Freefem++ et l'Hyperbolique

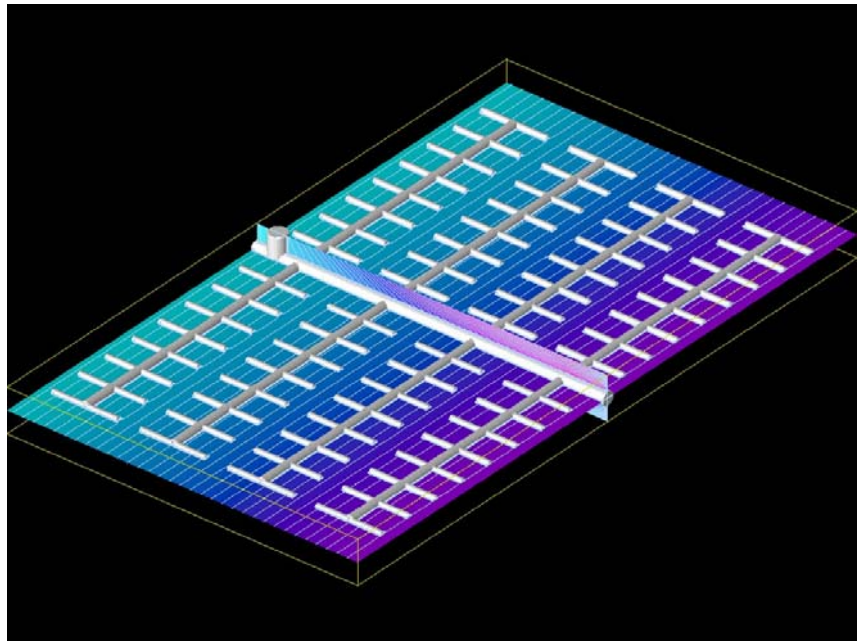
Olivier Pironneau

LJLL-UPMC

Historique

- 1985: MacFEM – PCFEM
- 1990: Interpreteur de formule (+ D. Bernardi) freefem
- 1995: freefem+ (+ Hecht)
- 2000: freefem++ (Hecht tout seul)
- 2000: freefem3D (DelPino, Havé , Pironneau,)
- 2005: une nouvelle documentation

- ff3D:
  - Input via POVRay (SGS et in(x,y,z))
  - Domaines fictifs + maillage auto par marching cube
  - Solveurs itératifs parallèles préconditionnés multigrid
  - Visualisation par medit (P. Fray)

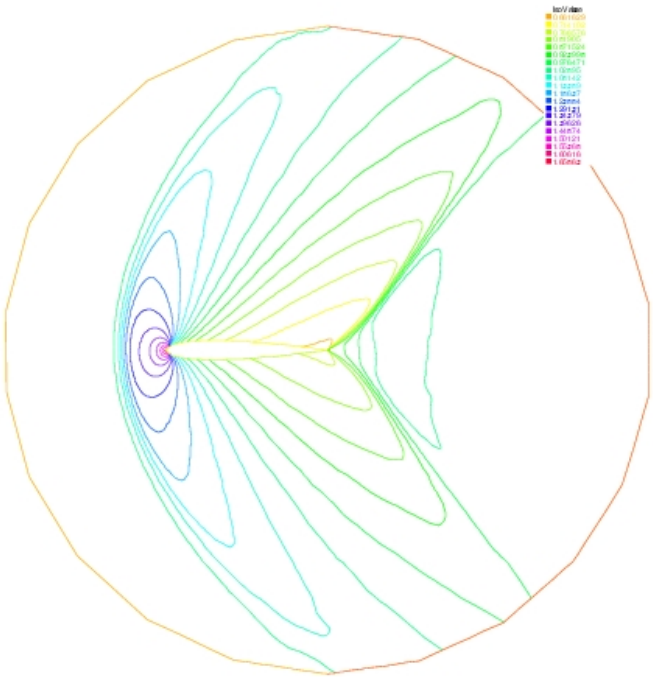


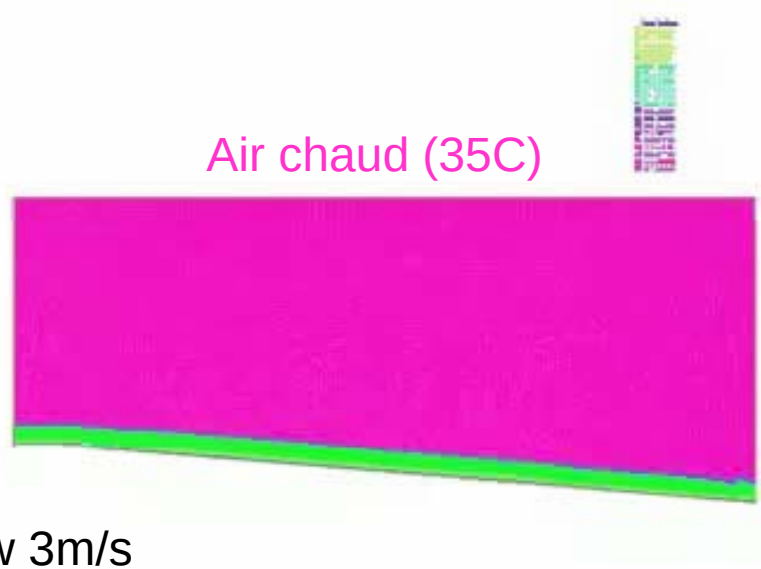
Le décentrage freefem par défaut

- Freefem= solveur elliptique
  + Caractéristique – Galerkin:

$$(\partial_t A + u\nabla A)\big|_x^{m+1/2} \simeq \frac{1}{\delta t}[A^{m+1}(x) - A^m(x - u(x)^{m+1/2}\delta t)]$$

```
=( A - convect(A,[u1,u2],dt) )/dt
```





In-flow 3m/s

Air froid (25C)

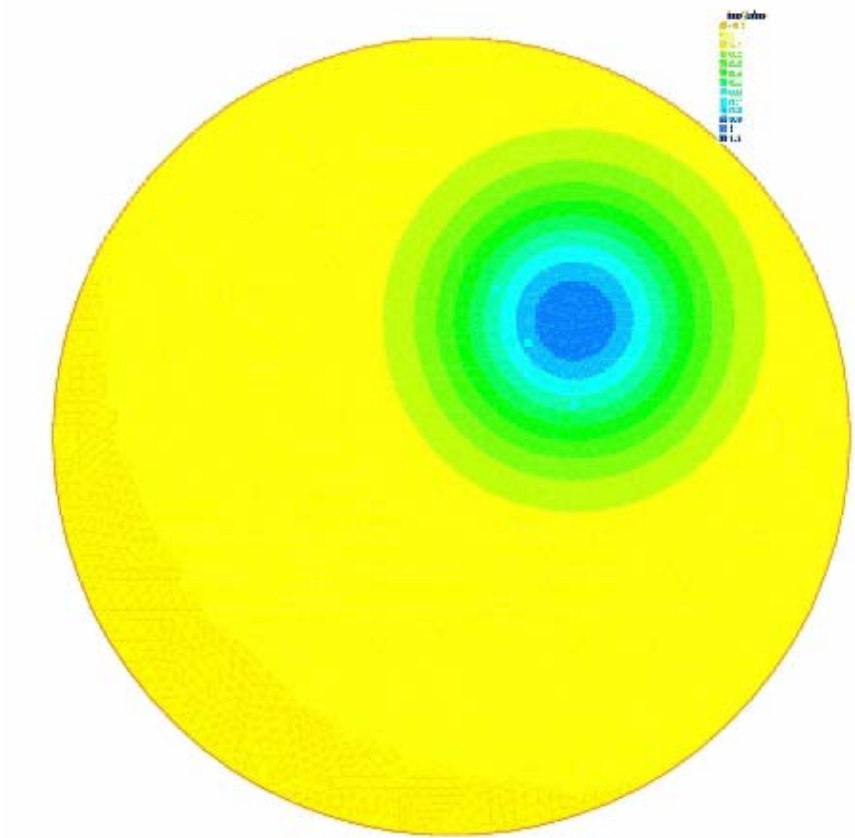
SUPG-Least Square Galerkin

$$\partial_t v + L(v) = 0 \Rightarrow \partial_t v + L(v) - \alpha L(\partial_t v + L(v))$$

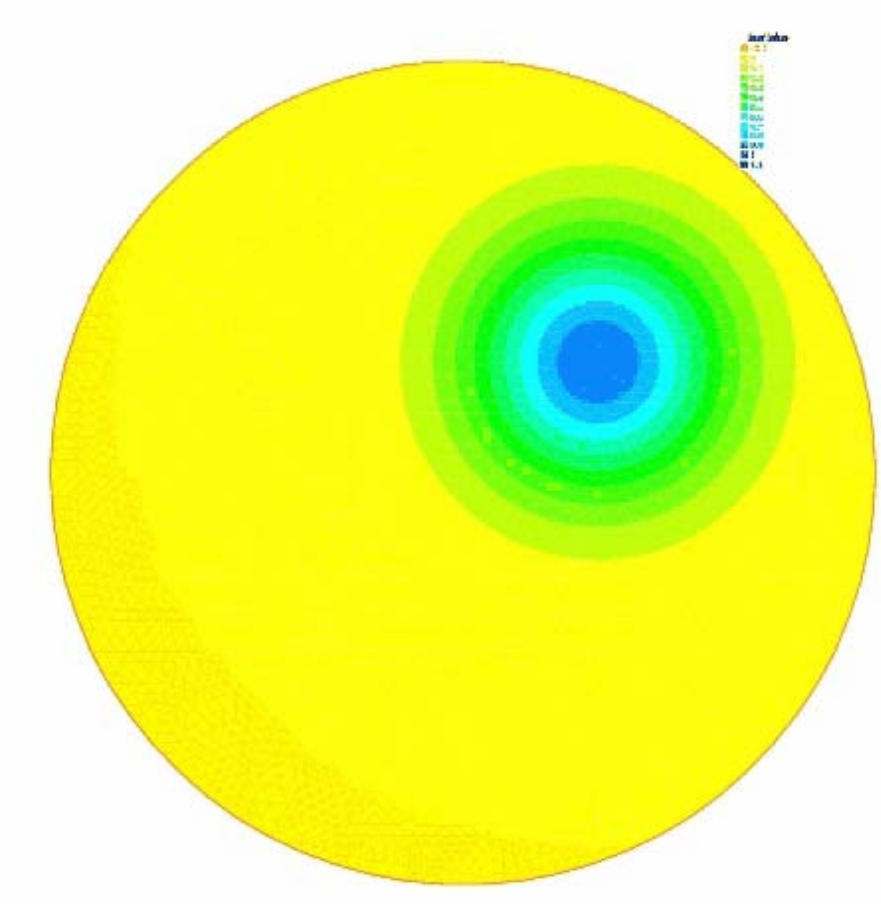
```
border a(t=0, 2*pi)     {    x = cos(t);    y = sin(t);  };
mesh th = buildmesh(a(70));
fespace Vh(th,P2);

Vh v,vh,u1 = y, u2 = -x, vo = exp(-10*((x-0.3)^2 +(y-0.3)^2)) ;
real t, dt = 0.1, tmax=3.14, mass0=int2d(th)(vo),alpha=0.1;

problem aa(v,vh) = int2d(th)((vh+ alpha*(u1*dx(vh)+u2*dy(vh)) )
                                  *(v/dt+u1*dx(v)+u2*dy(v)))
             int2d(th)((vo/dt)*(vh+alpha*(u1*dx(vh)+u2*dy(vh))));

for ( t=0; t< tmax ; t+=dt)
{ aa; vo=v; plot(v,fill=0,wait=0);
  cout<<"mass="<<int2d(th)(v)/mass0-1<<endl;
};
```

Résultats / Comparaison CG

SUPG Max=0.82



Caracteristique-Galerkin P2 max=0.9



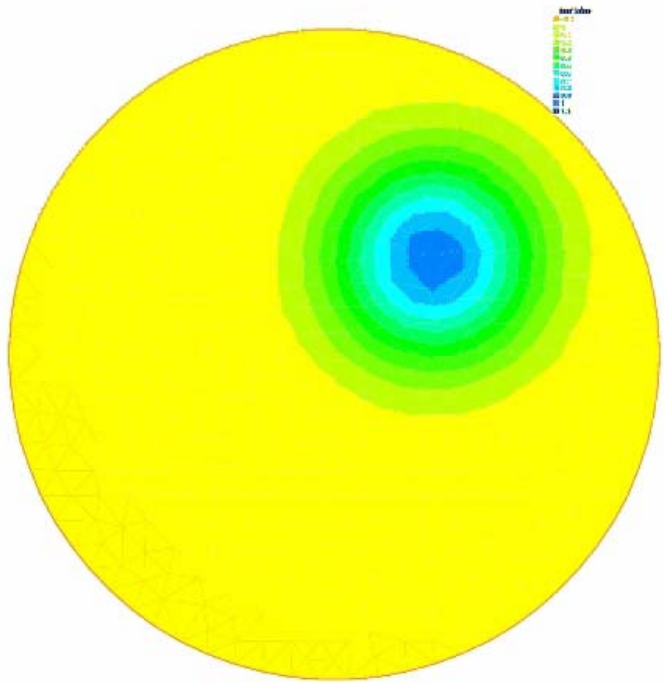
Eléments Finis Discontinus (Piperno et al)

```
fespace Vh(th,P1dc);  Vh v,vh;
varf  A(v,vh) = int2d(th)(v*vh/dt/2);
varf  B(vh,w) =intalledges(th)(vh*mean(w)*(N.x*u1+N.y*u2))
                        -int2d(th)( w*(u1*dx(vh)+u2*dy(vh)));
```

[N.x,N.y]=vecteur normal Mean(w)=(v+ + v-)/2

$$\int_T \frac{1}{2\delta t}(v^{m+1} - v^{m-1})w$$

$$= \int_T u\nabla w)v^m - \int_{\partial T} \bar{v}^m w(u.n)$$



Galerkin - Discontinu

```
border a(t=0, 2*pi)     {    x = cos(t);    y = sin(t);  };
mesh th = buildmesh(a(70));
fespace Vh(th,P1dc);
Vh vh,vo,u1 = y, u2 = -x, v = exp(-10*((x-0.3)^2 +(y-0.3)^2));
real dt = 0.03,t=0, tmax=3.14, al=0.5, alp=200;
macro n(u) (N.x*u1+N.y*u2)//

problem  A(v,vh) = int2d(th)(v*vh/dt-v*(u1*dx(vh)+u2*dy(vh)))
  + intalledges(th)(vh*(mean(v)*n(u)+alp*jump(v)*abs(n(u))))
  + int1d(th,1)((n(u)>0)*n(u)*v*vh)  -  int2d(th)(vo*vh/dt);

problem  Adual(v,vh) = int2d(th)((v/dt+(u1*dx(v)+u2*dy(v)))*vh)
+ intalledges(th)((1-nTonEdge)*vh*(abs(n(u))-n(u))/2*jump(v))
- int1d(th,1)((n(u)<0)*abs(n(u))*v*vh) - int2d(th)(vo*vh/dt);

for ( t=0; t< tmax ; t+=dt)  {vo=v; Adual;};
```

nTonEdge=combien de triangle s'appuie sur l'arete.

Go

$$\int_{\Omega} (\partial_t v + u\nabla v)w - \sum \int_{\partial T - \Gamma} u.n^{-}[v]w = 0$$

DG Despres-Kamga

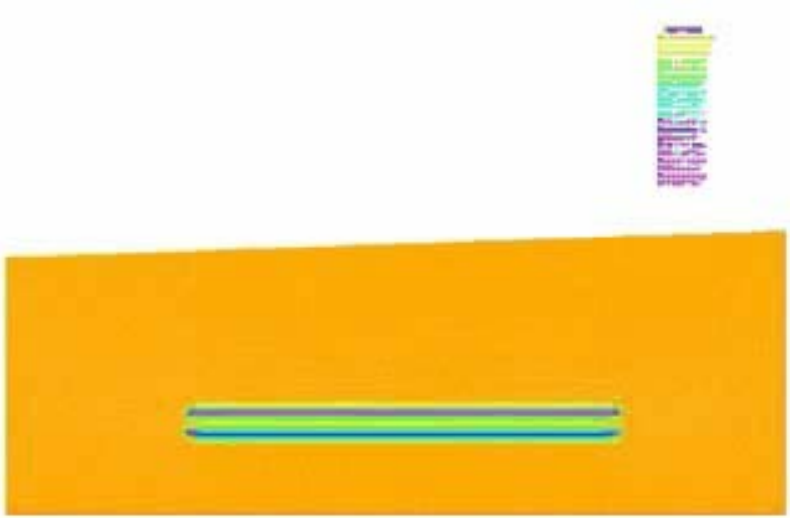
```
border aa(t=0, 2*pi){ x = cos(t);    y = sin(t);  }; mesh th = buildmesh (aa(70));

fespace Vh(th,P1dc);
Vh vh,vo,u1 = y, u2 = -x, v = exp(-10*((x-0.3)^2 +(y-0.3)^2));
real K1=0.04, K2=0.02, K12=-0.01, K21=-0.0, dt = 0.05,t=0, tmax=3.14, a=1;
macro n(u) (N.x*u1+N.y*u2) //

macro Kdn(u) (N.x* (K1* dx(u)+K12* dy(u)) + N.y*(K2* dy(u)+ K21* dx(u))) //
macro e(v) (v+jump(v)) //
macro KeD(u)(N.x*(K1* e(dx(u))+K12*e(dy(u)))+ N.y*(K2* e(dy(u))+ K21* e(dx(u)))) //

problem  A(v,vh) = int2d(th)(v*vh/dt-v*(u1*dx(vh)+u2*dy(vh))
  +2*(K1*dx(v)* dx(vh)+ K2*dy(v)* dy(vh) + K12*dy(v)* dy(vh)+ K21*dy(v)* dy(vh)))
+ intalledges(th)(((a*v-Kdn(v))*(a*vh-Kdn(vh))-
(a*e(v)+KeD(v))*(a*vh+Kdn(vh)))/a/2)
+ intalledges(th)(n(u)*vh*(vo*(n(u)>0) + e(vo)*(n(u)<0))) - int2d(th)(vo*vh/dt);

for ( t=0; t< tmax ; t+=dt){vo=v; A;}
```

$$\int_{\Omega}(w\partial_t v-(u\nabla w)v+2(K\nabla v)\nabla w)+\int_{u.n>0}u.nwv+\int_{u.n<0}u.nwv^{+}$$

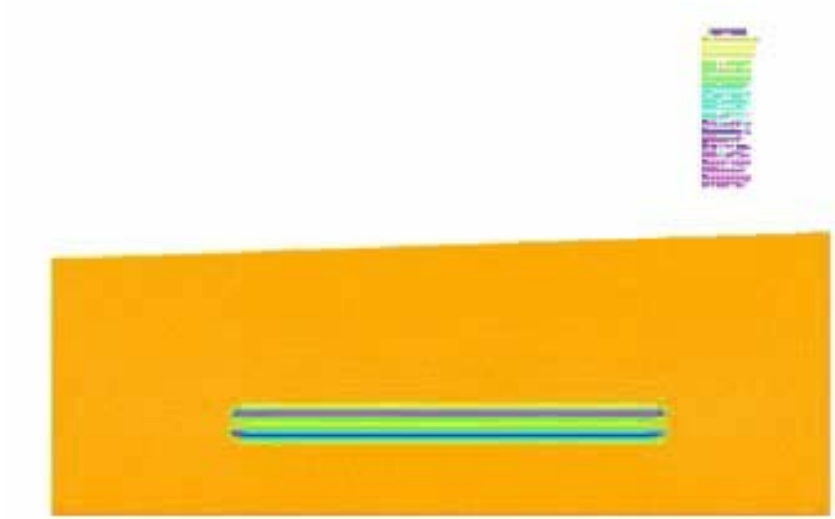
$$+\int_{\partial T}\frac{1}{2a}(av-K\nabla v.n)(aw-K\nabla w.n)-(av^{+}+K\nabla v^{+}.n)(aw+K\nabla w.n)=0$$

Résultats sur le cas Couplex

Despres-Kamga DG



Characteristics-Galerkin



Concentration d’Iode 129 après la rupture des conteneurs dans le site de Bure

La méthode PSI (Deconinck-Struijs)

Positive Streamwise Implicit (cf Perthame)

Schéma explicite nonlinéaire!, Programmé dans freefem par load module

```
load "mat_psi"

border a(t=0, 2*pi){    x = cos(t);    y = sin(t);}
mesh th = buildmesh(a(100));
fespace Vh(th,P1);
Vh vh,vo,u1=y, u2=-x, v=exp(-10*((x-0.3)^2+(y-0.3)^2)), rhs=0;
real dt = 0.05,t=0, tmax=3.14;

problem  A(v,vh) = int2d(th,qft=qf1pTlump)(v*vh/dt)
                  -int2d(th,qft=qf1pTlump)(vo*vh/dt) + rhs[];

for ( t=0; t< dt+0.001 ; t+=dt){
  vo=v;
  matrix B;
  MatUpWind0(B,th,vo,[u1,u2]);
  rhs[] = B* vo[] ;
  A;
};
```

mat_psi.cpp

```
#include "RNM.hpp"
. . .
class MatrixUpWind0 :  public E_F0 { public:  typedef Matrice_Creuse<R> * Result;
  MatrixUpWind0(const basicAC_F0 & args){. . .}
   static ArrayOfaType  typeargs() { return…}
   . . .
};

int   gladys(double q[3][2], double u[2],double c[3], double a[3][3], double area )  //PSI Deconninck
{
  double dw[3][2]; // basis function gradients times  area
  double ua[2], kk[3], beta[3]; // to define a[][]
. . .
  }


AnyType MatrixUpWind0::operator()(Stack stack) const
{
  Matrice_Creuse<R> * sparce_mat =GetAny<Matrice_Creuse<R>* >((*emat)(stack));
 . . .
 if (gladys(q,u,c,a,K.area) )
      { for (int i=0;i<3;i++)
          for (int j=0;j<3;j++)
        if (fabs(a[i][j]) >= 1e-30) Aij[make_pair(ii[i],ii[j])]+=a[i][j];
      }
. . .
  return sparce_mat;
}

class Init { public: Init();};
 Init init;
 Init::Init(){   cout << " lood: init Mat Chacon " << endl;
     Global.Add("MatUpWind0","(", new OneOperatorCode<MatrixUpWind0 >( ));
  }
```

Creation de mat_psi.dll

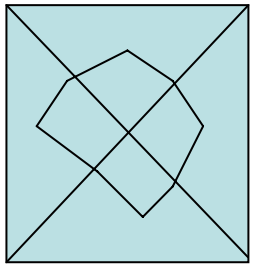
```
#!/bin/sh
# Create a loadable object from a C++ function defined in a .cpp file
# $Id: load.link,v 1.12 2005/07/12 09:22:20 hecht Exp $
do="yes"
if [ "$1" = "-n" ]; then
  shift
  do="no"
fi
if [ "$1" = "-win32" ]; then
  shift
  uu="WIN32"
else
  uu=`uname -s`
fi
if  [ "$1" = "-I" ]; then
  INC=$2; shift;shift;
fi
 if  [ "$1" = "-l" ]; then
  LIBS=$2; shift;shift;
fi
# Default compiler
if [ "$CXX" = "" ];
then
    CXX=g++
fi
```

```
FFsource=..INC="-Iinclude $INC"
SUF=so
if [ -f "$1.cpp" ] ; then
   case "$uu" in    Darwin)
   export MACOSX_DEPLOYMENT_TARGET=10.3
   SHARED="-bundle -undefined dynamic_lookup"  ;;
   CYGWIN*|FreeBSD)
      SHARED="-shared " ;;
   # 64 bit Linux needs -fPIC (ALH)
  Linux)
      FLAGS='-fPIC'
      SHARED="-shared " ;;

    WIN32)
      echo " window "
   SHARED="-v -shared --unresolved-symbols=ignore-all"
      FLAGS='  -mno-cygwin '
      LIBS="libff0.dll libff1.dll libff2.dll $LIBS"
      SUF=dll;;
   *)
   echo "sorry unknown achitecture "`uname`
   exit 1
   esac
  FLAGS="$FLAGS -g"
   echo $CXX -c $FLAGS $INC $PIC $1.cpp
   test $do = yes &&$CXX -c $INC $FLAGS $PIC  $1.cpp

   echo $CXX $SHARED $FLAGS $1.o -o $1.$SUF $LIBS
   test $do = yes &&$CXX $SHARED $FLAGS $1.o -o $1.$SUF $LIBS
fi
```

Go

Méthodes de Volumes Finis: ex Dervieux et al

- Un volume σ est associé à chaque sommet

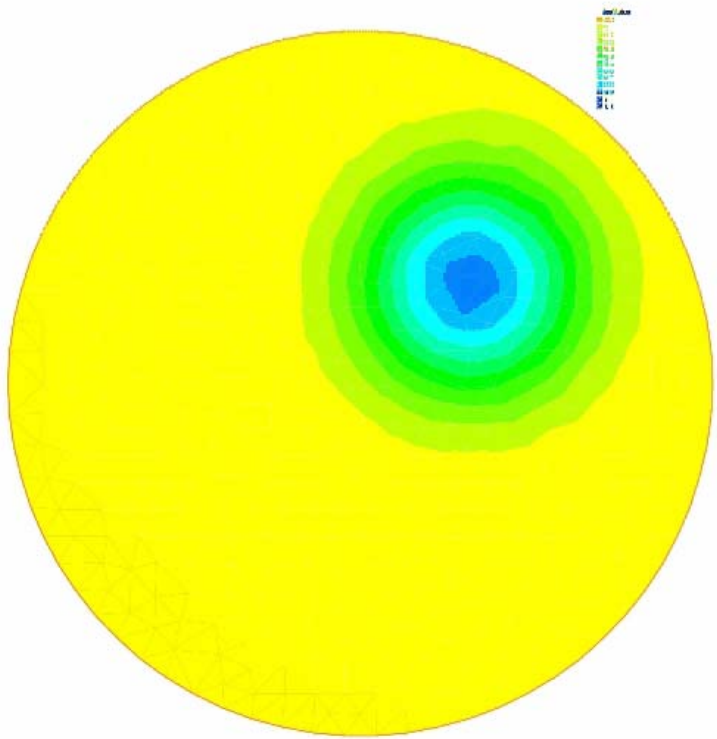


$$\partial_t v + \nabla . F(v) = 0 \Rightarrow \int_{\sigma} \partial_t v + \int_{\partial\sigma} F(v).n = 0$$

-L'assemblage triangle/triangle est possible

-La première intégrale vaut 1/3 de la même sur les triangles

-Il faut écrire un load module pour les intégrales de bords.



Max=0.43 (très diffusif)

Problemes Vectoriels

Perspectives

- Tout y est mais peut être faudrait il trouver un formalisme pour intégrer automatiquement certain de ces schémas (comme pour CG)
- Faire les cas tests classiques (marche montante etc)
- Sensitivity, optimisation?