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

Frédéric HECHT, Pierre JOLIVET and Georges SADAKA

www.georges-sadaka.fr

The TENTH tutorial and Workshop on FreeFem++ 12-13-14/12/18

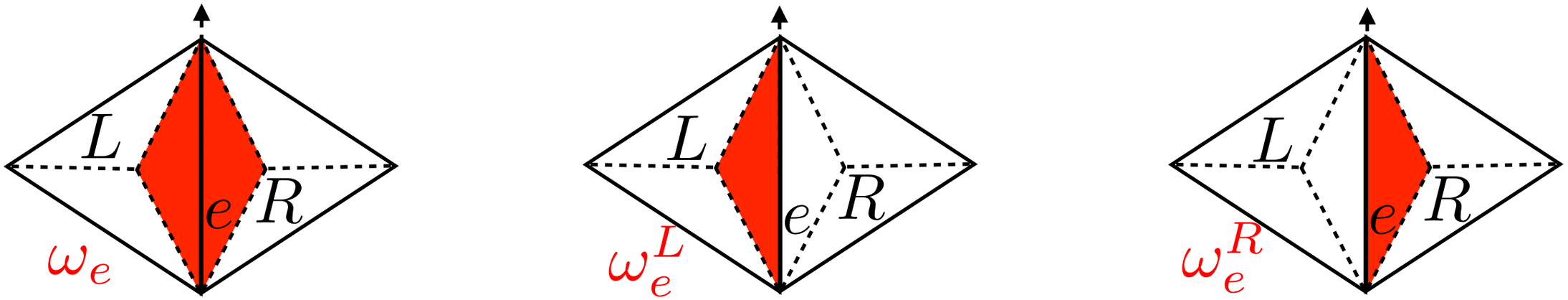
Outline

Outline

Thanks to : E. Audusse, A. Collin, V. Desveaux, F. Devuyst, D. Dutykh, P-H. Tournier and N. Seguin

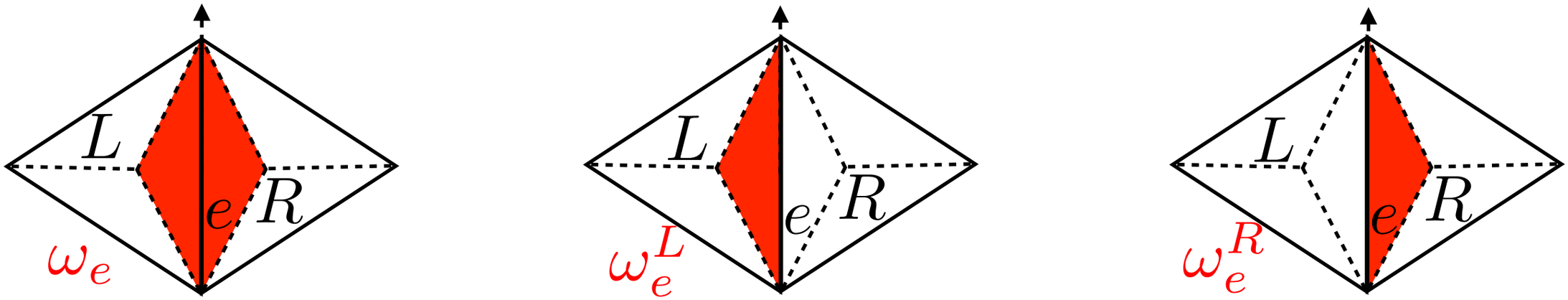
Intoduction

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.



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 :

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

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

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

- ⋆ `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.

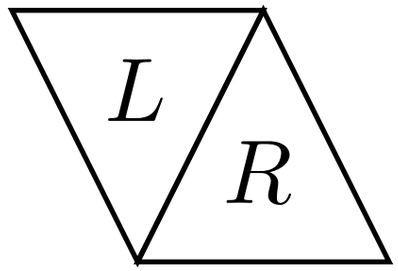
⋆ `area` gives the area of the current triangle.

⋆ `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

$\forall u, v \in V0_h$

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

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

Intoduction

$\forall u, v \in V0_h$

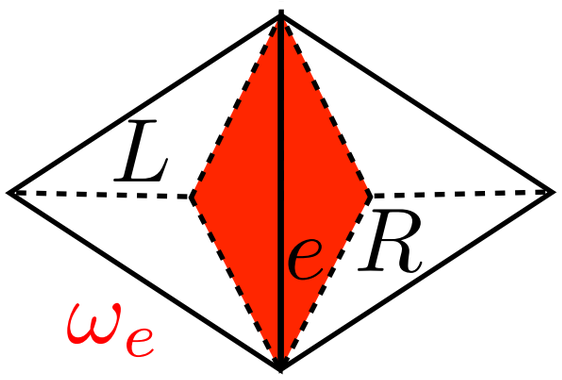
$$\texttt{intalledges(Th,qforder=1)(u*v)} = \sum_T \int_{\partial T} u \cdot v = \sum_T |\partial T| u,$$

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

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

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

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



Intoduction

$\forall u, v \in V0_h$

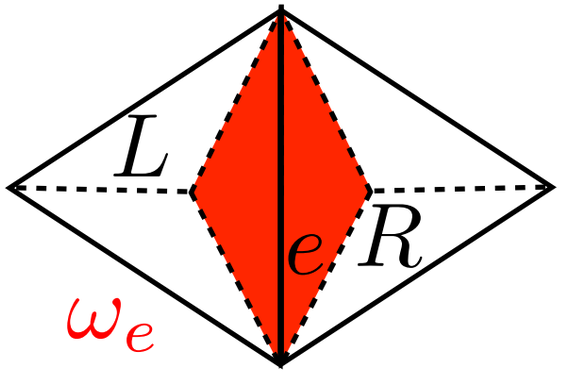
`intalledges(Th,qforder=1)(u*v)` $= \sum_T \int_{\partial T} u \cdot v = \sum_T |\partial T| u,$

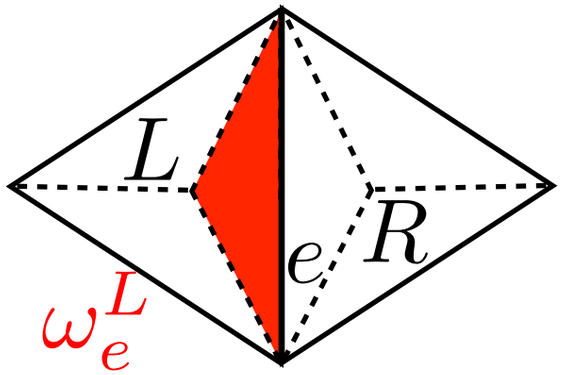
`intalledges(Th,qforder=1)(u*v*area/lenEdge)` $= \sum_T \int_{\partial T} \frac{|T| \cdot u \cdot v}{|\partial T|} = \sum_T |T| u,$

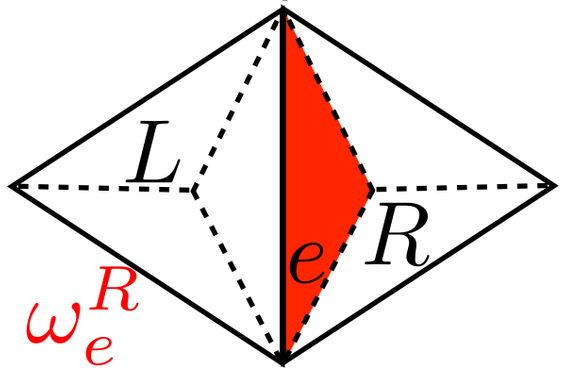
$\forall u \in V0_h, \forall v \in V0E_h$

`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 :

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

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

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

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

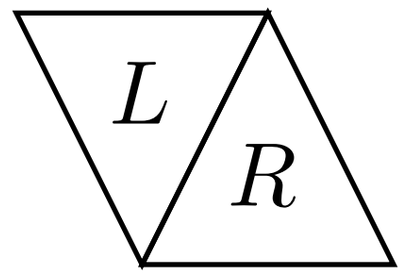
$$\mathcal{A}_{i,j} = \sum_T \int_{\partial T} \varepsilon_{edge} \frac{\varphi_i \psi_j}{|T||\partial T|}, \forall \varphi_i \in V0Edc_h, \forall \psi_j \in V0_h.$$

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

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

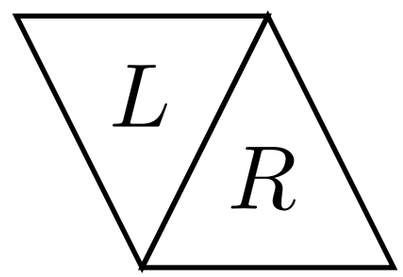
$$\text{B.C. : } \sum_{T_\Gamma} \int_{\partial T_\Gamma} \begin{cases} \left(\vec{\mathbf{V}} \cdot \vec{n}_{out}\right) u_h \cdot v_h & \text{if } \vec{\mathbf{V}} \cdot \vec{n}_{out} > 0 \\ 0 & \text{otherwise} \end{cases}$$



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

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

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

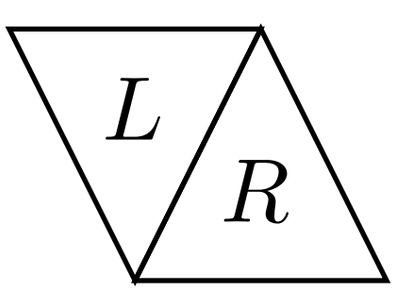



```
macro u0()(otherside(u))//
macro V10()(otherside(V1))//
macro V20()(otherside(V2))//
func NN=[N.x,N.y];
macro Upwind()([V1,V2]'*NN > 0. ? [V1,V2]'*NN*u : [V10,V20]'*NN*u0)//
macro UpwindBC()([V1,V2]'*NN > 0. ? [V1,V2]'*NN*u : 0.)//
varf vphiUpwind(unused,vh) = intalledges(Th,qforder=1)
(edgeOrientation*(InternalEdge*Upwind+BoundaryEdge*UpwindBC)*vh);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

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

$$\sum_T \int_{\partial T} \varepsilon_{edge} \left(\frac{1}{2} \left[\left(\vec{\mathbf{V}}_{|L} \cdot \vec{n}_{L\to R} \right) u_{h|L} + \left(\vec{\mathbf{V}}_{|R} \cdot \vec{n}_{L\to R} \right) u_{h|R} \right] - \frac{c_{Rus}}{2} [u_h] \right) v_h$$

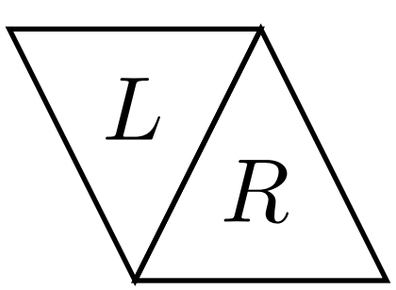
$$c_{Rus} = \max(|\lambda_L|, |\lambda_R|) = \max(|\vec{\mathbf{V}}_L \cdot \vec{n}_{L\to R}|, |\vec{\mathbf{V}}_R \cdot \vec{n}_{L\to R}|), [u_h] = u_{h|R} - u_{h|L}$$



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

$$\sum_T \int_{\partial T} \varepsilon_{edge} \left(\frac{1}{2} \left[\left(\vec{\mathbf{V}}_{|L} \cdot \vec{n}_{L\to R} \right) u_{h|L} + \left(\vec{\mathbf{V}}_{|R} \cdot \vec{n}_{L\to R} \right) u_{h|R} \right] - \frac{c_{Rus}}{2} [u_h] \right) v_h$$

$c_{Rus} = \max(|\lambda_L|, |\lambda_R|) = \max(|\vec{\mathbf{V}}_L \cdot \vec{n}_{L\to R}|, |\vec{\mathbf{V}}_R \cdot \vec{n}_{L\to R}|), [u_h] = u_{h|R} - u_{h|L}$



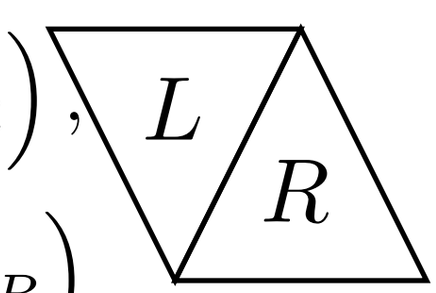

```
macro cRus()(max(abs([V1,V2]'*NN),abs([V10,V20]'*NN)))//
macro Rusanov()(.5*([V1,V2]'*NN*u+[V10,V20]'*NN*u0-cRus*jump(u)))//
macro RusanovBC()([V1,V2]'*NN > 0. ? [V1,V2]'*NN*u : 0.)//
varf vphiRusanov(unused,vh) = intalledges(Th,qforder=1)
(edgeOrientation*(InternalEdge*Rusanov+BoundaryEdge*RusanovBC)*vh);
phi[] = vphiRusanov(0,V0Edch);
```

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

$$\sum_T \int_{\partial T} \varepsilon_{edge} \begin{cases} \left(\vec{\mathbf{V}}_{|L} \cdot \vec{n}_{L\to R}\right) u_{h|L} v_h, & s_L \geq 0 \\ \left(\vec{\mathbf{V}}_{|R} \cdot \vec{n}_{L\to R}\right) u_{h|R} v_h, & s_R \leq 0 \\ \dfrac{s_R \left(\vec{\mathbf{V}}_{|L} \cdot \vec{n}_{L\to R}\right) u_{h|L} - s_L \left(\vec{\mathbf{V}}_{|R} \cdot \vec{n}_{L\to R}\right) u_{h|R} + s_L s_R [u_h]}{s_R - s_L} v_h, & s_L < 0 < s_R \end{cases}$$

$$s_L = \min(\lambda_L, \lambda_R) = \min\left(\vec{\mathbf{V}}_{|L} \cdot \vec{n}_{L\to R}, \vec{\mathbf{V}}_{|R} \cdot \vec{n}_{L\to R}\right),$$

$$s_R = \max(\lambda_L, \lambda_R) = \max\left(\vec{\mathbf{V}}_{|L} \cdot \vec{n}_{L\to R}, \vec{\mathbf{V}}_{|R} \cdot \vec{n}_{L\to R}\right)$$

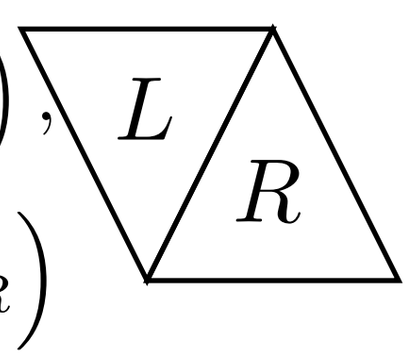


HLL flux : $\forall u_h \in V0_h, \forall v_h \in V0Edc_h, \phi_{num}(u_h, v_h) =$

$$\sum_T \int_{\partial T} \varepsilon_{edge} \begin{cases} \left(\vec{\mathbf{V}}_{|L} \cdot \vec{n}_{L\to R}\right) u_{h|L} v_h, & s_L \geq 0 \\ \left(\vec{\mathbf{V}}_{|R} \cdot \vec{n}_{L\to R}\right) u_{h|R} v_h, & s_R \leq 0 \\ \dfrac{s_R \left(\vec{\mathbf{V}}_{|L} \cdot \vec{n}_{L\to R}\right) u_{h|L} - s_L \left(\vec{\mathbf{V}}_{|R} \cdot \vec{n}_{L\to R}\right) u_{h|R} + s_L s_R [u_h]}{s_R - s_L} v_h, & s_L < 0 < s_R \end{cases}$$

$$s_L = \min(\lambda_L, \lambda_R) = \min\left(\vec{\mathbf{V}}_{|L} \cdot \vec{n}_{L\to R}, \vec{\mathbf{V}}_{|R} \cdot \vec{n}_{L\to R}\right),$$

$$s_R = \max(\lambda_L, \lambda_R) = \max\left(\vec{\mathbf{V}}_{|L} \cdot \vec{n}_{L\to R}, \vec{\mathbf{V}}_{|R} \cdot \vec{n}_{L\to R}\right)$$




```
macro sL()(min([V1,V2]'*NN,[V10,V20]'*NN))//
macro sR()(max([V1,V2]'*NN,[V10,V20]'*NN))//
macro invdiffnp(a,b)((a<0. & b>0.) ? 1./max(b-a,1.e-30) : 0. )//
macro HLL()( [V1,V2]'*NN*u*(sL>=0) + [V10,V20]'*NN*u0*(sR<=0) + (sR*[V1,V2]'*NN*u
        -sL*[V10,V20]'*NN*u0+sL*sR*jump(u))*invdiffnp(sL,sR)*(sL<0.)*(sR>0.) ) )//
macro HLLBC()([V1,V2]'*NN > 0. ? [V1,V2]'*NN*u : 0.)//
varf vphiHLL(unused,vh) = intalledges(Th,qforder=1)
(edgeOrientation*(InternalEdge*HLL+BoundaryEdge*HLLBC)*vh);
phi[] = vphiHLL(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, $\forall T, \forall u_h, v_h \in V0_h$:

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

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;
```

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;
```

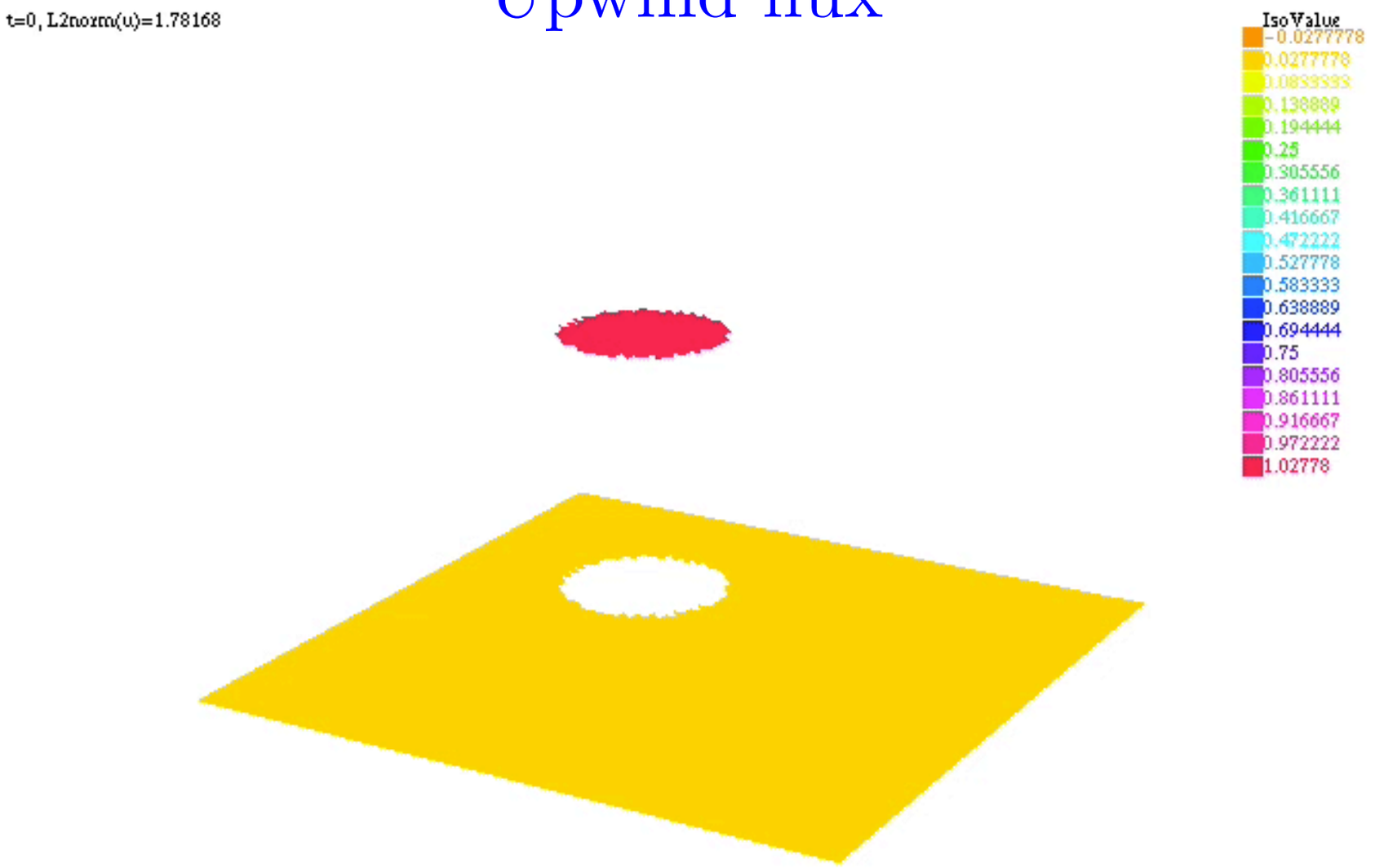
Explicit Euler scheme : $u_h^{n+1} = u_h^n - \delta t \mathcal{A} \phi_{num}(u_h^n, v_h)$

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

Advection equation

```
real x0=-1.5,y0=-1.5,R=4.;
V0h u=dist(x-x0,y-y0)<=(R/4.),V1=1.,V2=1.;
```

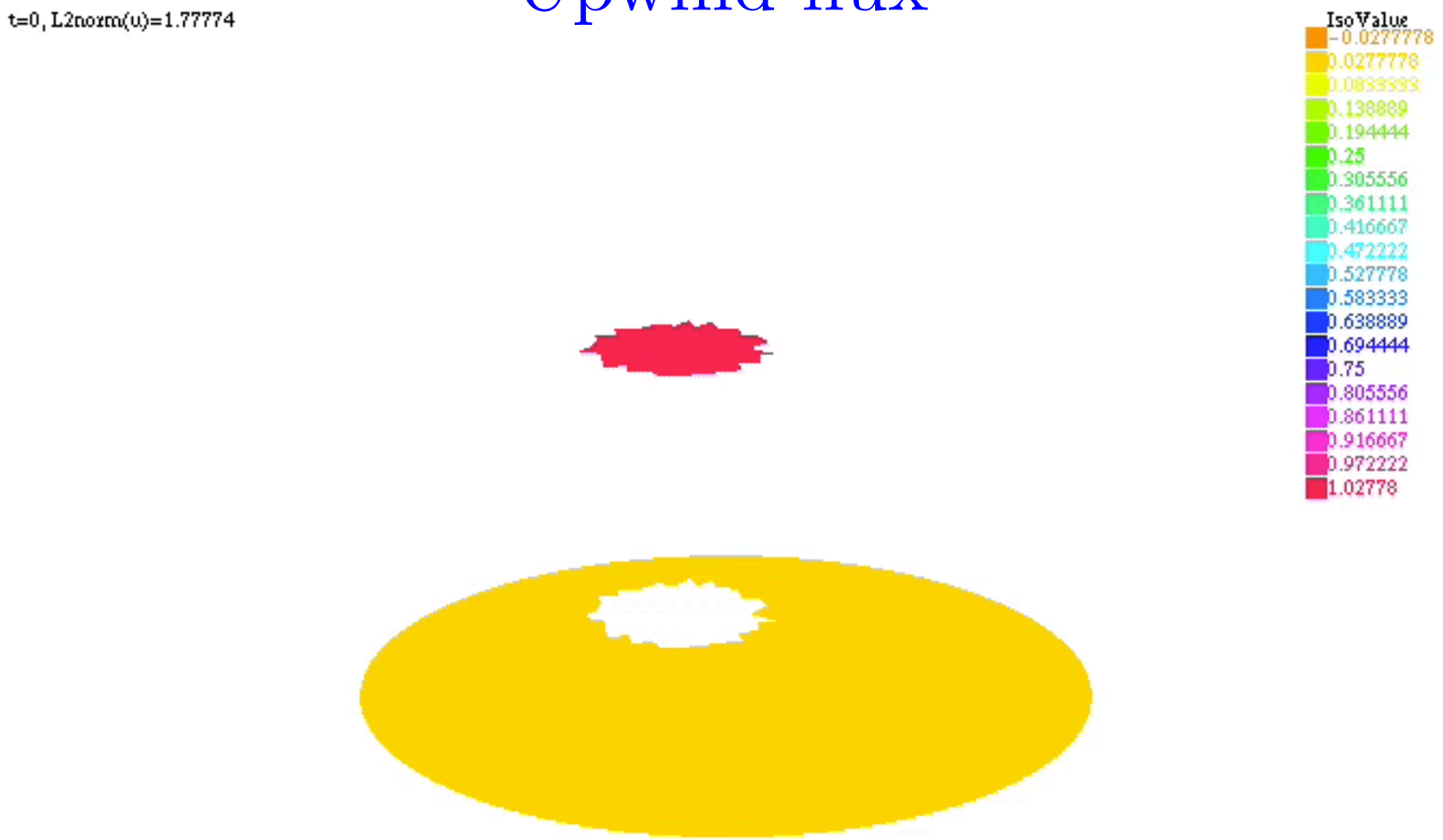
Upwind flux



http://www.georges-sadaka.fr/movies/FreeVol_Transport_Upwind_1_1.m4v

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

Upwind flux



http://www.georges-sadaka.fr/movies/FreeVol_Transport_Upwind_y_-x.m4v

```
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[];
}
```

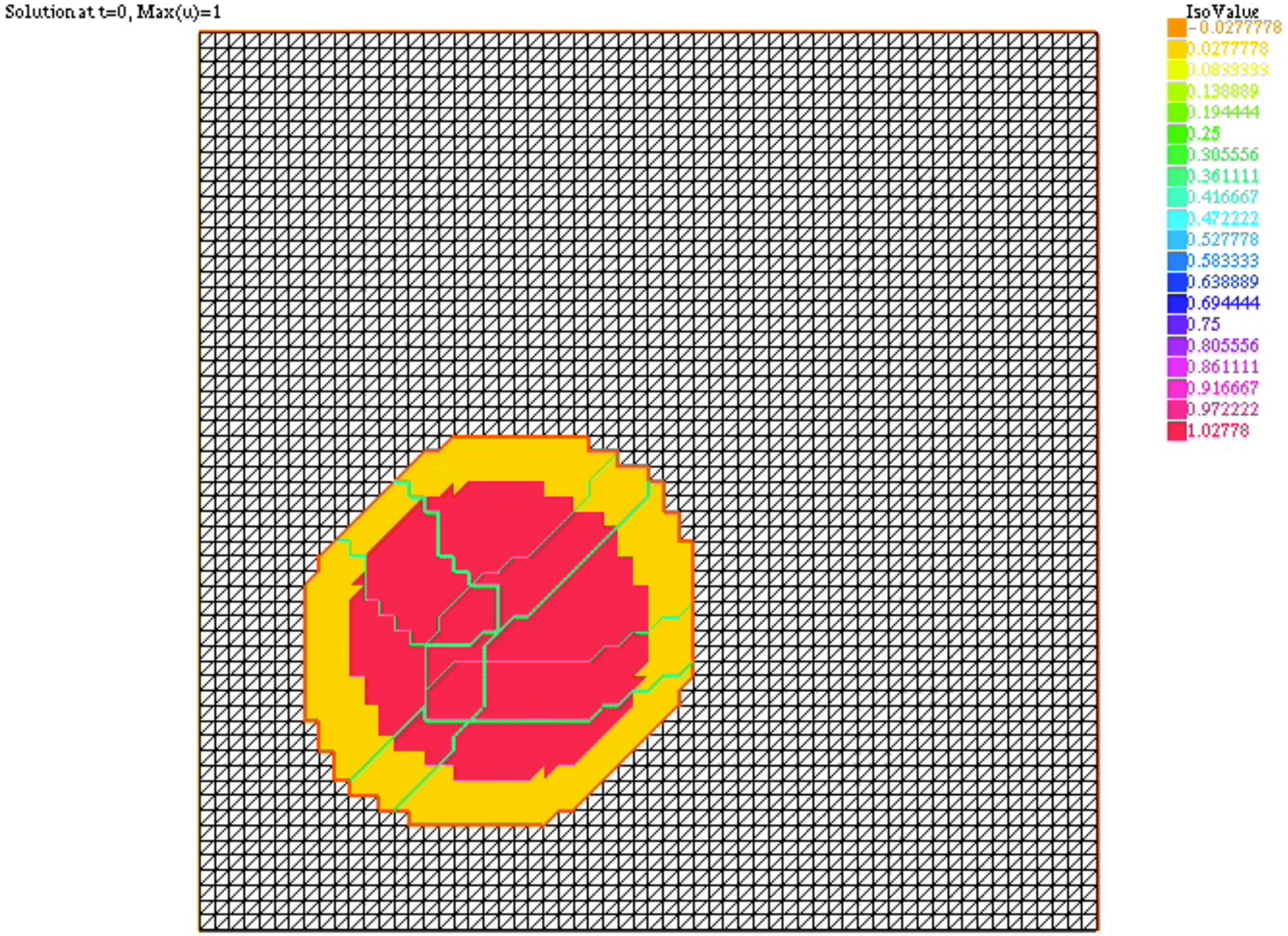
```
load "Element_PkEdge"
real x0=-1.,y0=-1.,CFL=.9,R=4.,TF=4.;
mesh ThGlobal=square(30,30,[x*6.-3.,y*6.-3.]);
fespace V0Edch(Th,P0edgedc);
fespace V0h(Th,P0);
V0h u=dist(x-x0,y-y0)<=(R/4.),up,V1=1.,V2=1.;
V0Edch phi;
varf vA(phii,psij)=intalledges(Th,qforder=1)(edgeOrientation*phii*psij/lenEdge/area);
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[];
}
```

```
include "paralmeshexchange.edp"
```

```
mpiAllReduce(dt,mdt,mpiCommWorld,mpiMIN);
exchange(A,up[],scaled=true);
```

Upwind flux

V1=y,V2=−x;



http://www.georges-sadaka.fr/movies/FreeVol_Transport_Upwind_y_-x_adapt_GS.m4v

Outline

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:

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

hydrostatic reconstruction

$$z_{b_{LR}} = z_{b_{RL}} = \max(z_{b_L}, z_{b_R})$$

$$H^r_{LR} = \begin{cases} \max(H_L + z_{b_L} - z_{b_{LR}}, 0) & \text{on internal edge} \\ \max(H_L, 0) & \text{on boundary edge} \end{cases}$$

$$H^r_{RL} = \begin{cases} \max(H_R + z_{b_R} - z_{b_{RL}}, 0) & \text{on internal edge} \\ \max(H_R, 0) & \text{on boundary edge} \end{cases}$$

```
macro zb0()(otherside(zb))//
macro Hr()(max(H+zb-max(zb,zb0),0.))// Hr_LR
macro HrB()(max(H,0))// Hr_LR on the boundary
macro Hr0()(otherside(H))// Hr_RL
```

hydrostatic reconstruction

$$z_{b_{LR}} = z_{b_{RL}} = \max(z_{b_L}, z_{b_R})$$

$$H^r_{LR} = \begin{cases} \max(H_L + z_{b_L} - z_{b_{LR}}, 0) & \text{on internal edge} \\ \max(H_L, 0) & \text{on boundary edge} \end{cases}$$

$$H^r_{RL} = \begin{cases} \max(H_R + z_{b_R} - z_{b_{RL}}, 0) & \text{on internal edge} \\ \max(H_R, 0) & \text{on boundary edge} \end{cases}$$

```
macro zb0()(otherside(zb))//
macro Hr()(max(H+zb-max(zb,zb0),0.))// Hr_LR
macro HrB()(max(H,0))// Hr_LR on the boundary
macro Hr0()(otherside(H))// Hr_RL
```

$$\mathbf{U}^r = \begin{pmatrix} H^r \\ H^r u \\ H^r v \end{pmatrix}, F(\mathbf{U}^r) = \begin{pmatrix} H^r u \\ H^r u^2 + \frac{g}{2} H^{r^2} \\ H^r uv \end{pmatrix}, G(\mathbf{U}^r) = \begin{pmatrix} H^r v \\ H^r uv \\ H^r v^2 + \frac{g}{2} H^{r^2} \end{pmatrix}.$$

```
macro F()[Hr*u,Hr*u^2+.5*gravity*Hr^2,Hr*u*v]//
macro G()[Hr*v,Hr*v*u,Hr*v^2+.5*gravity*Hr^2]//
macro F0()[Hr0*u0,Hr0*u0^2+.5*gravity*Hr0^2,Hr0*u0*v0]//
macro G0()[Hr0*v0,Hr0*v0*u0,Hr0*v0^2+.5*gravity*Hr0^2]//
macro FB()[HrB*u,HrB*u^2+.5*gravity*HrB^2,HrB*u*v]//
macro GB()[HrB*v,HrB*v*u,HrB*v^2+.5*gravity*HrB^2]//
```

Well-balanced Shallow water system :

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

Rusanov flux : $\forall \mathbf{U}_h^r \in V03_h, \forall \mathbf{V}_h \in V0Edc3_h, \phi_{num}(\mathbf{U}_h^r, \mathbf{V}_h) =$

$$\sum_T \int_{\partial T} \varepsilon_{edge} \left(\frac{1}{2} \left(\left[F(\mathbf{U}_{h|L}^r), G(\mathbf{U}_{h|L}^r) \right] \cdot \vec{n}_{L \to R} + \left[F(\mathbf{U}_{h|R}^r), G(\mathbf{U}_{h|R}^r) \right] \cdot \vec{n}_{L \to R} \right) - \frac{c}{2} [\mathbf{U}_h^r] \right) \mathbf{V}_h$$

$$c = \max_{\lambda \in \{\lambda_1, \lambda_2, \lambda_3\}} (|\lambda_L|, |\lambda_R|), |\lambda_{1|L}| = \left| [u_L, v_L] \cdot \vec{n}_{L \to R} - \sqrt{g H_L^r} \right|$$

```
macro lambda1()([u,v]'*NN-sqrt(gravity*Hr))//
macro cRus()(max(max(abs(lambda1),abs(lambda3)),max(abs(lambda10),abs(lambda30))))//
macro Source()[0.,-.5*gravity*(Hr^2-H^2)*N.x,-.5*gravity*(Hr^2-H^2)*N.y]//
macro Rusanov()(.5*([F,G]'*NN+[F0,G0]'*NN-cRus*[jump(Hr),jump(Hr*u),jump(Hr*v)]))//
macro NBC()([FB,GB]'*NN)//
macro WallBC()[0.,(sqrt(gravity*HrB)*HrB*[u,v]'*NN+.5*gravity*HrB^2)*N.x,
(sqrt(gravity*HrB)*HrB*[u,v]'*NN+.5*gravity*HrB^2)*N.y]//
varf vphiRusanov([unH,unHu,unHv],[vH,vHu,vHv]) = intalledges(Th,qforder=1)
(edgeOrientation*(InternalEdge*(Rusanov+Source)+BoundaryEdge*NBC)'*[vH,vHu,vHv]);
```

Malpasset Dambreak (December 2 1959)

SW2D Malpasset Dambreak: 0 s

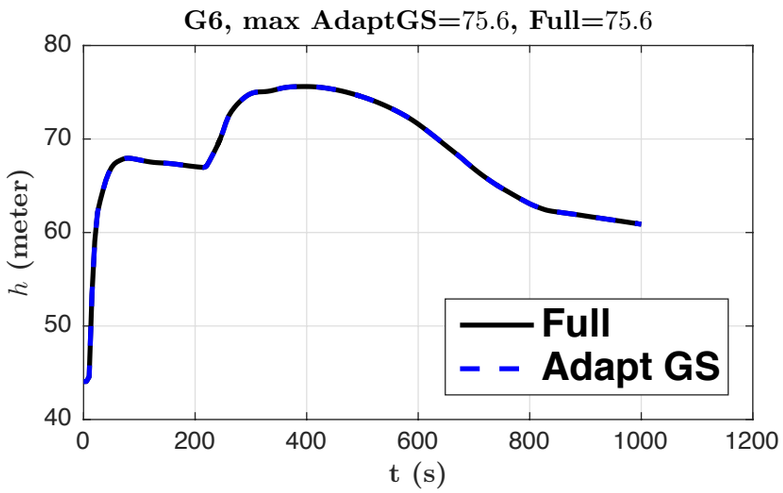
FreeVol++
Run on 16 cores
HLL scheme
no manning friction
first order
CFL=0.9
Ntri=33170

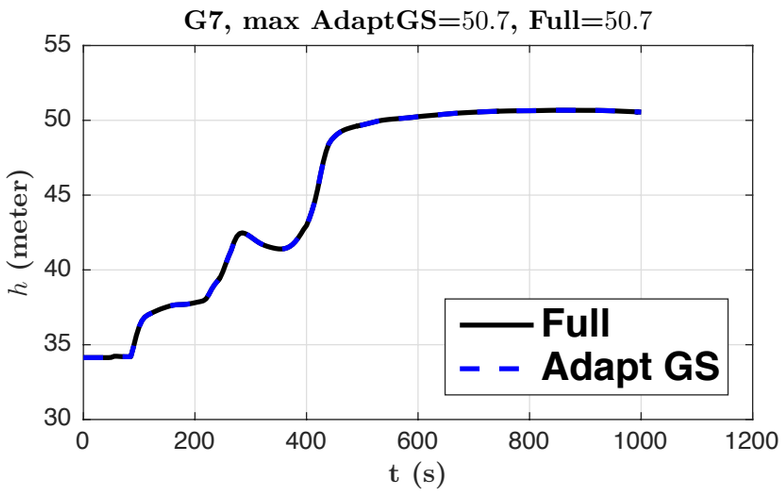
cpu time :
Adapt GS: 06:50:48
Full : 09:37:39
ratio=1.4

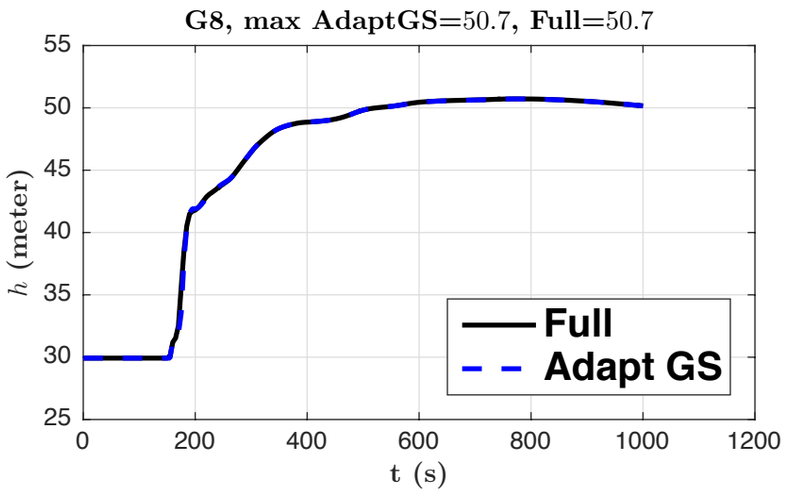
Authors : Frédéric Hecht, Pierre Jolivet and Georges Sadaka : 17/03/2018

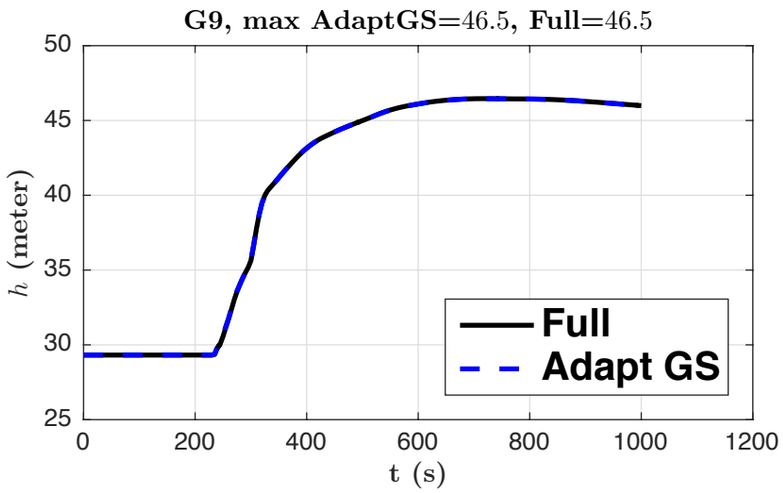
http://www.georges-sadaka.fr/movies/SW2D_Malpasset_FreeVol_O1_np_16.mov

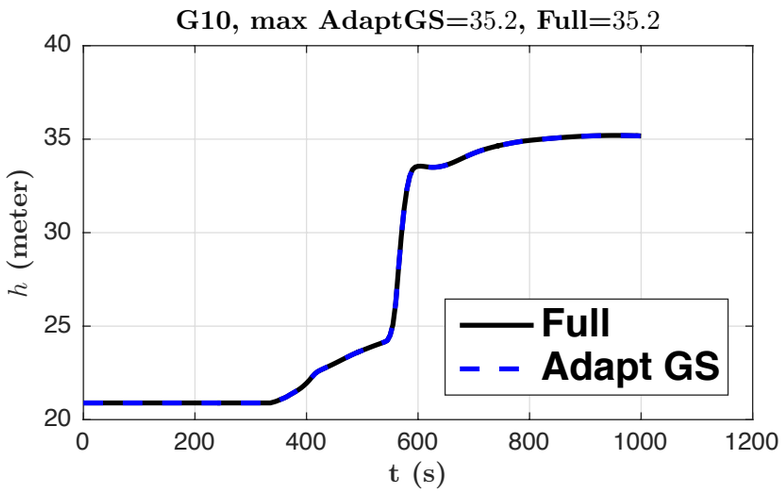
Malpasset Dambreak (December 2 1959)

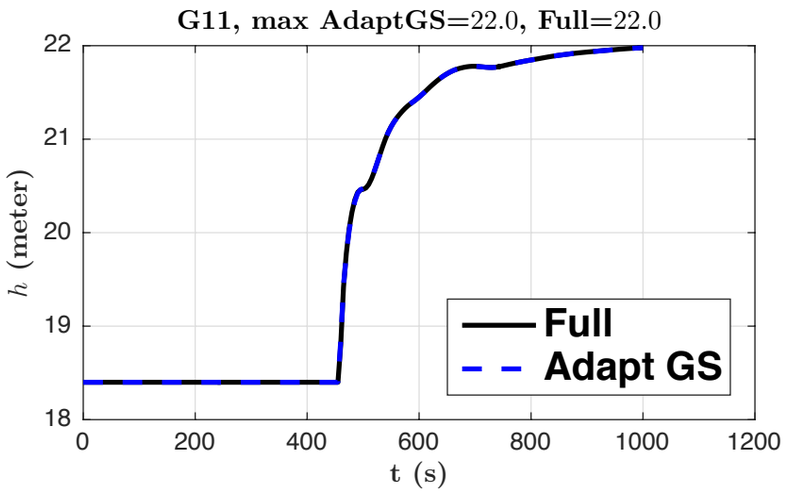


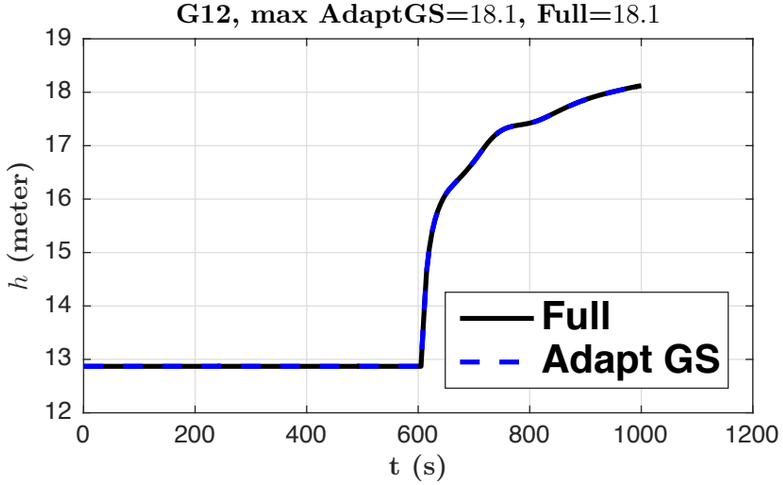


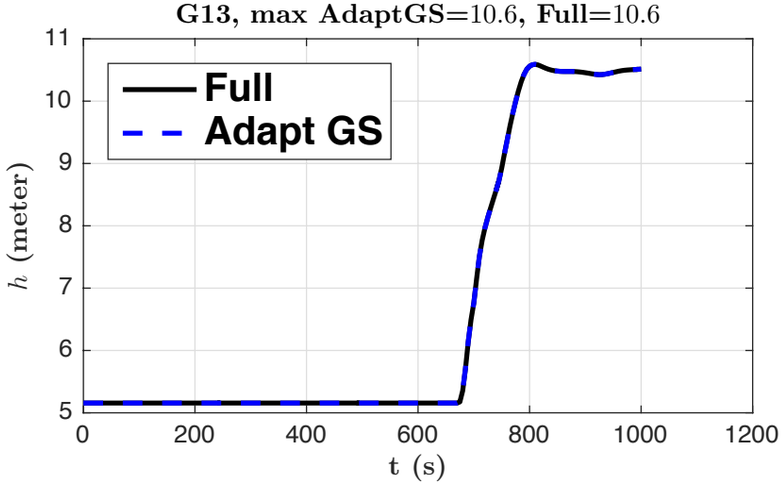


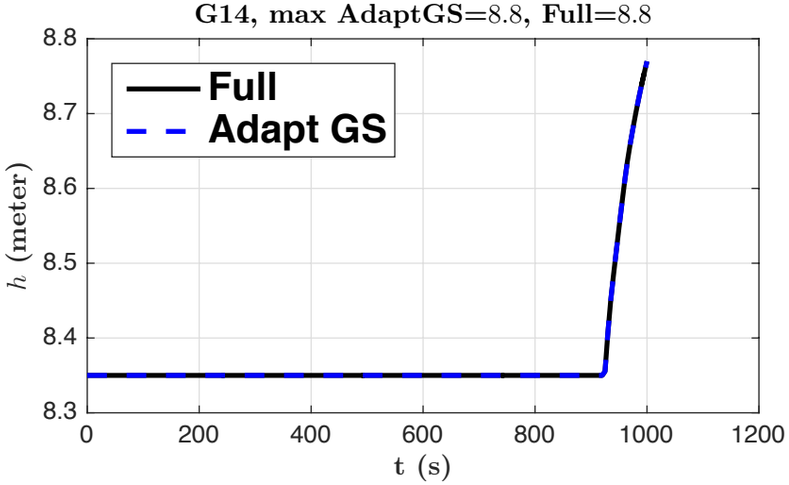




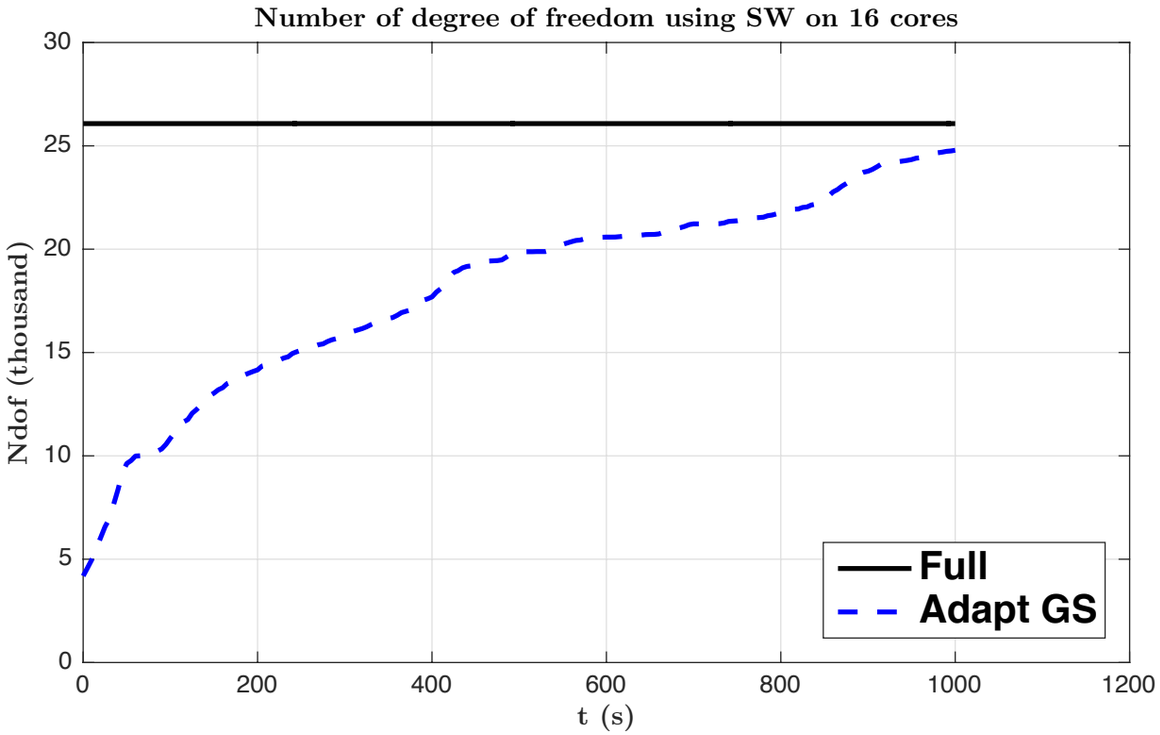


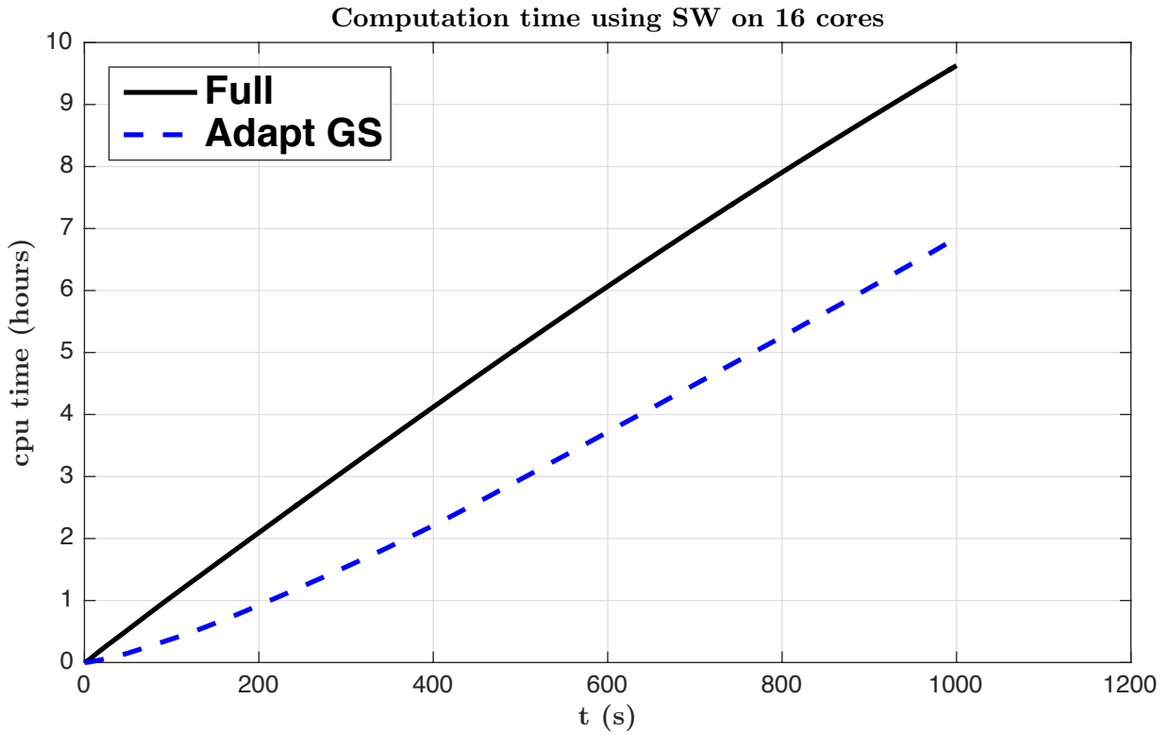




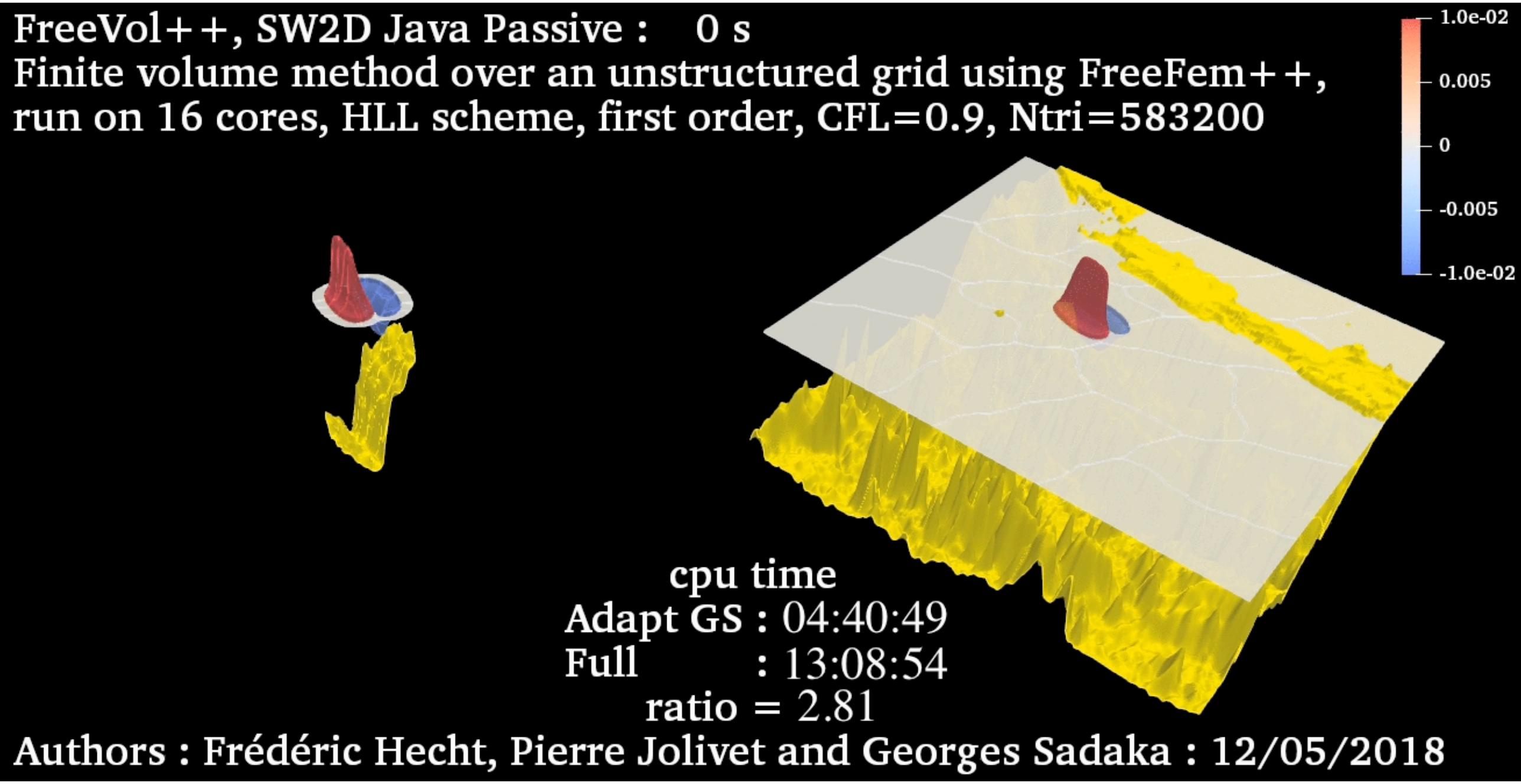


Malpasset Dambreak (December 2 1959)



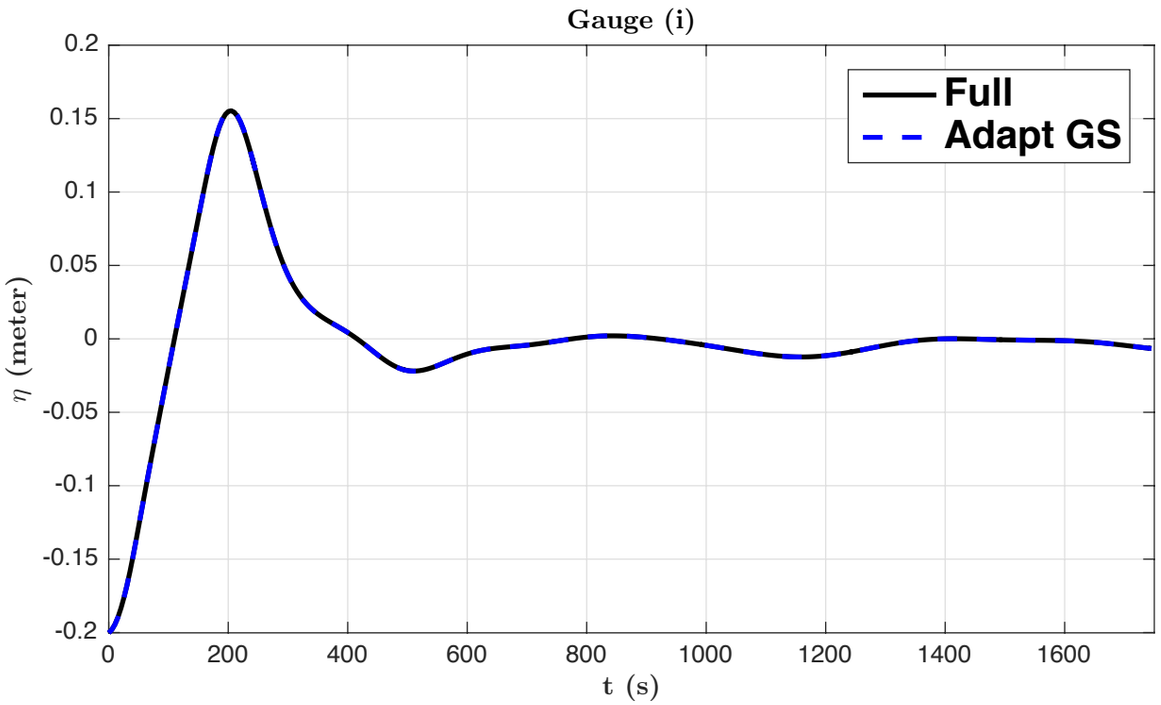


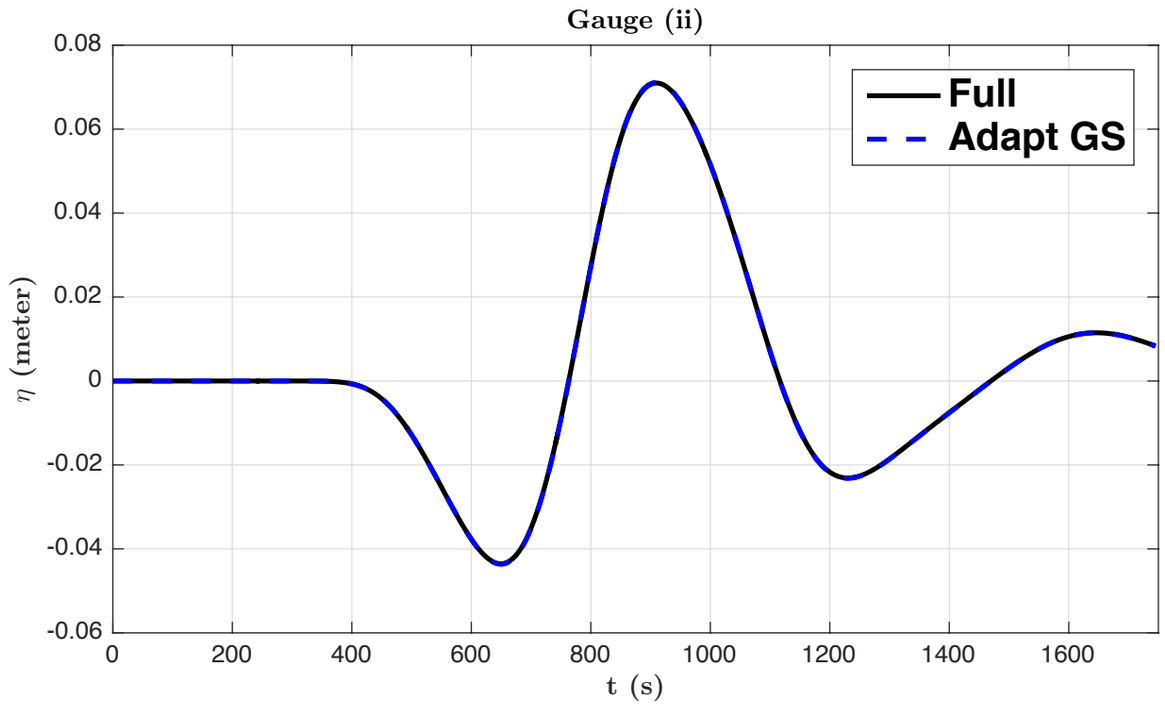
Propagation of a *Tsunami* wave near Java (July, 2006)

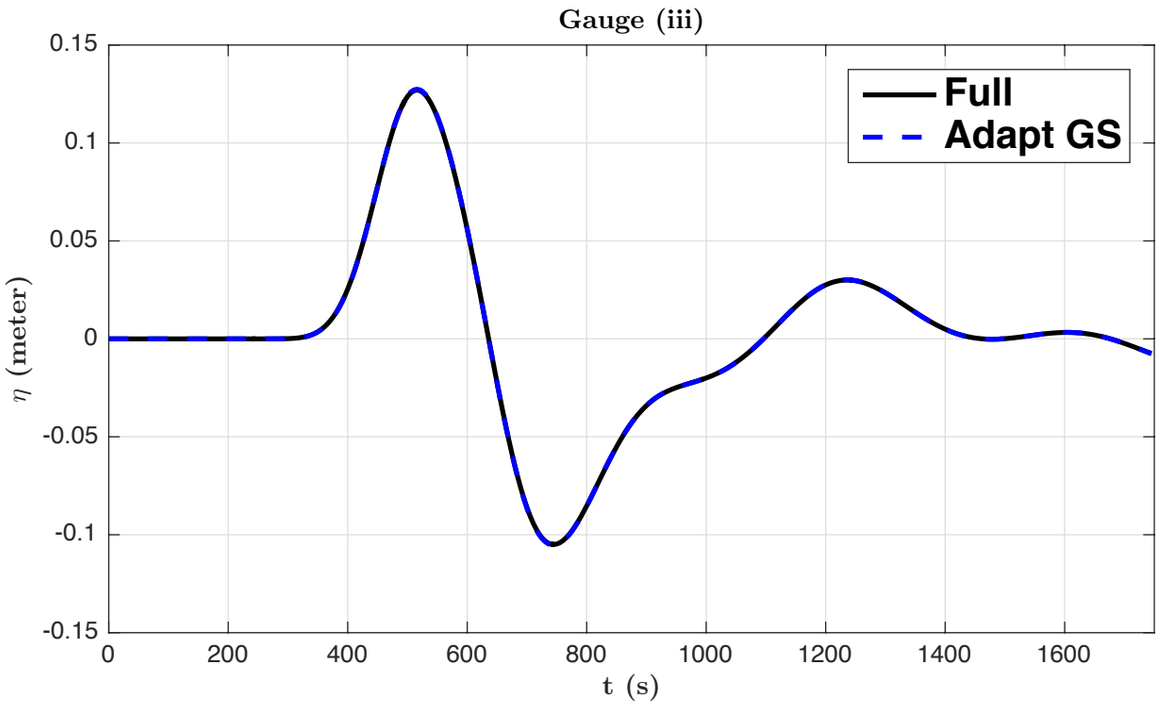


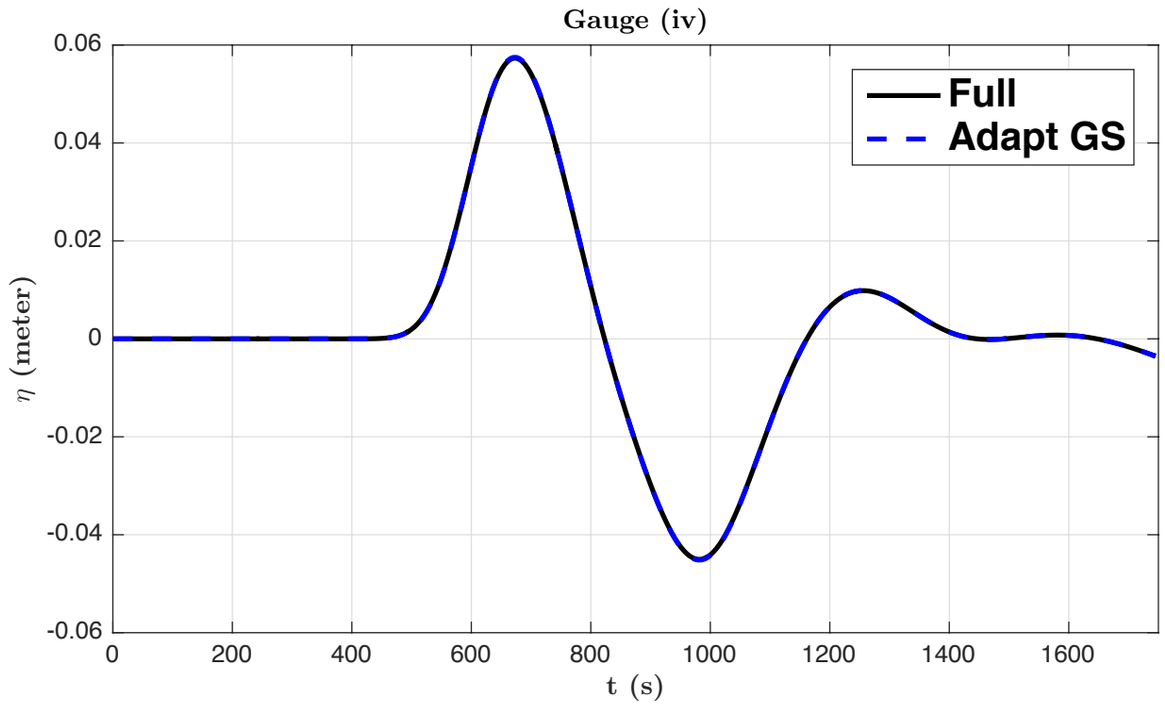
http://www.georges-sadaka.fr/movies/SW2D_Java_Passive_FreeVol_O1_np_16.mov

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

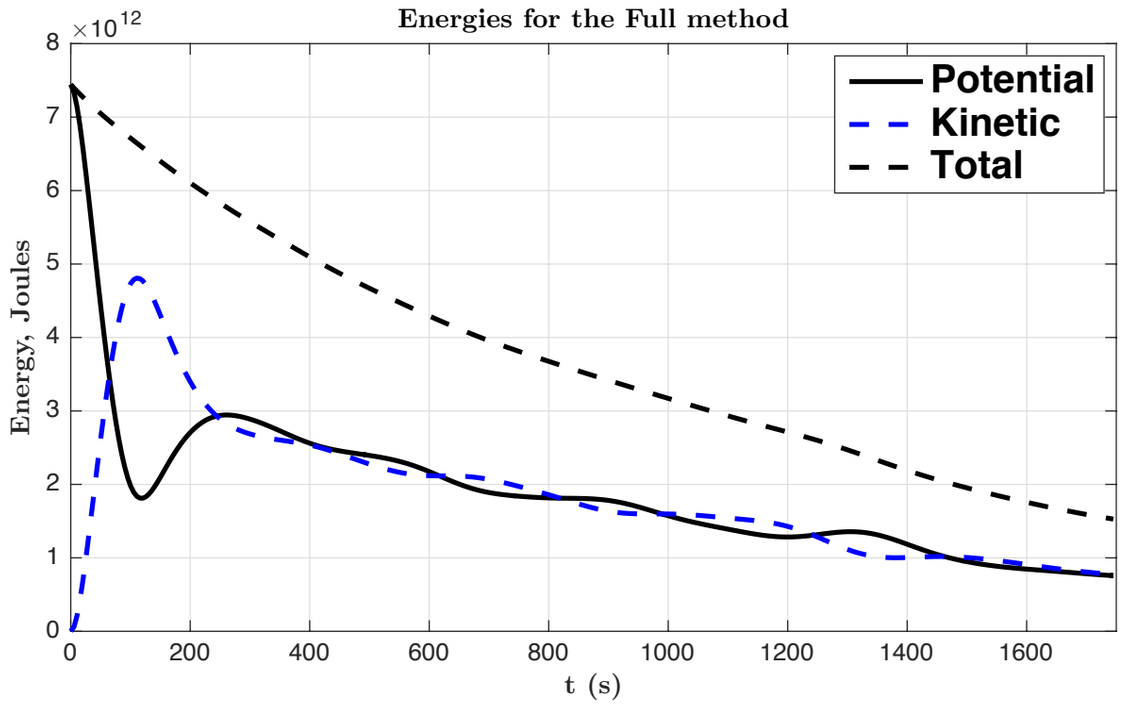


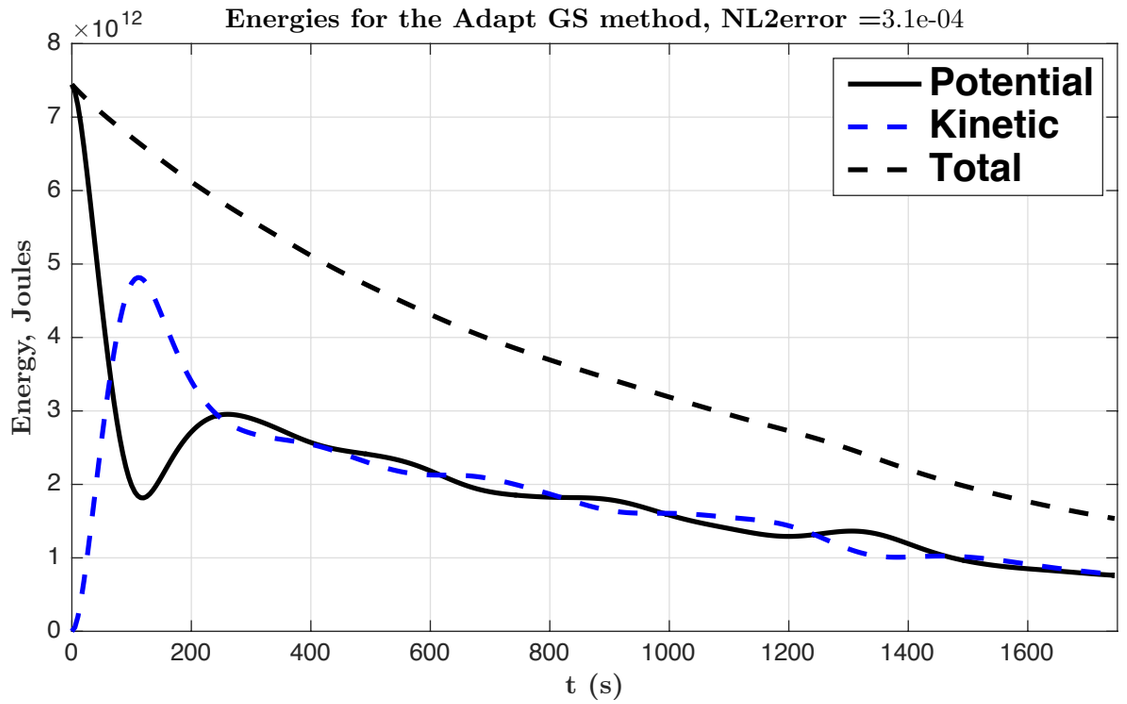


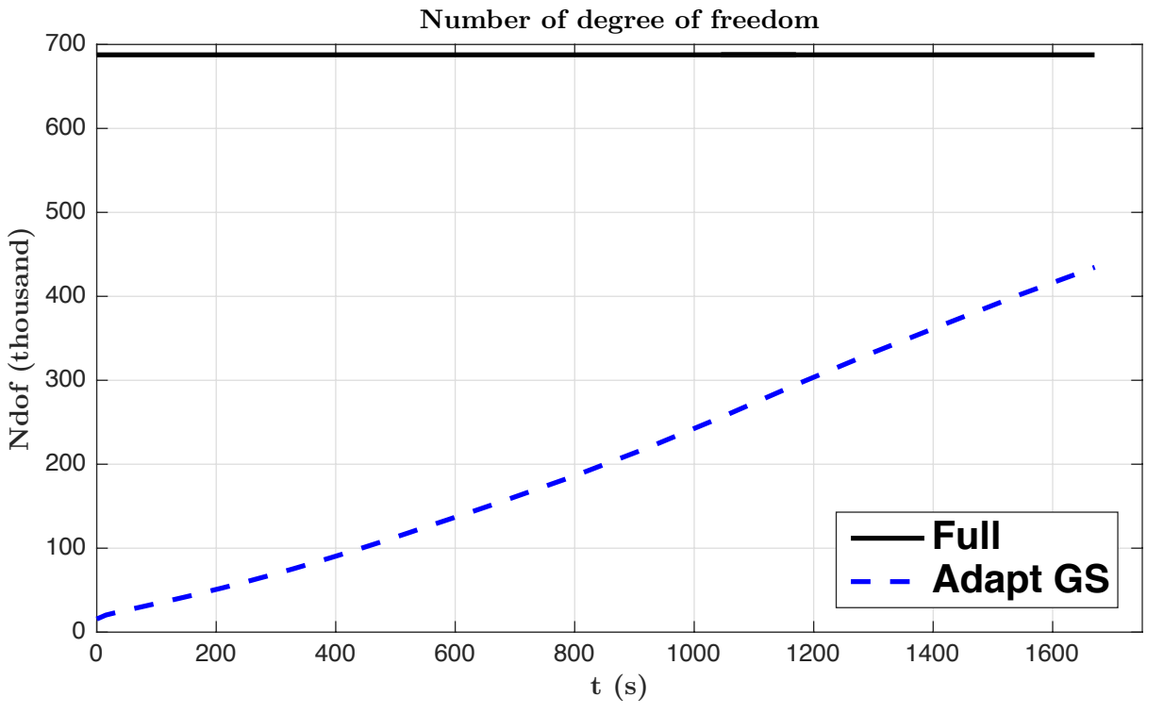


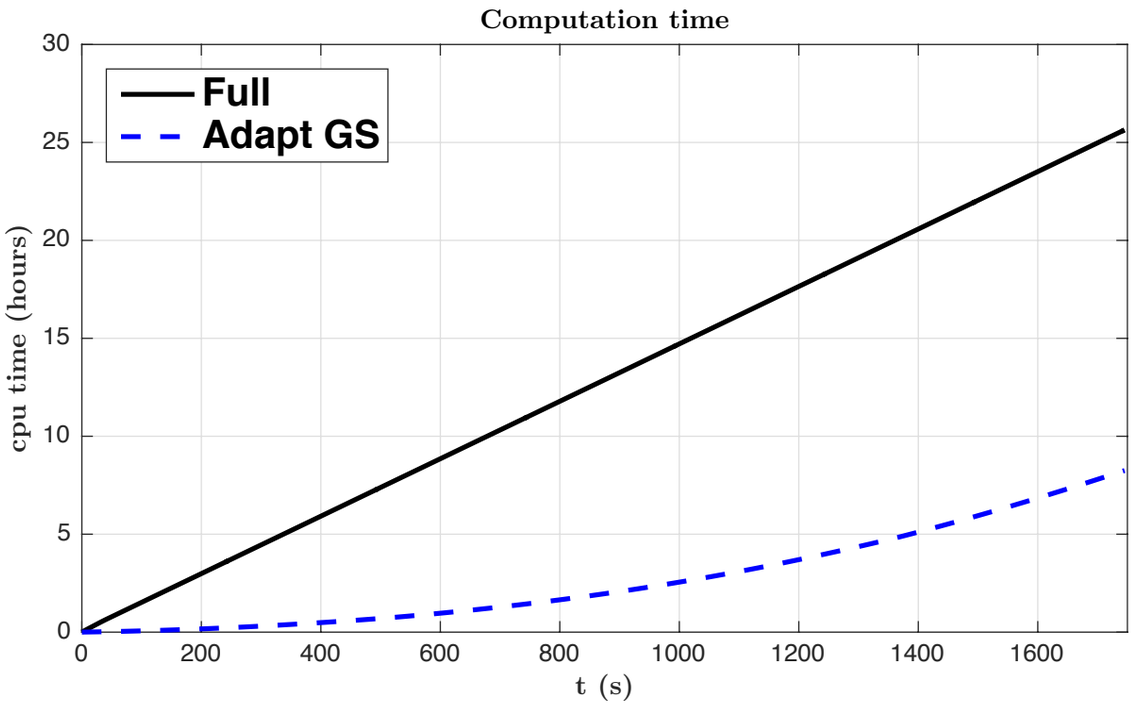


Propagation of a *Tsunami* wave near Java (July, 2006)





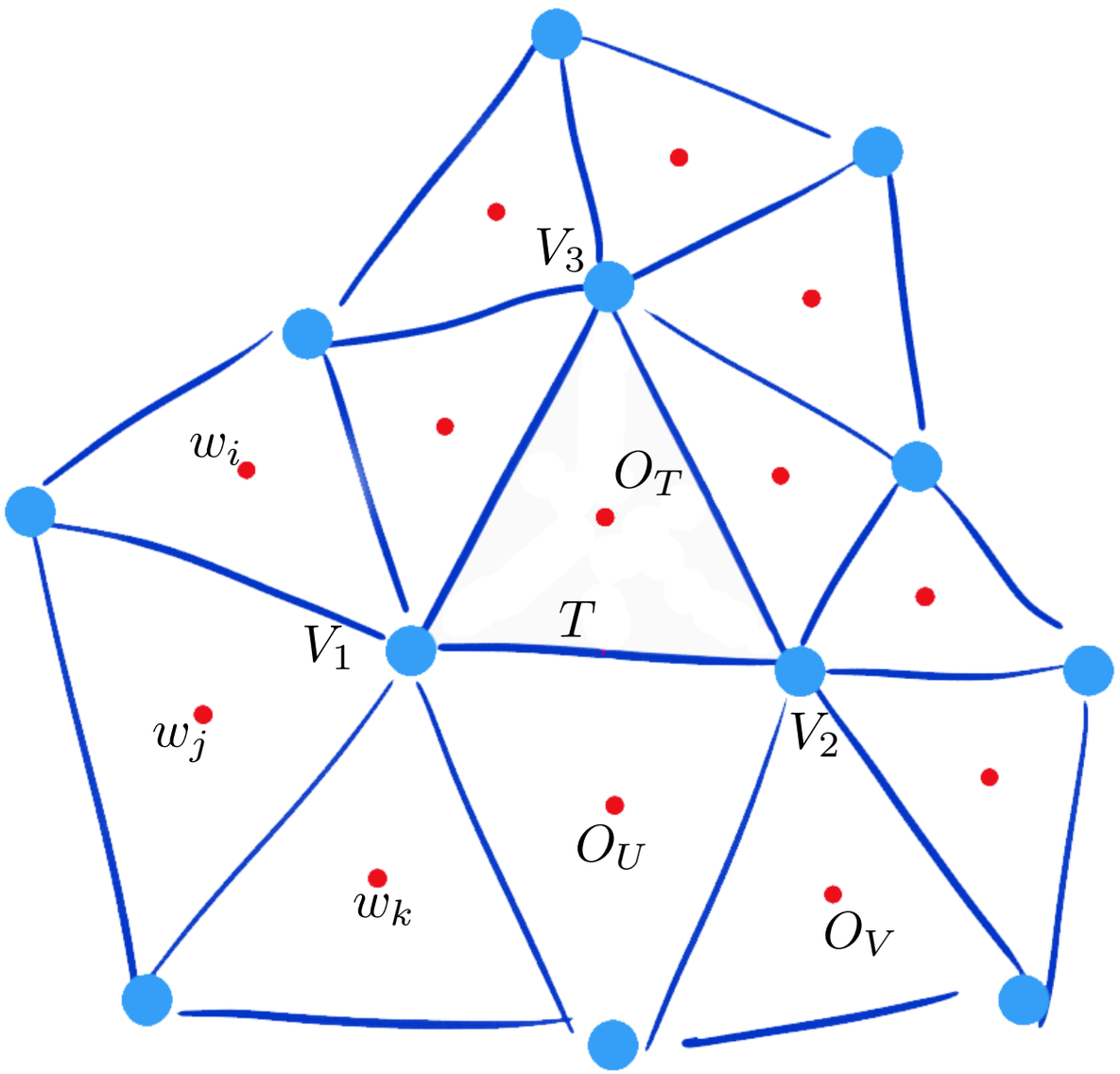




Outline

Thanks to: Dutykh and Tournier

Dutykh *et* al. 2009

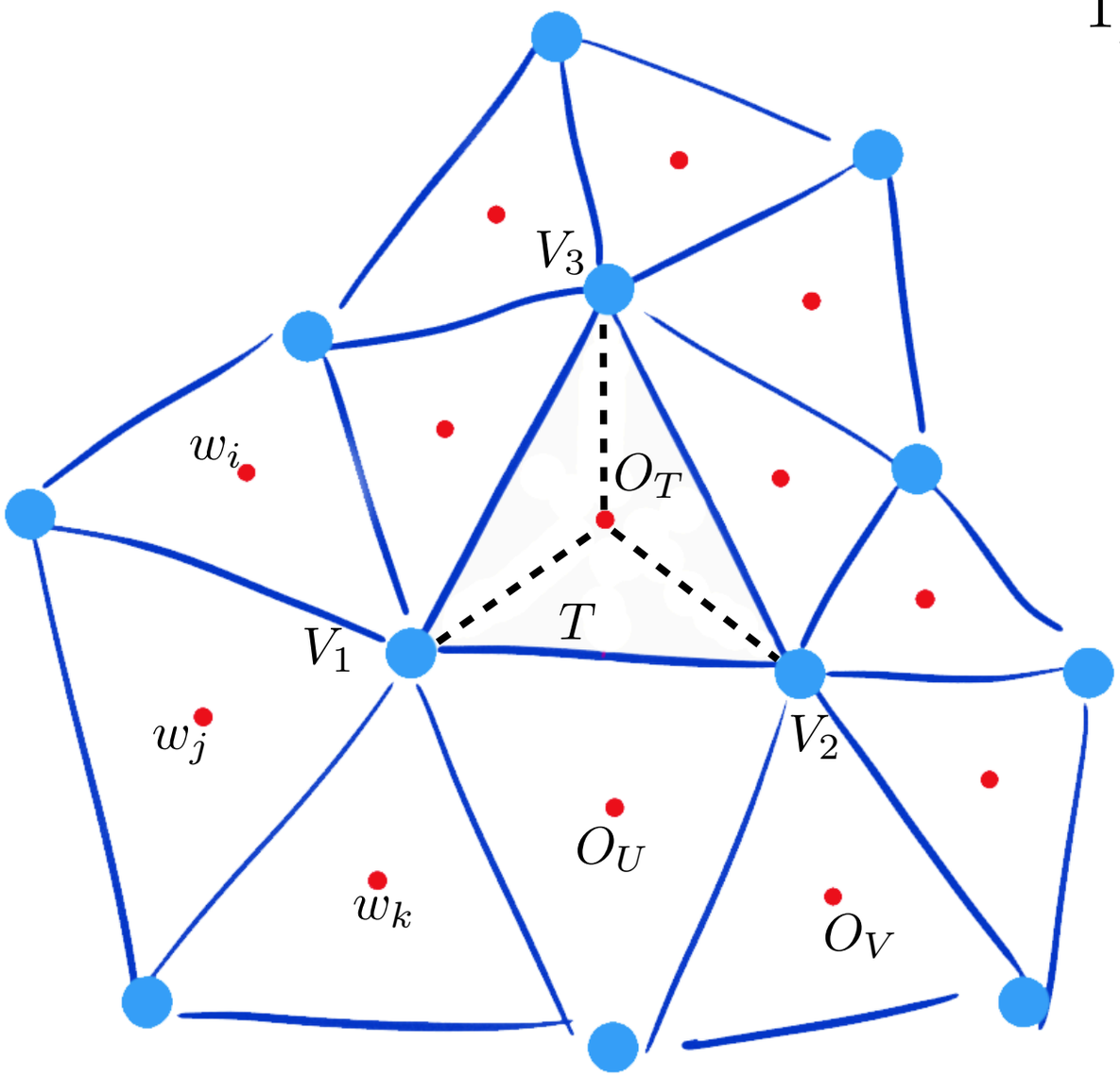


Thanks to: Dutykh and Tournier

Dutykh *et* al. 2009

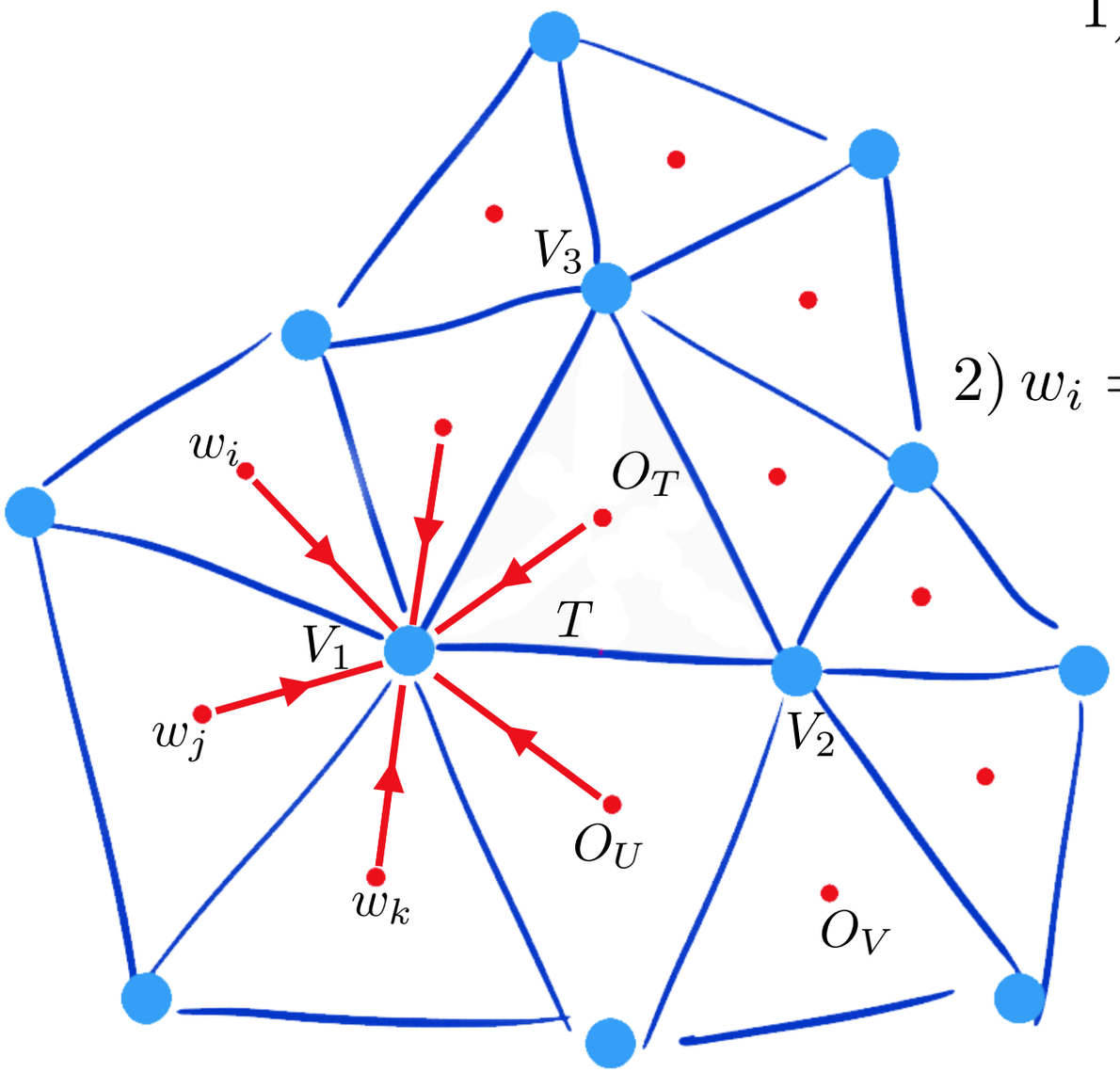
$$u_{V_j} = \sum_{i=1}^{d(N)} w_i u_{O_i} \Big/ \sum_{i=1}^{d(N)} w_i, j = \{1, ..., Th.nv\}$$

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



Dutykh *et* al. 2009

Holmes *et* al. 1989



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

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

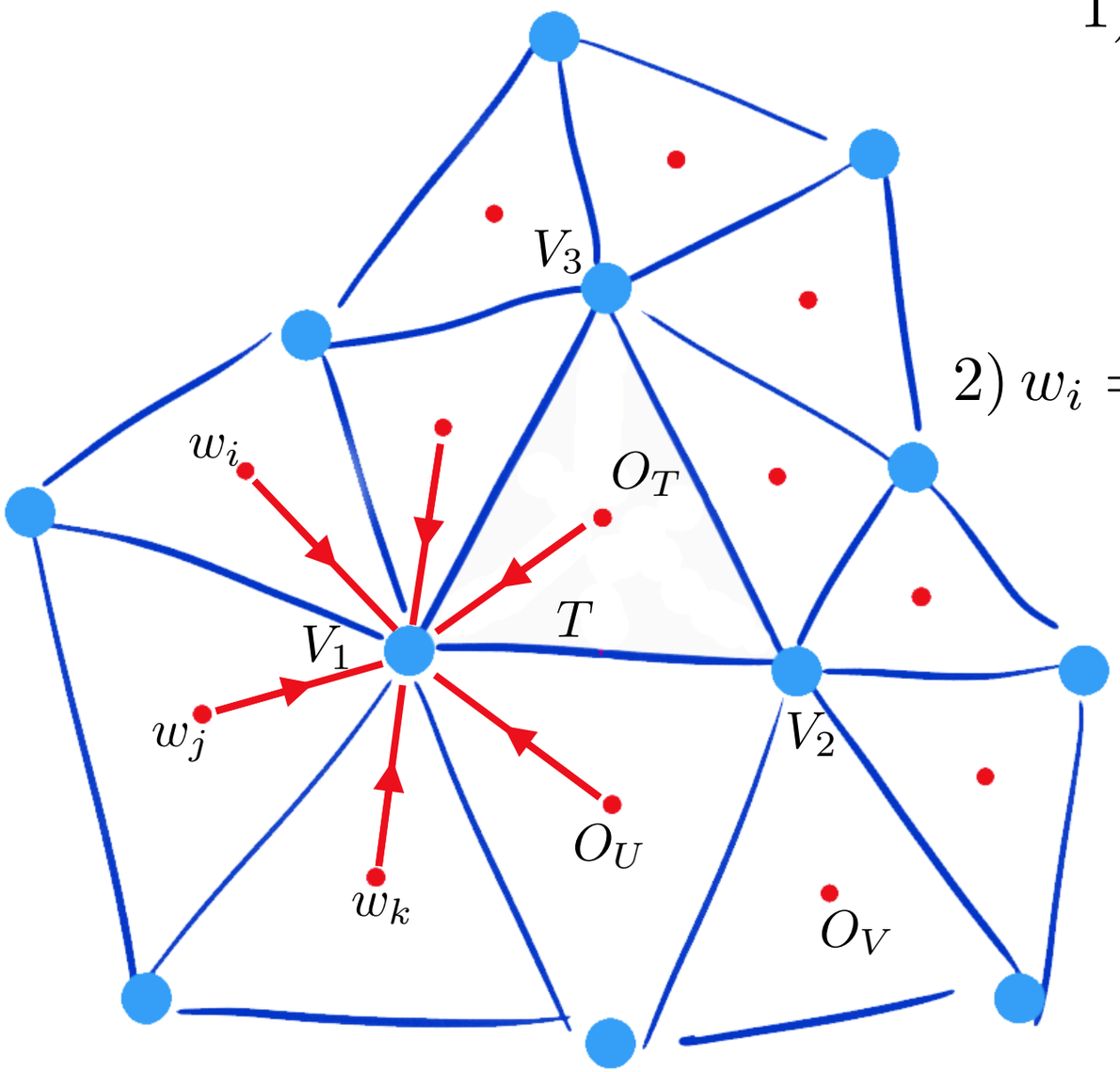
$$2)\, w_i = 1 + \Delta w_i, \min \left(\frac{1}{2} \sum_{i=1}^{d(N)} \left(||\vec{X}_{O_i} - \vec{X}_{V_j}|| \Delta w_i \right)^2 \right)$$

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

Thanks to: Dutykh and Tournier

Dutykh *et* al. 2009

Holmes *et* al. 1989



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

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

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

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

$$u_1 - u_0 = (\nabla u)_T \cdot (\vec{X}_1 - \vec{X}_T) + \mathcal{O}(h^2),$$

$$u_2 - u_0 = (\nabla u)_T \cdot (\vec{X}_2 - \vec{X}_T) + \mathcal{O}(h^2),$$

$$u_3 - u_0 = (\nabla u)_T \cdot (\vec{X}_3 - \vec{X}_T) + \mathcal{O}(h^2).$$

least square

Gradient computation

$$u = \exp(-x^2/2-y^2/2), u_x = -x\exp(-x^2/2-y^2/2), u_{xx} = (-1+x^2)\exp(-x^2/2-y^2/2)$$

$$\|dxFV(u) - u_x\|_{L^2}, u \in \mathbb{P}_0 \quad , \quad \|dxxFV(u) - u_{xx}\|_{L^2}, u \in \mathbb{P}_0$$

$$\|dx(u) - u_x\|_{L^2}, u \in \mathbb{P}_1 \quad , \quad \|dxx(u) - u_{xx}\|_{L^2}, u \in \mathbb{P}_2$$

Np	rate $dx(u)$ $u \in \mathbb{P}_1$	CPU time	rate $dxFV(u)$ $u \in \mathbb{P}_0$	CPU time	rate $dxx(u)$ $u \in \mathbb{P}_2$	CPU time	rate $dxxFV(u)$ $u \in \mathbb{P}_0$	CPU time
256	1.45184	0.002s	1.4065	0.284s	1.40217	0.012s	1.15787	0.541s
512	1.42169	0.013s	1.36179	1.095s	1.39432	0.047s	0.93441	2.067s
1024	1.42988	0.061s	1.44301	4.325s	1.43562	0.193s	1.09217	8.651s

Table 1: L^2 rate comparison of first and second order derivative in FreeFem++ between FE and FV on a circle.

Outline

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

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

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

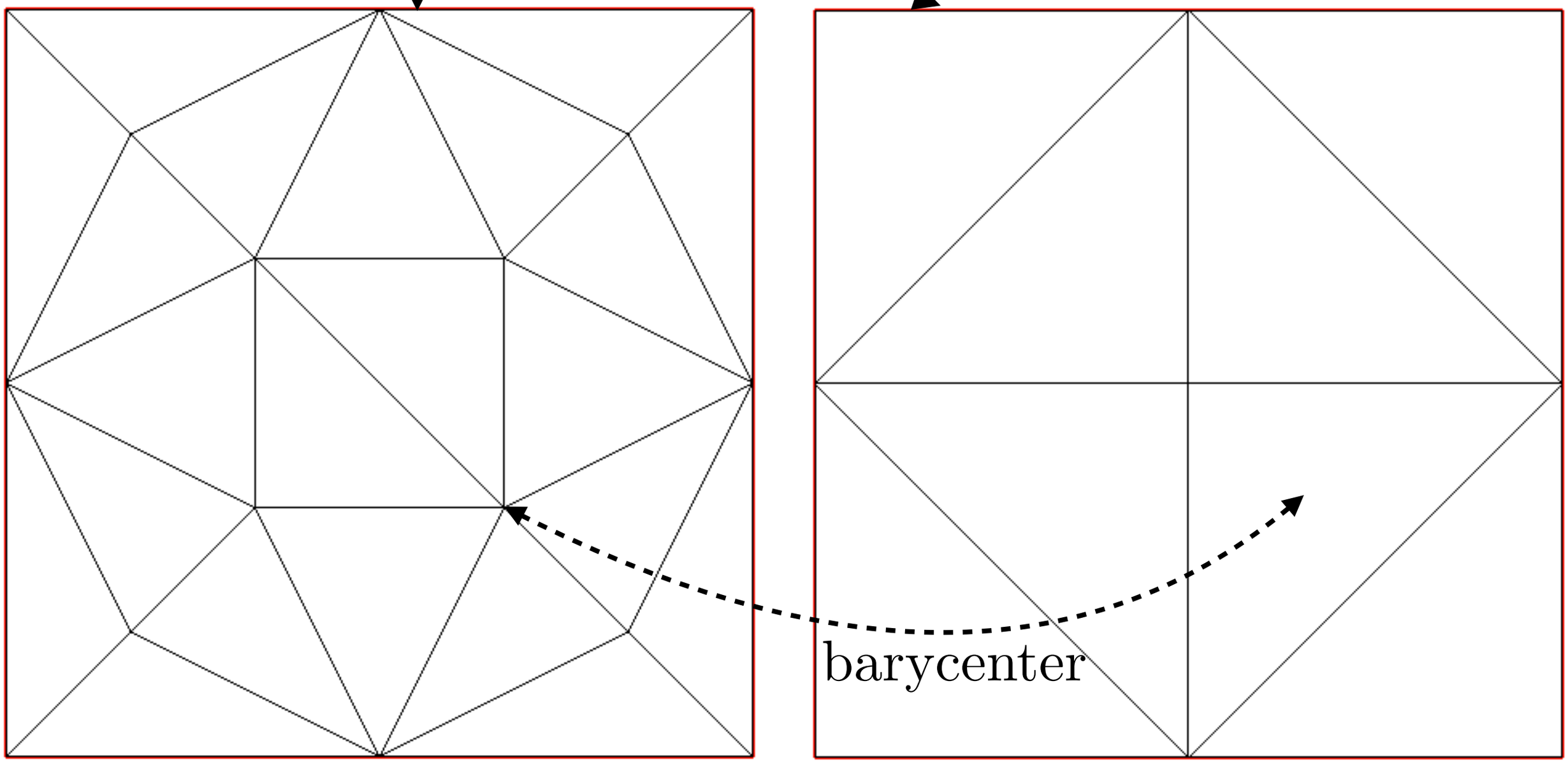
$$\mathcal{A}_{BBM}\partial_t\mathbf{E} + \mathbf{div}([F(\mathbf{E}), G(\mathbf{E})]) = 0,$$

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

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

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

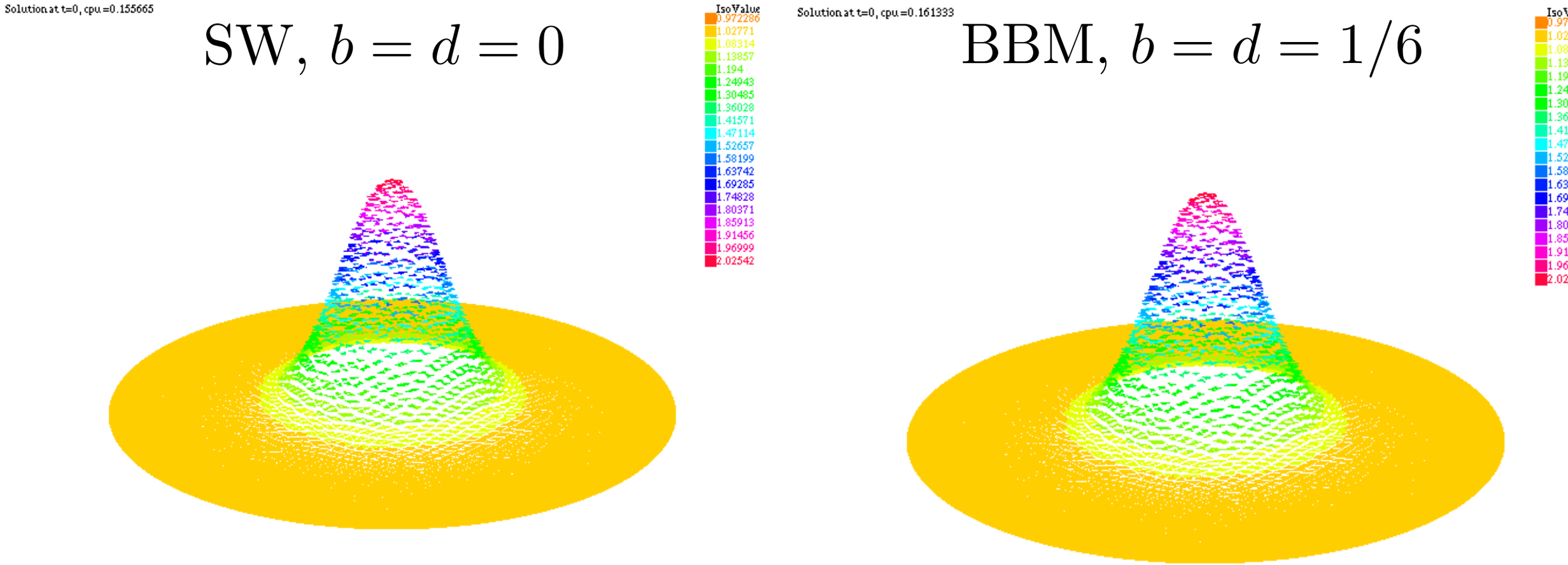
$\mathrm{Th1}, \mathbb{P}_1$ $\mathrm{Th}, \mathbb{P}_0, \mathbb{P}_{0edge}$



```
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
```

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

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

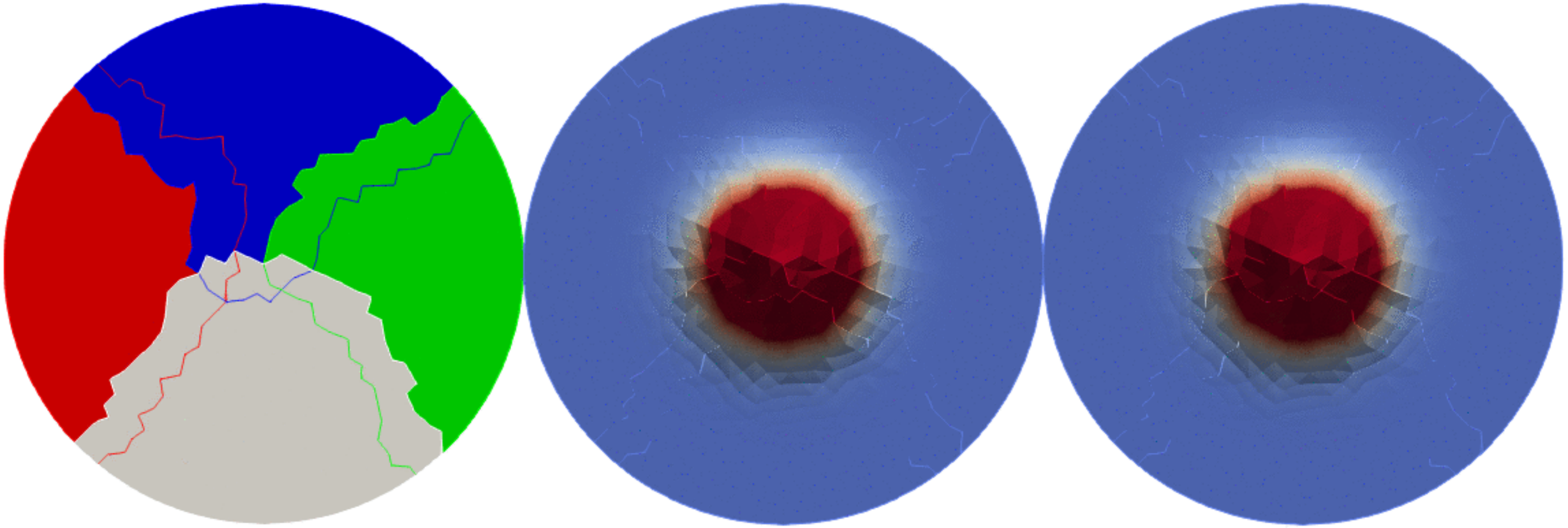


http://www.georges-sadaka.fr/movies/FreeVol_SW_BBM.m4v

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

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

SW, $b = d = 0$ BBM, $b = d = 1/6$



http://www.georges-sadaka.fr/movies/FreeVol_SW_BBM_paral.m4v

Outline

Perspectives

- publish this work
- higher order scheme and its parallelization
- 3D version

Thanks for your attention!

www.georges-sadaka.fr