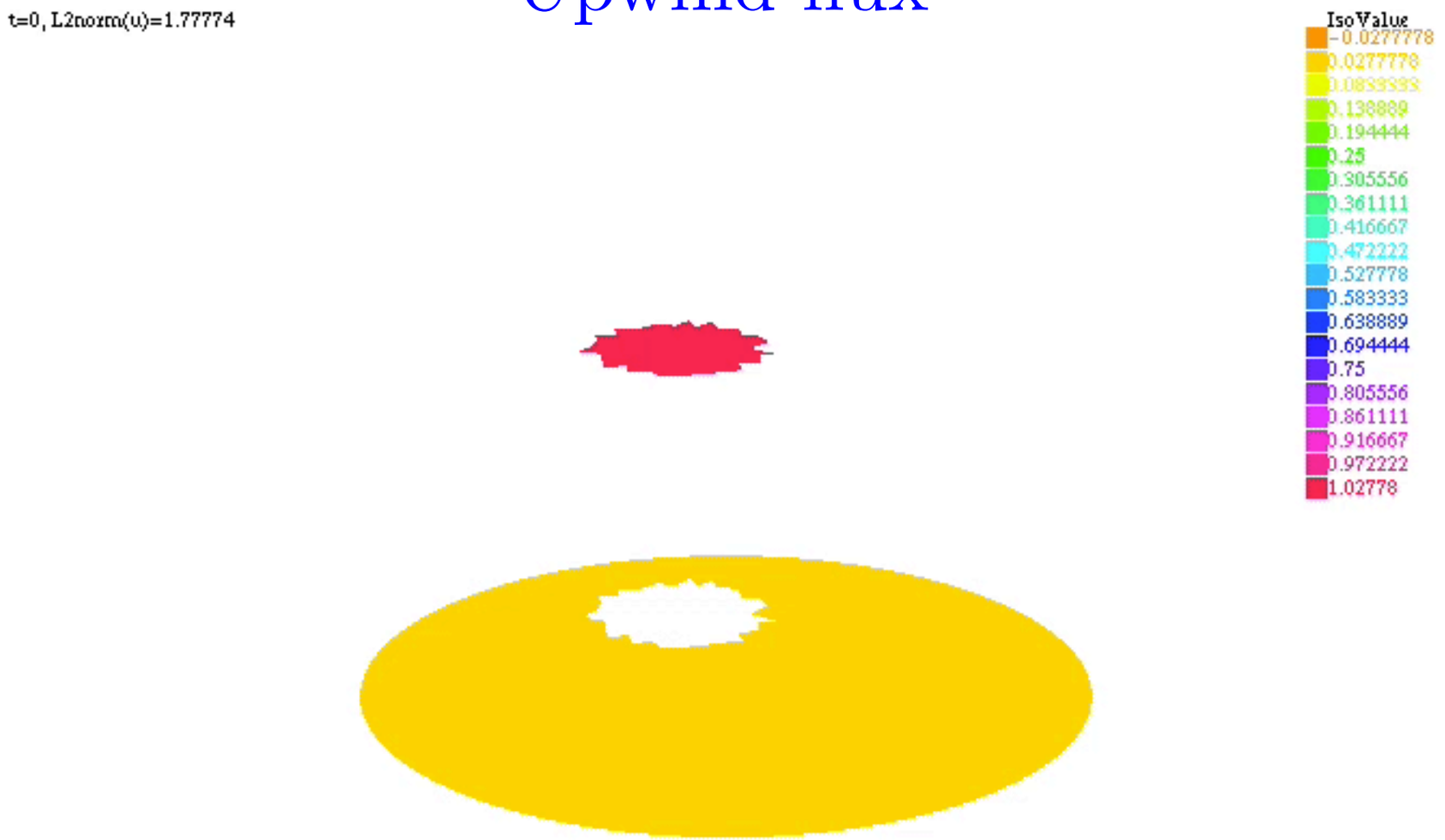
```
macro cCFL()(max(abs([V1,V2]'*NN),abs([V10,V20]'*NN)))//
varf vdt(uu,vh) = intalledges(Th,qforder=1)(vh/area*cCFL);
V0h pdt;    pdt[]= vdt(0,V0h);
real dt=CFL/pdt[].max;
```

Explicit Euler scheme : $u_h^{n+1} = u_h^n - \delta t \mathcal{A}\, \phi_{num}(u_h^n, v_h)$

```
V0h u,up;
for(real t=0.; t<TF; t+=dt){
    phi[]=vphiUpwind(0,V0Edch);
    up[] = VA*phi[];
    u[] -= dt*up[];
}
```

```
real x0=-1.5,y0=-1.5,R=4.;
V0h u=dist(x-x0,y-y0)<=(R/4.),V1=y,V2=-x;
```

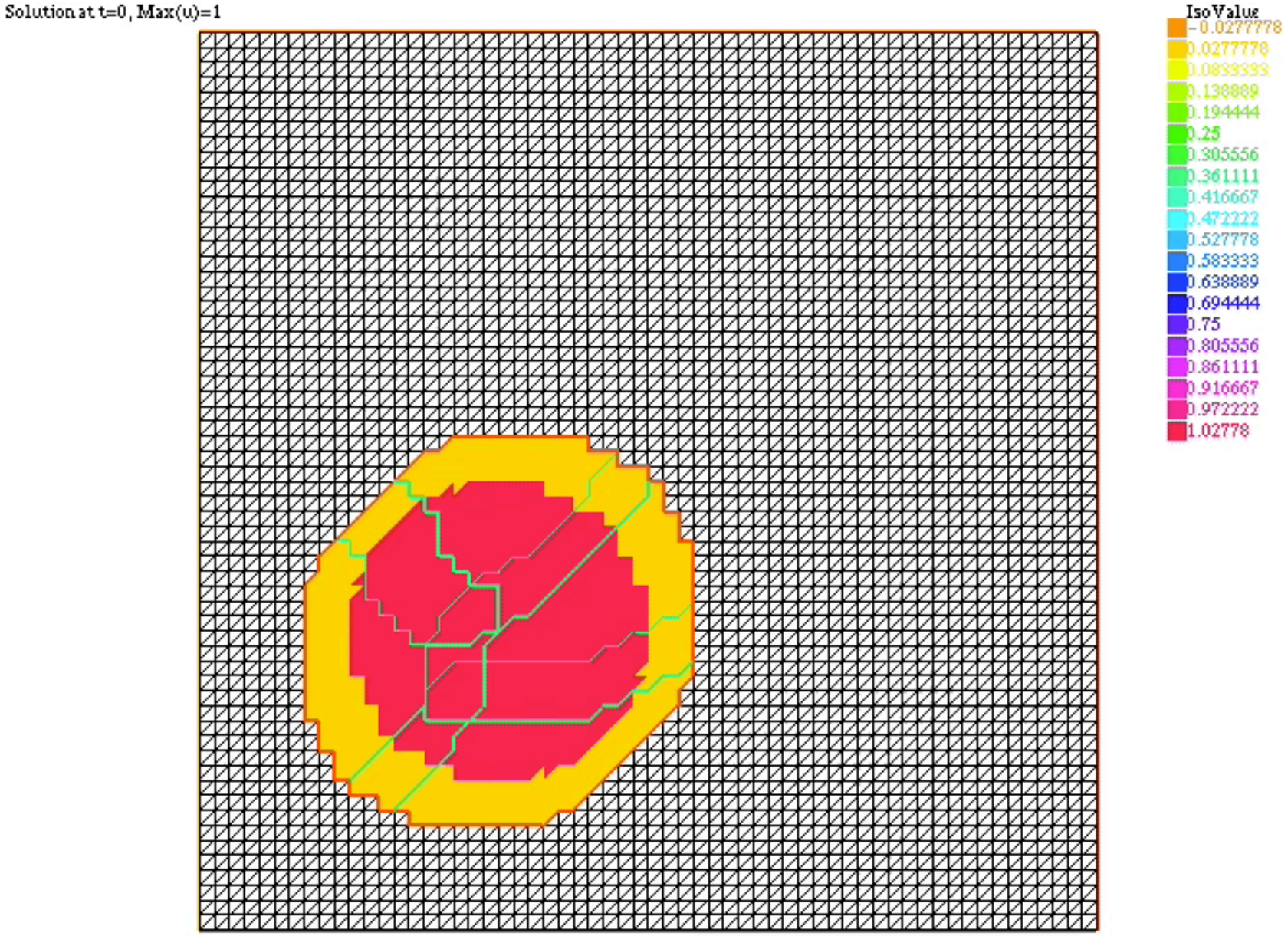
Upwind flux



```
load "Element_PkEdge"
real x0=-1.,y0=-1.,CFL=.9,R=4.,TF=4.;
mesh ThGlobal=square(30,30,[x*6.-3.,y*6.-3.]); mesh Th=ThGlobal;
fespace V0Edch(Th,P0edgedc);
fespace V0h(Th,P0);
V0h u=dist(x-x0,y-y0)<=(R/4.),up,V1=1.,V2=1.;
V0Edch phi;
varf vA(phii,psij)=intalledges(Th,qforder=1)(edgeOrientation*phii*psij/lenEdge/area);
matrix VA = vA(V0Edch,V0h);
func NN=[N.x,N.y];
macro u0()(otherside(u))//
macro V10()(otherside(V1))//
macro V20()(otherside(V2))//
macro Upwind()([V1,V2]'*NN > 0. ? [V1,V2]'*NN*u : [V10,V20]'*NN*u0)//
macro UpwindBC()([V1,V2]'*NN > 0. ? [V1,V2]'*NN*u : 0.)//
varf vphiUpwind(unused,vh) = intalledges(Th,qforder=1)
(edgeOrientation*(InternalEdge*Upwind+BoundaryEdge*UpwindBC)*vh);
macro cCFL()(max(abs([V1,V2]'*NN),abs([V10,V20]'*NN)))//
varf vdt(uu,vh) = intalledges(Th,qforder=1)(vh/area*cCFL);
V0h pdt;    pdt[]= vdt(0,V0h);
real dt=CFL/pdt[].max;
for(real t=0.; t<TF; t+=dt){
    phi[]=vphiUpwind(0,V0Edch);
    up[] = VA*phi[];
    u[] -= dt*up[];
}
```

Upwind flux

V1=y,V2=−x;



The 2D Saint-Venant system for Shallow Water writes as follows :

$$\partial_t \mathbf{U} + \mathbf{div}([F(\mathbf{U}), G(\mathbf{U})]) = S(\mathbf{U}),$$

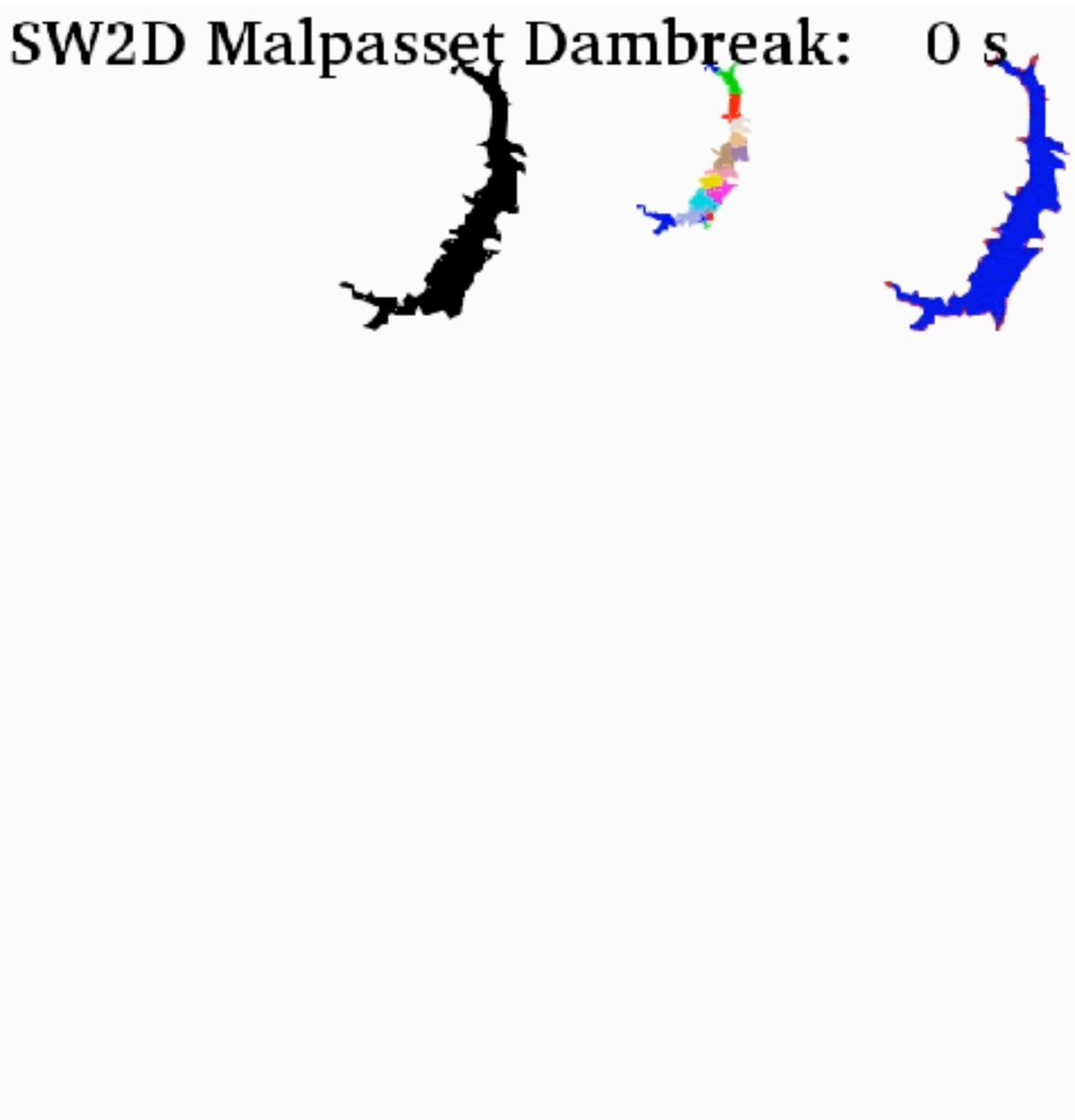
$$\mathbf{U} = \begin{pmatrix} H \\ Hu \\ Hv \end{pmatrix}, F(\mathbf{U}) = \begin{pmatrix} Hu \\ Hu^2 + \frac{g}{2}H^2 \\ Huv \end{pmatrix}, G(\mathbf{U}) = \begin{pmatrix} Hv \\ Huv \\ Hv^2 + \frac{g}{2}H^2 \end{pmatrix},$$

$$S(\mathbf{U}) = \begin{pmatrix} 0 \\ -gH\partial_x z_b \\ -gH\partial_y z_b \end{pmatrix} = \begin{pmatrix} 0 \\ \partial_x(\frac{g}{2}H^2) \\ \partial_y(\frac{g}{2}H^2) \end{pmatrix}$$

find $\mathbf{U}_h \in V03_h$ such that $\forall \mathbf{V}_h = [vH_h, vHu_h, vHv_h]^t \in V0Edc3_h$ we have:

$$\partial_t \mathbf{U}_h + \mathcal{A} \sum_T \int_{\partial T} \varepsilon_{edge} \left([F(\mathbf{U}_h), G(\mathbf{U}_h)] \cdot \vec{n}_{out} - \frac{g}{2} H^2 \, [0, \vec{n}^x_{out}, \vec{n}^y_{out}]^t \right) \cdot \mathbf{V}_h = 0$$

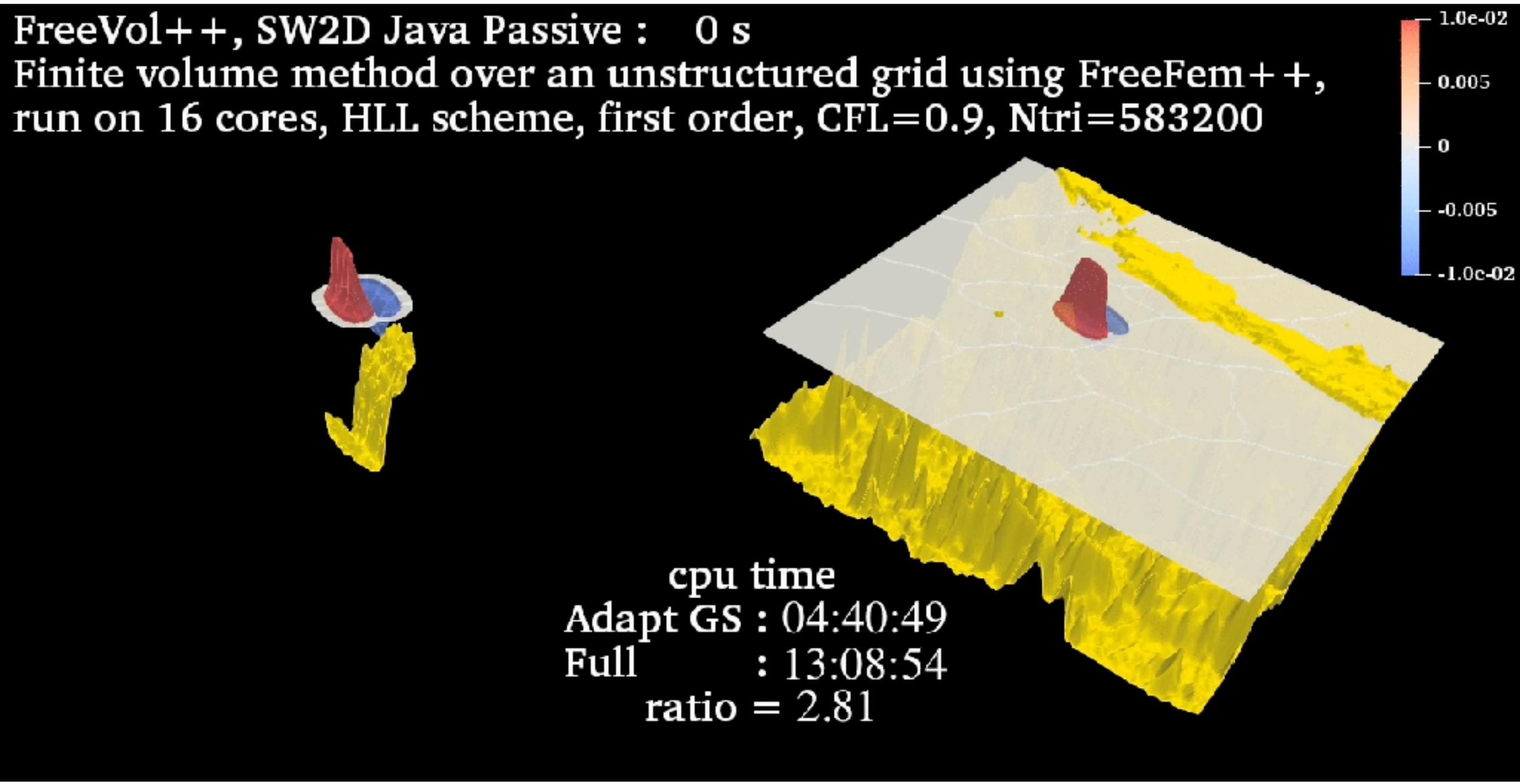
Malpasset Dambreak (December 2 1959)



FreeVol++
Run on 16 cores
HLL scheme
no manning friction
first order
CFL=0.9
Ntri=33170

cpu time :
Adapt GS: 06:50:48
Full : 09:37:39
ratio=1.4

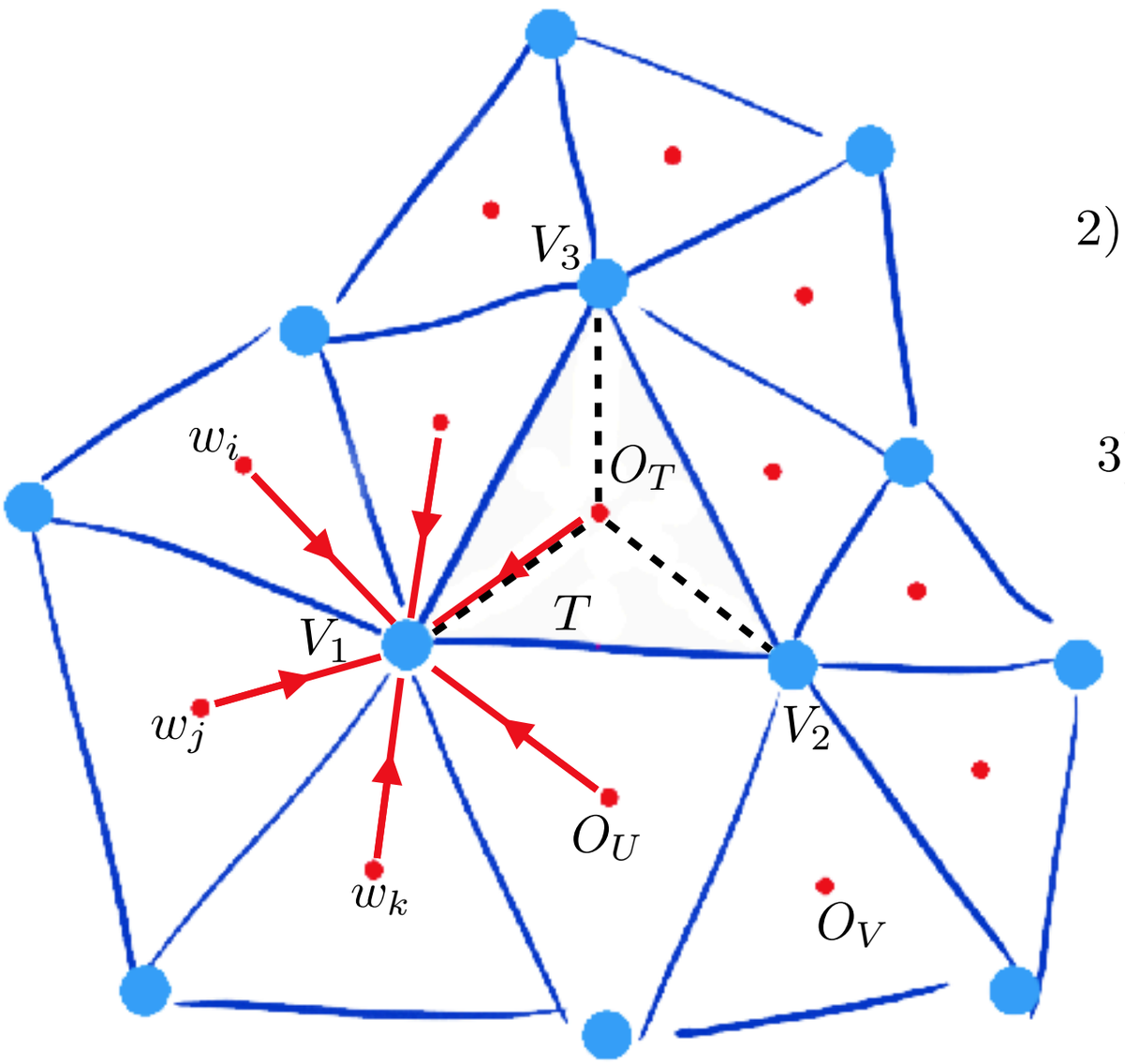
Propagation of a *Tsunami* wave near Java (July, 2006)



Gradient computation

Dutykh *et* al. 2009

Holmes *et* al. 1989



1) $w_i = 1 + \Delta w_i, \min \left(\frac{1}{2} \sum_{i=1}^{d(N)} \left(||\vec{X}_{O_i} - \vec{X}_{V_j}|| \Delta w_i \right)^2 \right)$

$$\Delta w_i = \frac{\lambda(x_{O_i} - x_{V_j}) + \mu(y_{O_i} - y_{V_j})}{||\vec{X}_{O_i} - \vec{X}_{V_j}||^2}$$

2) $u_{V_j} = \sum_{i=1}^{d(N)} w_i u_{O_i} \Big/ \sum_{i=1}^{d(N)} w_i, j = \{1, ..., Th.nv\}$

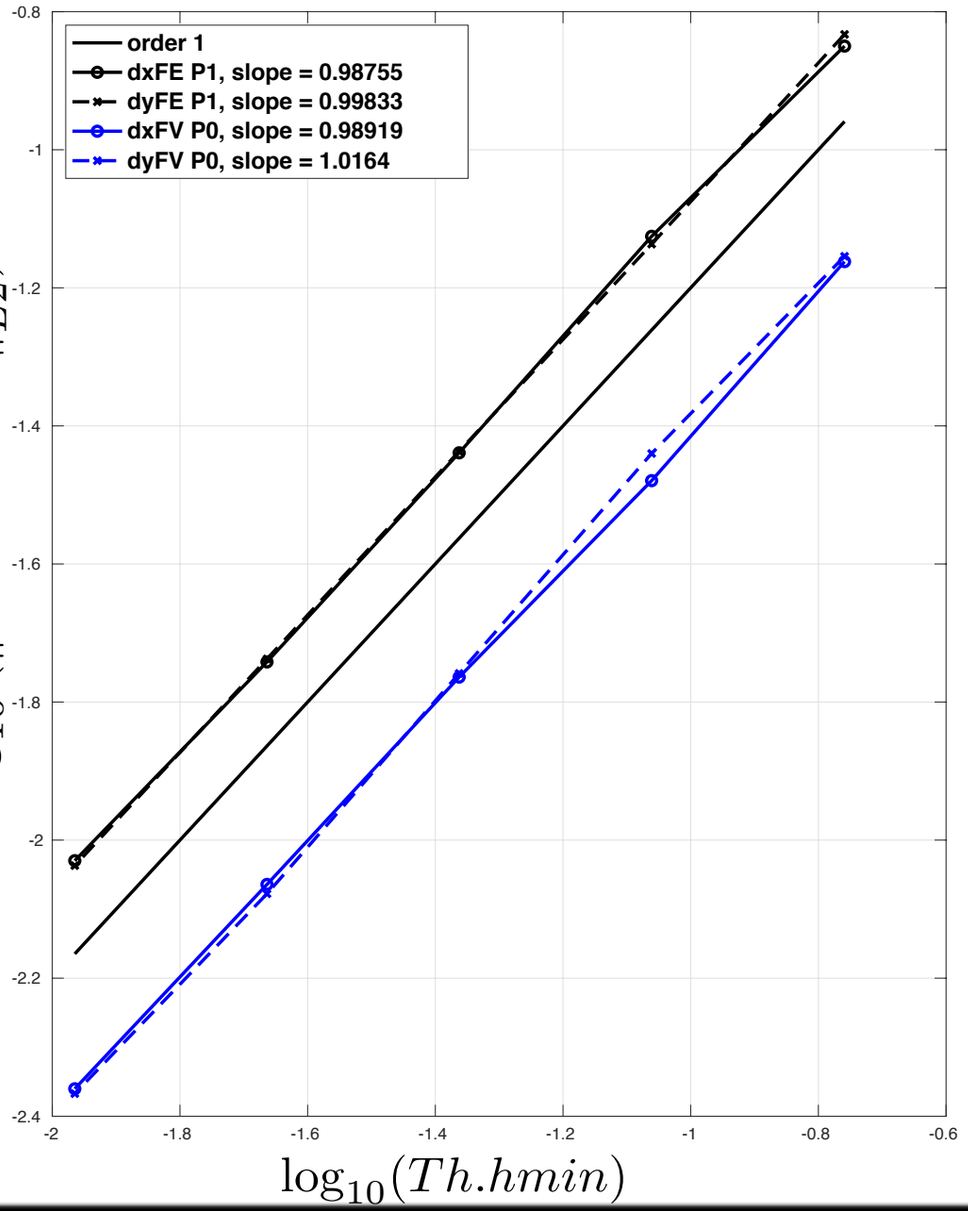
3)
$$\begin{aligned} u_{V_1} - u_O &= (\nabla u)_T \cdot (\vec{X}_1 - \vec{X}_T) \\ u_{V_2} - u_O &= (\nabla u)_T \cdot (\vec{X}_2 - \vec{X}_T) \\ u_{V_3} - u_O &= (\nabla u)_T \cdot (\vec{X}_3 - \vec{X}_T) \end{aligned}$$

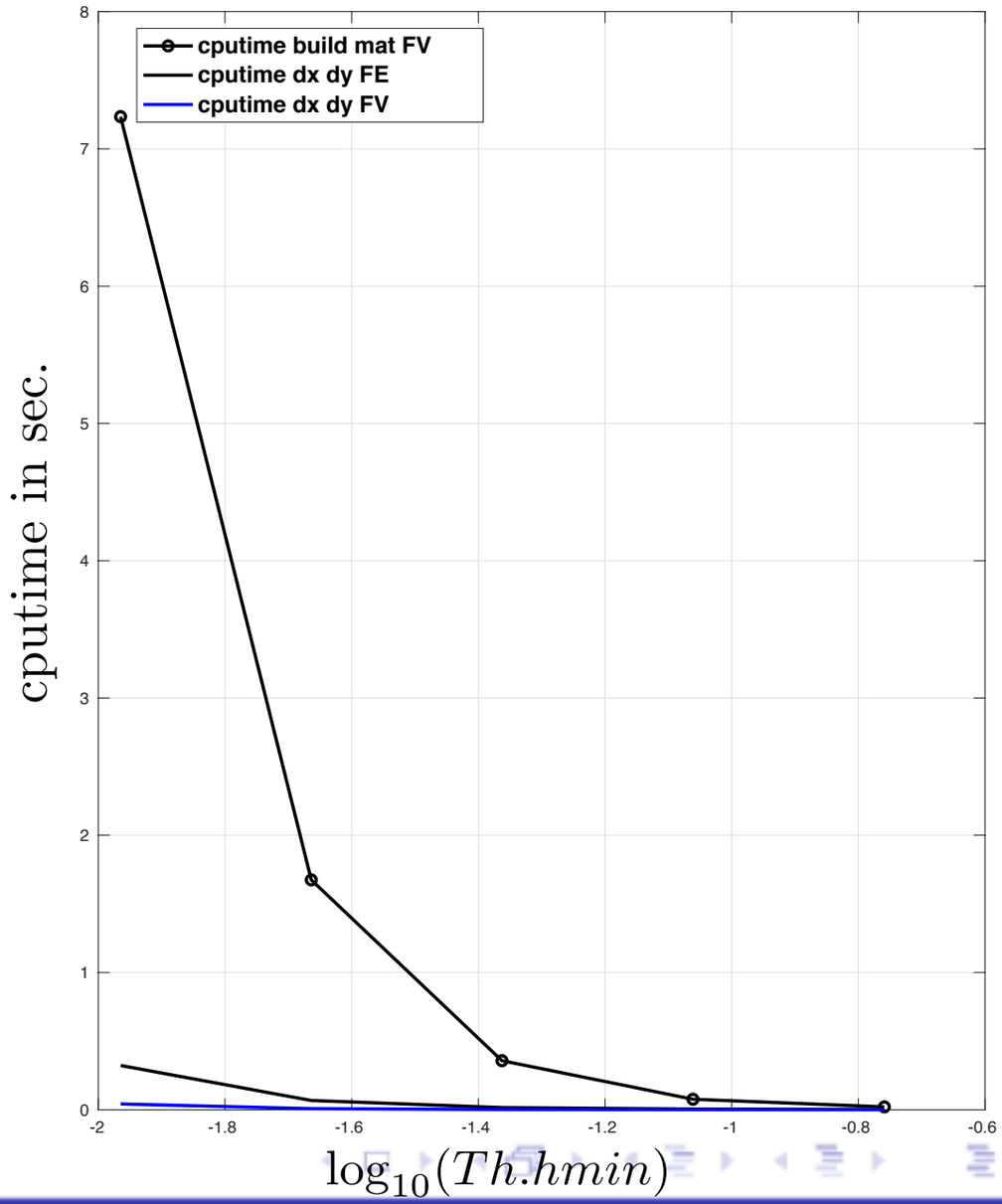
least square, with

$$\omega_i = \frac{||\vec{X}_1 - \vec{X}_T||^{-1}}{\sum_{j=1}^{3} ||\vec{X}_j - \vec{X}_T||^{-1}}, i = \{1, ..., Th.nt\}$$

Gradient computation

```
load "MatD-VFP0"
border C(t=0,2.*pi) { x=5.*cos(t); y=5.*sin(t);}
mesh Th=buildmesh(C(Np));
matrix AD = MatVFD(Th);
fespace Ph(Th,P0);
fespace Ph2(Th,[P0,P0]);
Ph u0=exp(-x^2/2.-y^2/2.);
Ph2 [u0x,u0y];
u0x[] = AD*u0[];
```

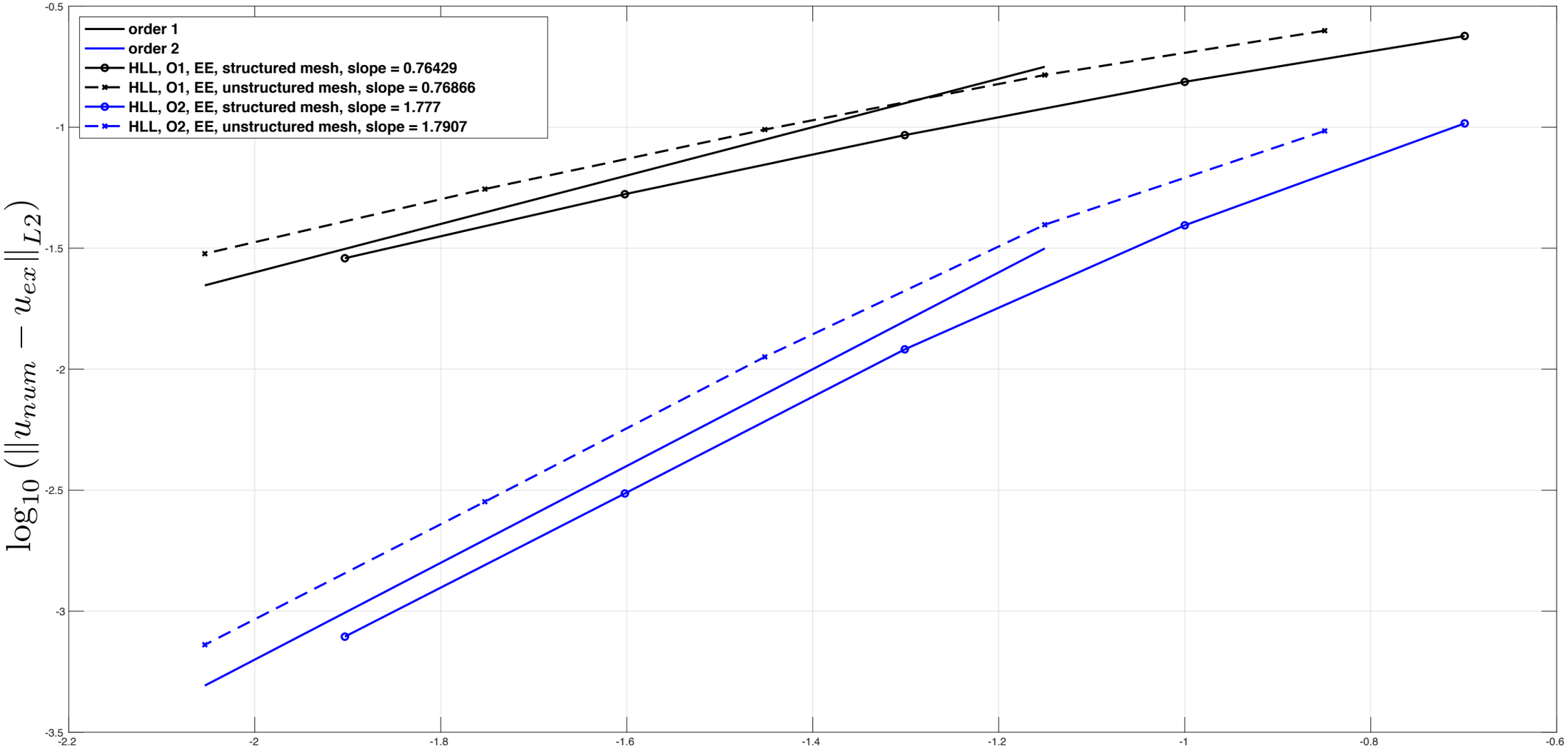




Gradient computation

$$\begin{cases} \partial_t u + \vec{\mathbf{V}} \cdot \vec{\nabla} u = 0, \text{ where } \vec{\mathbf{V}} = [.5, 0], \text{ on } \Omega = [0, 10] \times [0, 3] \\ u(t, x, y) = \dfrac{1}{cosh^2(c_x(x - x_c - V1 \cdot t)^2 + c_y(y - y_c - V2 \cdot t)^2)} \\ (x_c, y_c) = (5., 1.5), c_x = c_y = 5. \end{cases}$$

Convergence rate in space with HLL flux, explicit Euler time scheme and a fixed $\delta t = 1.e{-}4$



$$\eta_t + \nabla \cdot \mathcal{V} + \nabla \cdot (\eta \mathcal{V}) - b\Delta\eta_t = 0;$$

$$\mathcal{V}_t + \nabla\eta + \tfrac{1}{2}\nabla|\mathcal{V}|^2 - d\Delta\mathcal{V}_t = 0,$$

where b and d are positive parameters such that $b + d = 1/3$.

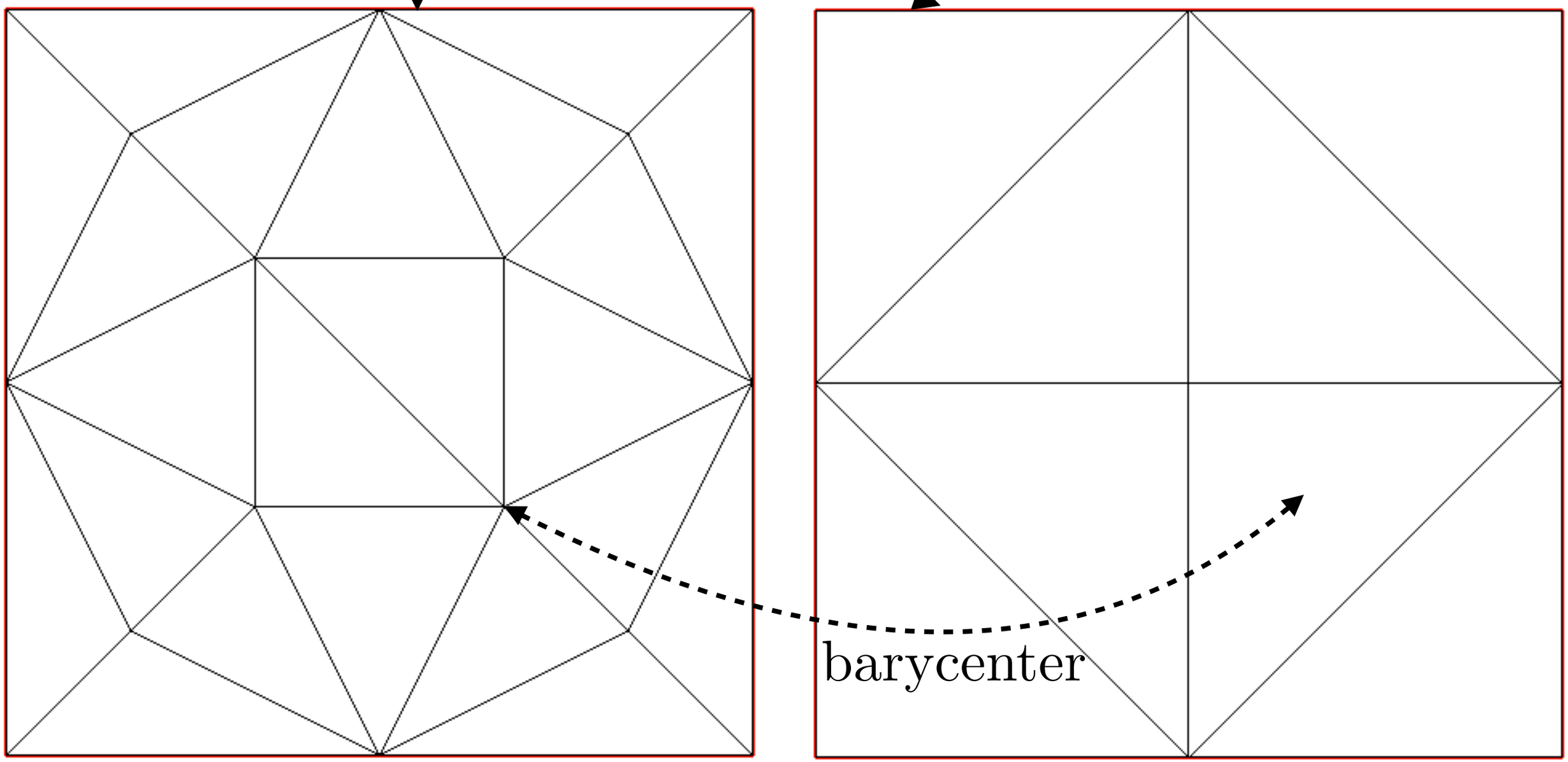
$$\mathcal{A}_{BBM}\partial_t\mathbf{E} + \mathbf{div}([F(\mathbf{E}), G(\mathbf{E})]) = 0,$$

$$\mathcal{A}_{BBM} = \begin{pmatrix} \bullet - b\Delta\bullet & 0 & 0 \\ 0 & \bullet - d\Delta\bullet & 0 \\ 0 & 0 & \bullet - d\Delta\bullet \end{pmatrix},$$

$$\mathbf{E} = \begin{pmatrix} \eta \\ u \\ v \end{pmatrix}, F(\mathbf{E}) = \begin{pmatrix} (1+\eta)u \\ \eta + \frac{1}{2}(u^2+v^2) \\ 0 \end{pmatrix}, G(\mathbf{E}) = \begin{pmatrix} (1+\eta)v \\ 0 \\ \eta + \frac{1}{2}(u^2+v^2) \end{pmatrix}.$$

$$\underbrace{\mathcal{A}_{BBM}\partial_t \mathbf{E}}_{\mathbf{FE}} + \underbrace{\mathbf{div}([F(\mathbf{E}), G(\mathbf{E})])}_{\mathbf{FV}} = 0,$$

$\mathrm{Th1}, \mathbb{P}_1$ $\mathrm{Th}, \mathbb{P}_0, \mathbb{P}_{0edge}$

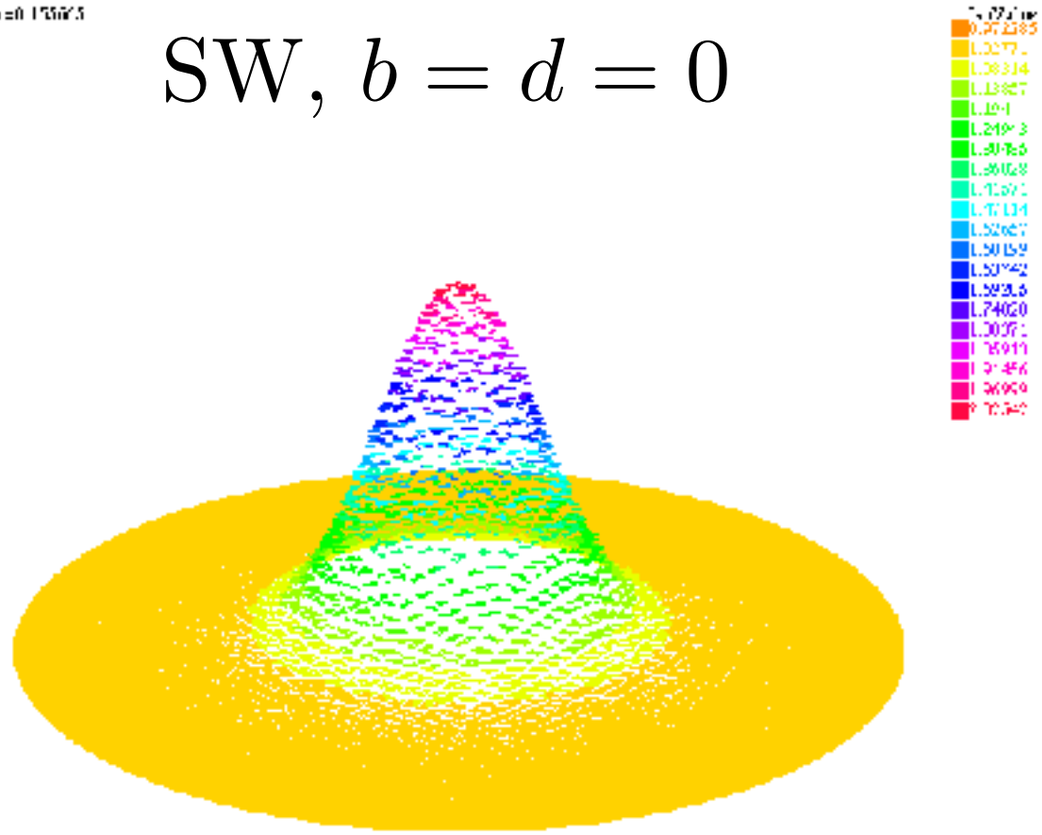


```
fespace Vh0(Th,P0);
Vh0 Bx=x,By=y;
real[int,int] PF(Bx[].n,2);
for(int i=0;i<Bx[].n;i++){
    PF(i,0)=Bx[](i);
    PF(i,1)=By[](i);
}
load "Curvature"
load "isoline"
int[int] ll=[1,2,3,4];
real[int,int] b12(1,3);
real l12=extractborder(Th,ll,b12);
border BB(t=0,1){ P=Curve(b12,t);label=2;}
mesh Th1=buildmesh(BB(b12.m-1),points=PF,nbvx=PF.n+Th.nbe);
fespace Vh1(Th1,P1);
Vh0 u0=x+y;
Vh1 u1;
u1=u0; // P0 to P1
u0=u1; // P1 to P0
```

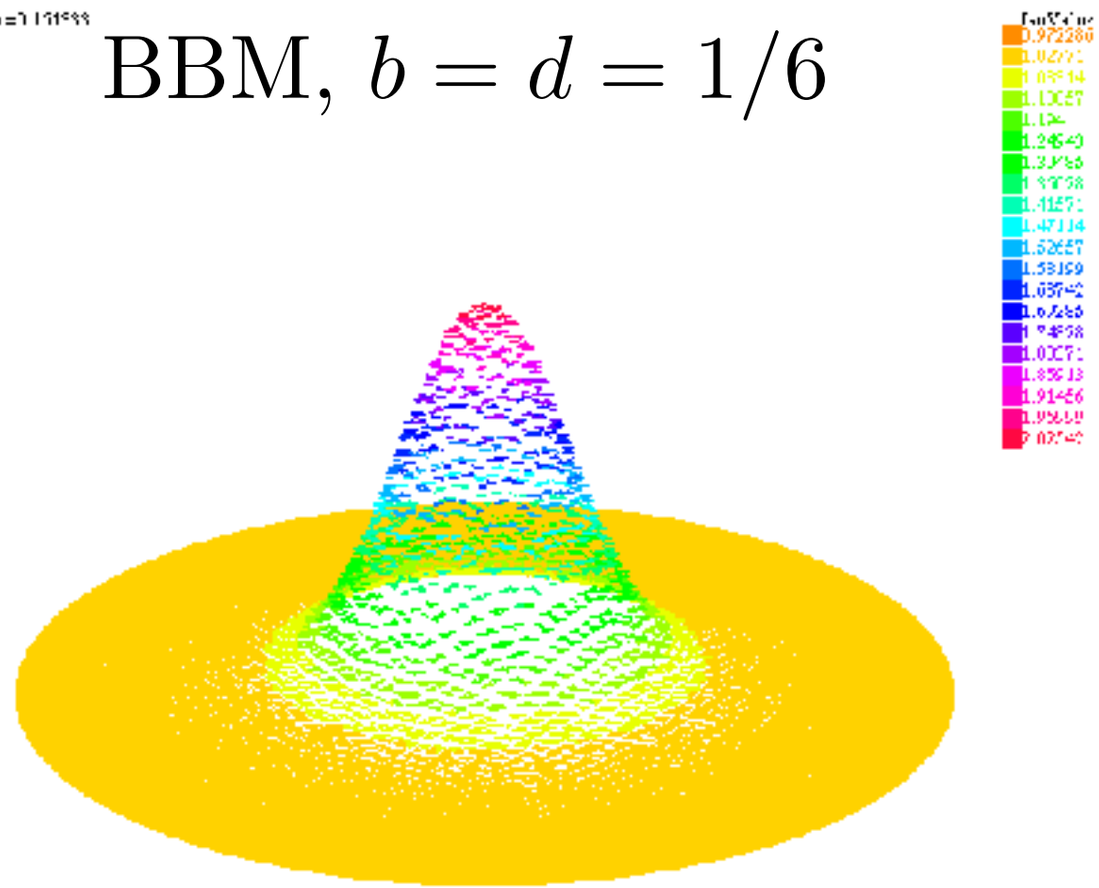
$$\eta_t + \nabla \cdot \mathcal{V} + \nabla \cdot (\eta \mathcal{V}) - b\Delta\eta_t = 0;$$

$$\mathcal{V}_t + \nabla\eta + \tfrac{1}{2}\nabla|\mathcal{V}|^2 - d\Delta\mathcal{V}_t = 0,$$

SW, $b = d = 0$



BBM, $b = d = 1/6$



Perspectives

- publish this work : (on fire)
  - hal-version : hal-03047531, v1
  - some codes will be in the next FreeFem++ version
- define in ff-c++, some fluxes, some limiters, ...
- Higher order scheme Δu_0 ...
- more validation ...
- 3D version ...

PUB !!!

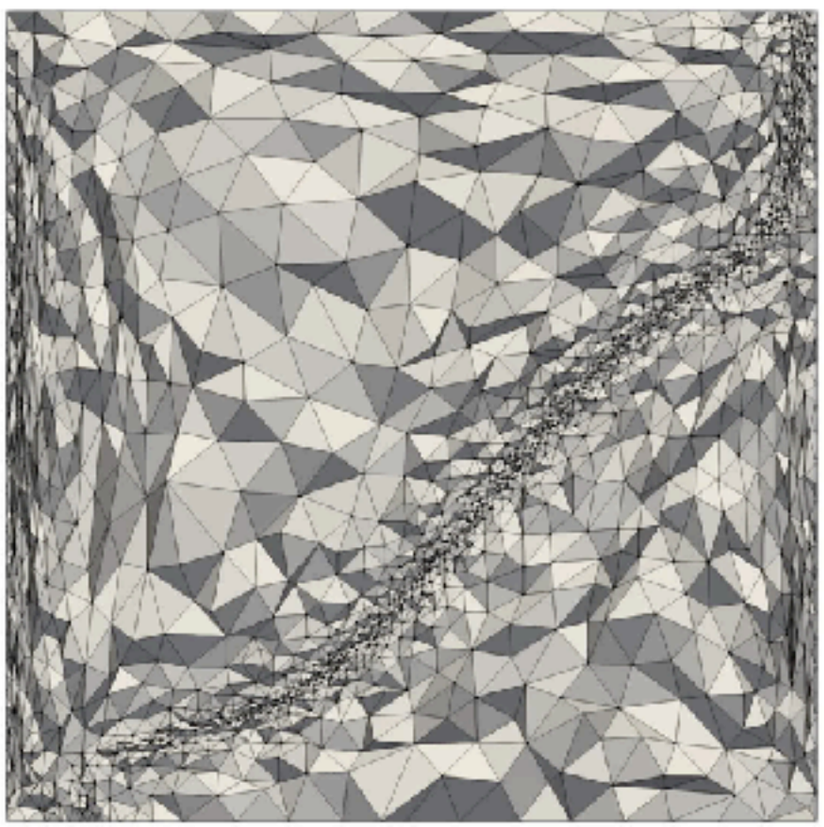
http://lmrs-num.math.cnrs.fr/2020CPCP2.html

Computer Physics Communications 257 (2020) 107492

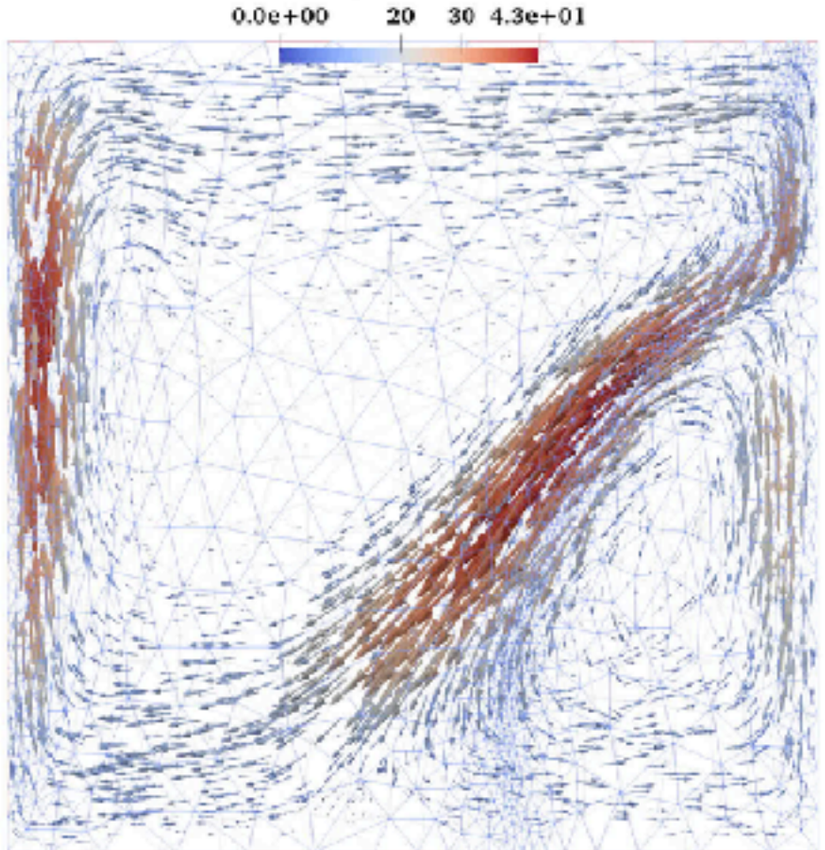
Parallel finite-element codes for the simulation of two-dimensional and three-dimensional solid–liquid phase-change systems with natural convection☆,☆☆

Georges Sadaka [a], Aina Rakotondrandisa [a], Pierre-Henri Tournier [b], Francky Luddens [a], Corentin Lothodé [a], Ionut Danaila [a,*]

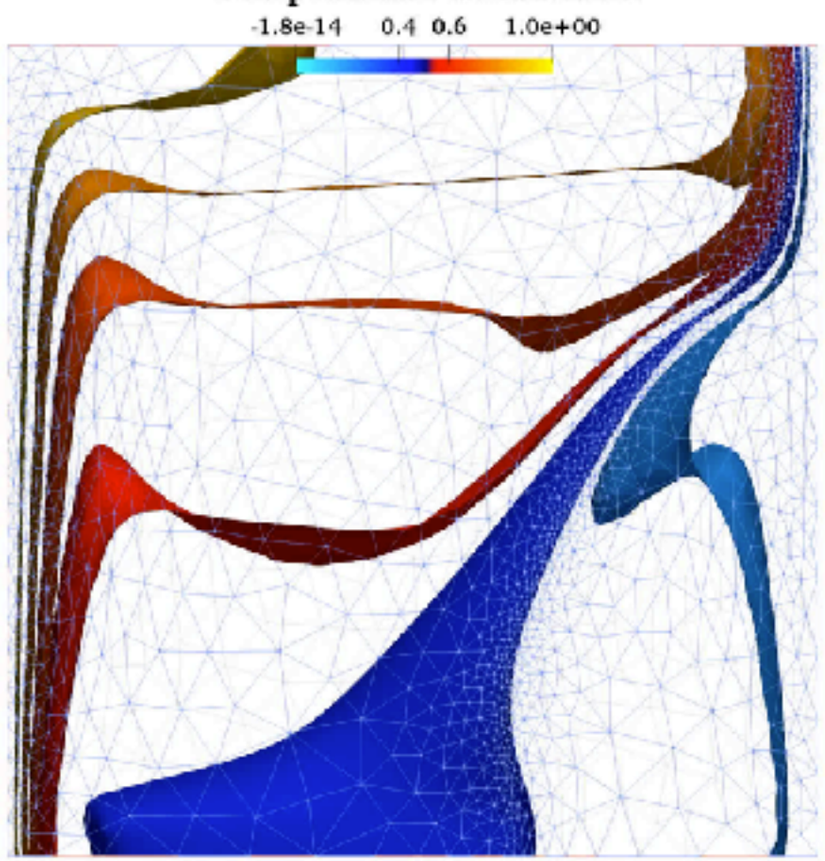
Slice of the mesh (central section)

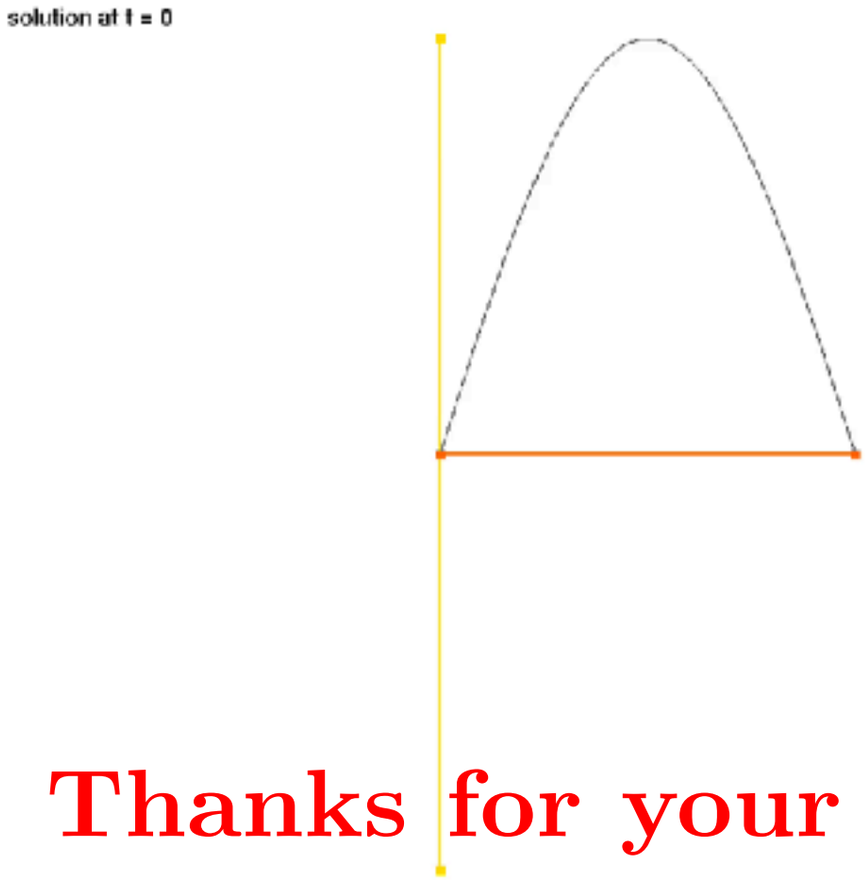


Velocity vector field



Temperature isosurfaces





Intermediate tutorial with FreeFem++ :

Wave equation in 1D, 2D, 3D

with adaptmesh, mmg3D, parmmg3D

DDM using PETSc

visualisation with medit and paraview

Thanks for your attention!

www.georges-sadaka.fr