Finite Elements for Fluids
Prof. Sheu

Yannick Deleuze

National Taiwan University

ydeleuze@ntu.edu.tw

http://homepage.ntu.edu.tw/~ydeleuze/

NTU, Taipei, May 4, 2016

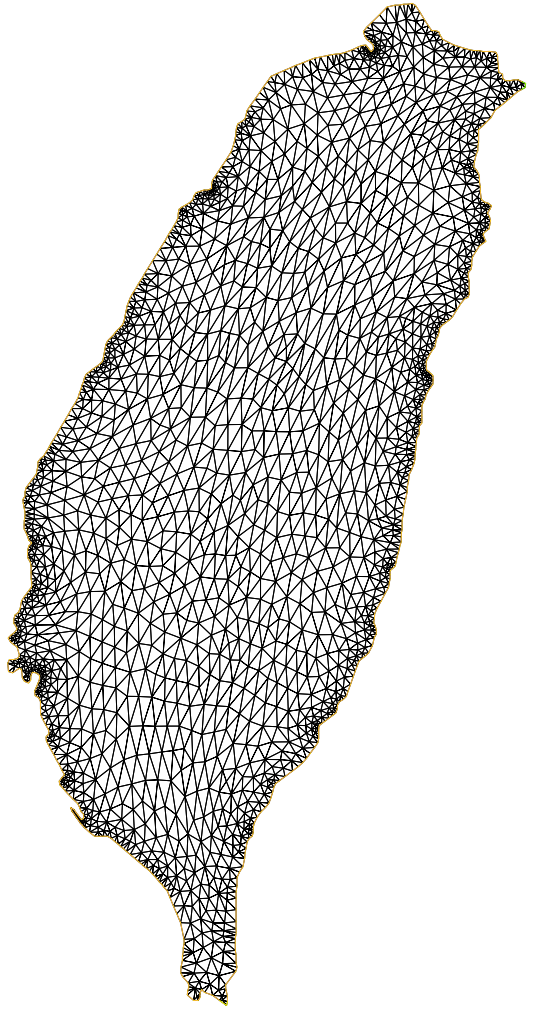
Outline

Outline

Outline

Triangular meshes

Using triangular elements, one can approximation more general domains Ω than with rectangular elements.
Let $\mathcal{T}$ be a triangulation.



Outline

Barycentric coordinates

Let A^1, A^2, and A^3 be three nonaligned points.

Proposition

$\forall M \in \mathbb{R}^2$, $\exists!(\lambda_1, \lambda_2, \lambda_3)$ *and* $\lambda_1 + \lambda_2 + \lambda_3 = 1$ *such that*

$$M = \sum_{i=1}^{3} \lambda_i \, A^i.$$

i.e. $\overrightarrow{OM} = \sum_{i=1}^{3} \lambda_i \overrightarrow{OA^i}$

Proof.

Let $M = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ in cartesian coordinates and $A^i = \begin{pmatrix} x_1^i \\ x_2^i \end{pmatrix}_{i=1,2,3}$.

We look for $(\lambda_1, \lambda_2, \lambda_3)$ such that

$$\begin{cases} \lambda_1 + \lambda_2 + \lambda_3 = 1 \\ \lambda_1 x_1^1 + \lambda_2 x_1^2 + \lambda_3 x_1^3 = x_1 \\ \lambda_1 x_2^1 + \lambda_2 x_2^2 + \lambda_3 x_2^3 = x_2. \end{cases}$$

3 equations, 3 unknowns $\lambda_1, \lambda_2, and \lambda_3$. The determinant

$$\begin{vmatrix} 1 & 1 & 1 \\ x_1^1 & x_1^2 & x_1^3 \\ x_2^1 & x_2^2 & x_2^3 \end{vmatrix} \neq 0$$

iff A^i are not aligned. $\forall M \in \mathbb{R}^2$, $\exists!(\lambda_1, \lambda_2, \lambda_3)$ and $\lambda_1 + \lambda_2 + \lambda_3 = 1$ such that $M = \sum_{i=1}^{3} \lambda_i A^i$. □

Barycentric coordinates

Definition

(i) (A^1, A^2, A^3), three nonaligned points, forms an affine frame.

(ii) $(\lambda_1(M), \lambda_2(M), \lambda_3(M))$ are the barycentric coordinates of M in that affine frame.

We can change the coordinates from cartesian frame to barycentric by solving the 3×3 system shown before and from barycentric to cartesian using $\overrightarrow{OM} = \sum_{i=1}^{3} \lambda_i \overrightarrow{OA^i}$.

Proposition

The barycentric coordinates are invariant by an affine transformation F, defined by $F(M) = A\,\overrightarrow{OM} + b$ and where A is linear and b is a vector.

Proposition

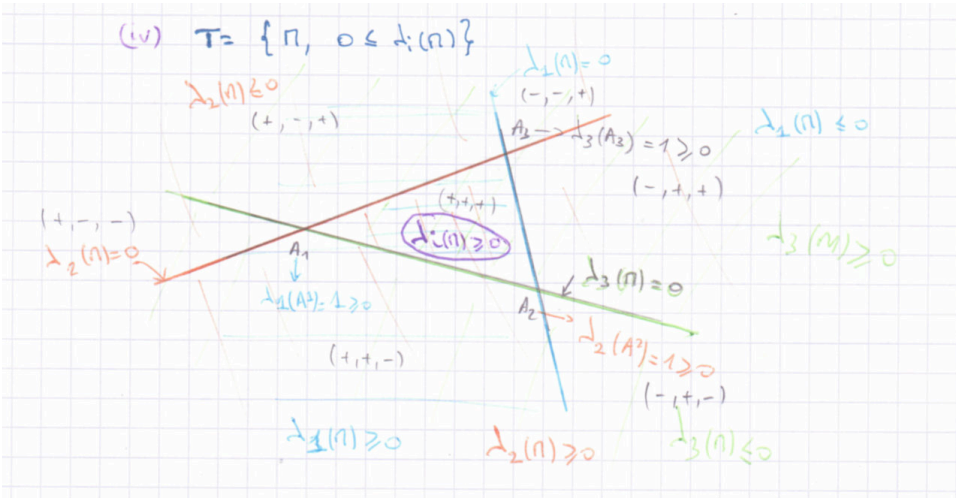
Let (A^1, A^2, A^3) *be an affine frame.*

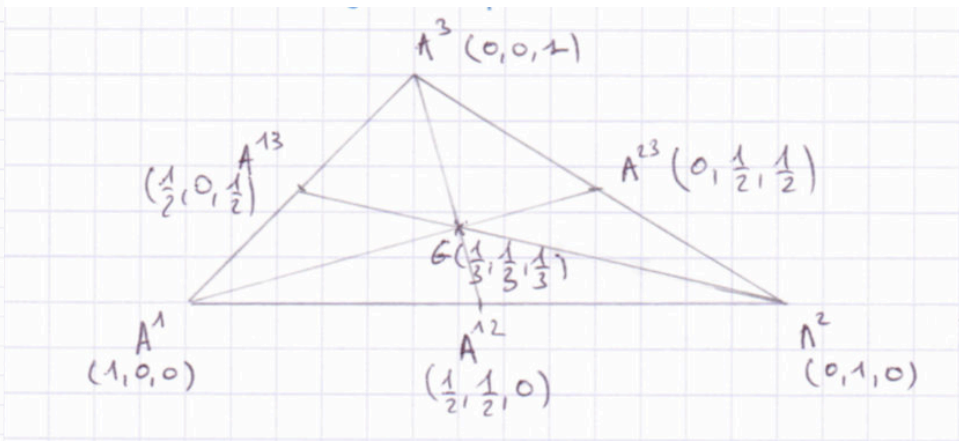
(i) $\lambda_i(A^j) = \delta_{ij}$,

(ii) $\lambda_i \in \mathbb{P}_1$,

(iii) $M \in (A^i, A^j)$ *iff* $\lambda_k(M) = 0$, $i \neq j$ *and* $k \neq i$, $k \neq j$,

(iv) *Let* T *be the triangle* (A^1, A^2, A^3) *then* $M \in T$ *iff* $0 \leq \lambda_i(M) \leq 1$, $i = 1, 2, 3$.





Outline

Linear Triangular Elements (P_1 Triangular Elements)

- $\dim P_1 = 3$
- standard basis $\{1, x, y\}$

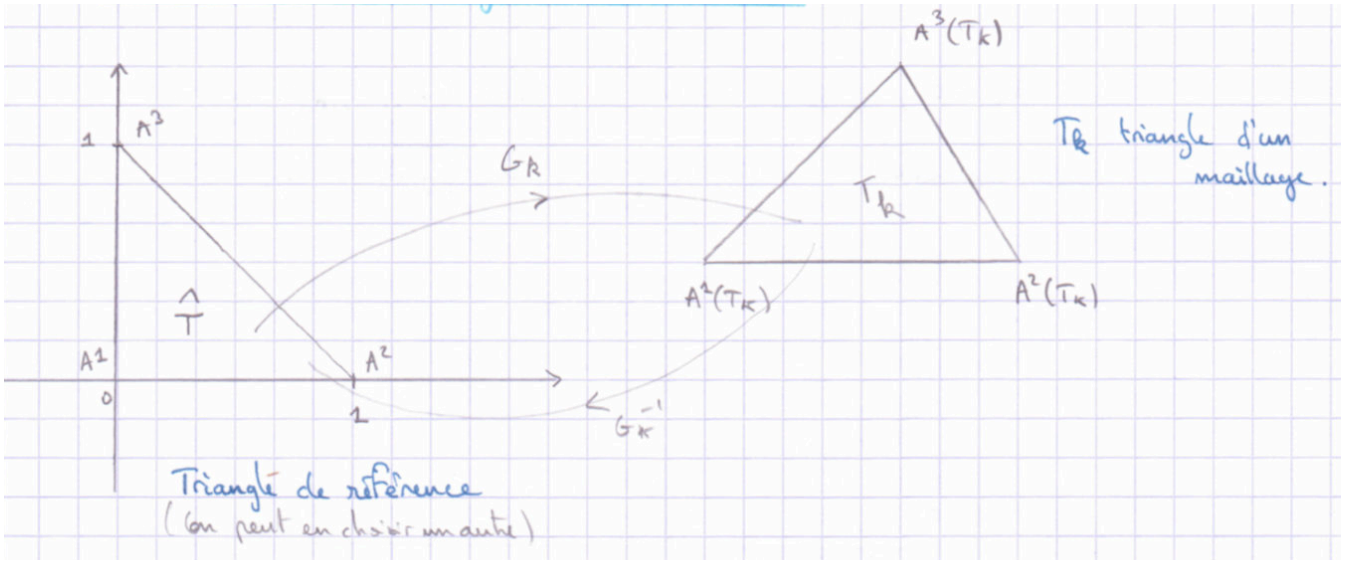
Linear basis functions

Let T be the triangle (A^1, A^2, A^3), The basis polynomials are the λ_i in the frame (A^1, A^2, A^3) because $\lambda_i(A^j) = \delta_{ij}$.

Proposition

Let $V_h = \{v_h \in (C)^0, v_{h|_{T_K}} \in P_1, \forall T_K \in \mathcal{T}\}$.
The elements of V_h *are determined uniquely by their values at the vertices of the triangles.*

Linear Triangular Elements (P_1 Triangular Elements)



There is a unique affine application G_K such that $G_K(A^i) = A^i(Tk)$ and it is bijective. In the reference triangle $\hat{T}$, the barycentric coordinates can be stated easily by the cartesian coordinates.

$$M = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad A^1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad A^2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad A^3 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Then $M = (1 - x_1 - x_2)\, A^1 + x^1\, A^2 + x_2\, A^3$
with $(1 - x_1 - x_2) + x_1 + x^2 = 1$, so that

$$\lambda_1(M) = (1 - x_1 - x_2), \quad \lambda_2(M) = x_1, \quad \lambda_3(M) = x_2.$$

Linear Triangular Elements (P_1 Triangular Elements)

Exemples

Computation of integrals over the triangle T_K of polynomial functions will be needed

(i) $\int_{T_K} \lambda_2(M)^2 dy$

(ii) $\nabla \lambda_i$

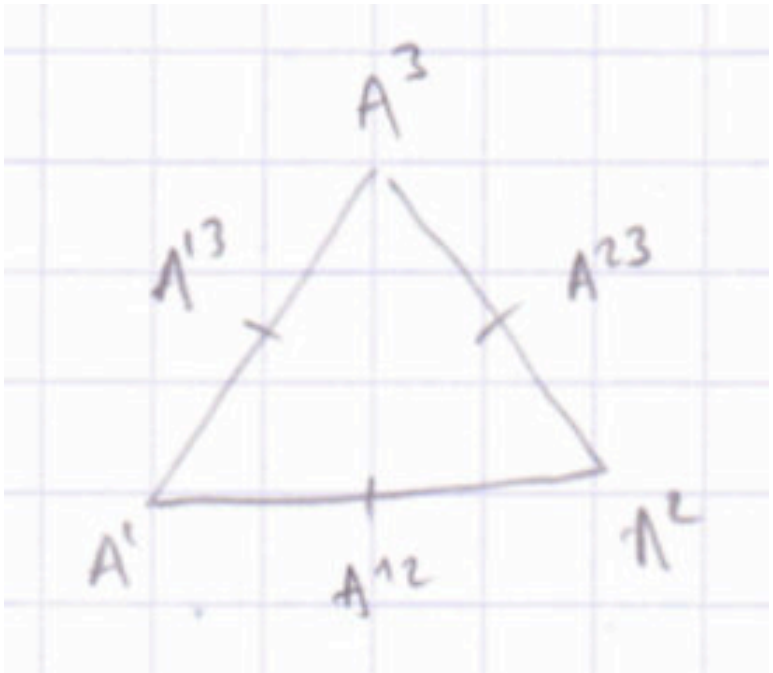
Outline

Quadratic Triangular Elements (P_2 Triangular Elements)

- $\dim P_2 = 6$
- standard basis $\{1, x, y, xy, x^2, y^2\}$

Proposition

Let (A^1, A^2, A^3) *be three nonaligned points, then* $(\lambda_1^2, \lambda_2^2, \lambda_3^2, \lambda_1\lambda_2, \lambda_2\lambda_3, \lambda_1\lambda_3)$ *form a basis of* P_2.



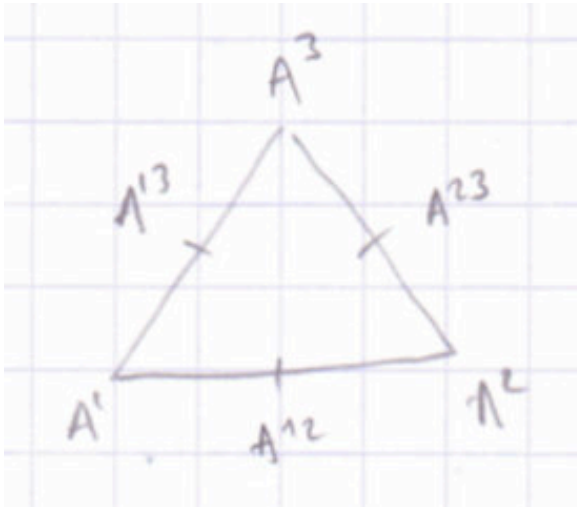
Quadratic Triangular Elements (P_2 Triangular Elements)

Linear basis functions

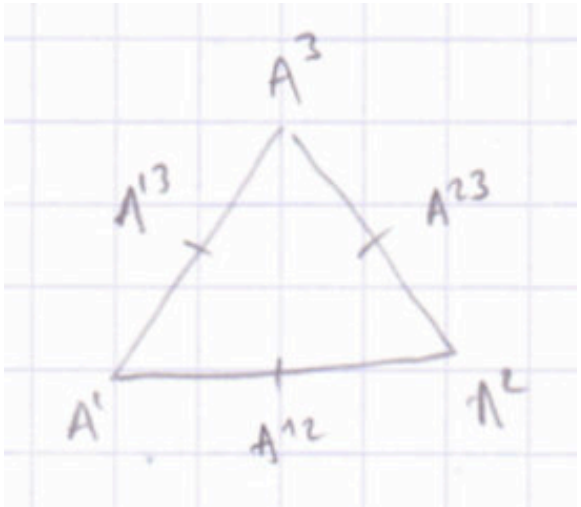
Let T be the triangle (A^1, A^2, A^3), The basis polynomials are
$\lambda_1(2\lambda_1 - 1)$, $\lambda_2(2\lambda_2 - 1)$, $\lambda_3(2\lambda_3 - 1)$, $4\lambda_1\lambda_2$, $4\lambda_2\lambda_3$, $4\lambda_3\lambda_1$ in the frame (A^1, A^2, A^3).

Proposition

Let $V_h = \{v_h \in (C)^0, v_{h|_{T_K}} \in P_2, \forall T_K \in \mathcal{T}\}$.
The elements of V_h *are determined uniquely by their values at the vertices of the triangles and middle of the edges.*



Quadratic Triangular Elements (P_2 Triangular Elements)



(i) Let us construct a polynomial of P_2 such that

$$P_{1,2}(A^{1,2}) = 1 \text{ and } P_{1,2}(A^i) = P_{1,2}(A^{i,i+1}) = 0 \tag{1}$$

$\Rightarrow P_i, i+1 = 4\lambda_i\lambda_{i+1}$

(ii) Let us construct a polynomial of P_2 such that

$$P_1(A^1) = 1 \text{ and } P_1(A^i) = P_1(A^{i,i+1}) = 0 \tag{2}$$

$\Rightarrow P_i = \lambda_i(2\lambda_i - 1)$

Outline

Cubic Triangular Elements (P_3 Triangular Elements)

- $\dim P_3 = 10$
- standard basis $\{1, x, y, xy, x^2, y^2, x^3, x^2y, xy^2, y^3\}$

Proposition

Let (A^1, A^2, A^3) *be three nonaligned points, then*
$(\lambda_1^3, \lambda_2^3, \lambda_3^3, \lambda_1^2\lambda_2, \lambda_1^2\lambda_3, \lambda_2^2\lambda_3, \lambda_2^2\lambda_1, \lambda_3^2\lambda_1, \lambda_3^2\lambda_2, \lambda_1\lambda_2\lambda_3)$ *form a basis of* P_2.

Linear basis functions

Let T be the triangle (A^1, A^2, A^3), The basis polynomials are

$$p_i = \frac{1}{2}\lambda_i(3\lambda_i - 1)(3\lambda_i - 2), \qquad 1 \leq i \leq 3$$

$$p_{ii,j} = \frac{9}{2}\lambda_i\lambda_j(3\lambda_i - 1), \qquad 1 \leq i, j \leq 3, i \neq j$$

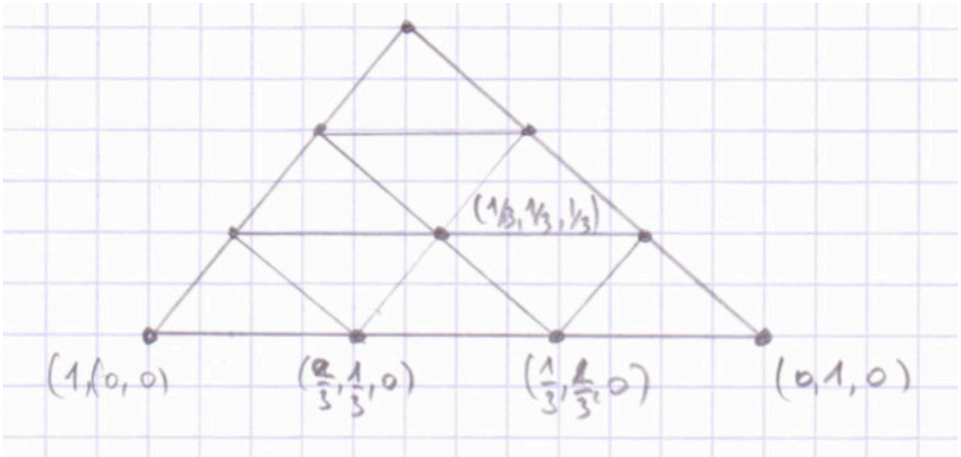
$$p_0 = 27\lambda_1\lambda_2\lambda_3$$

in the frame (A^1, A^2, A^3).

Cubic Triangular Elements (P_3 Triangular Elements)

Proposition

Let $V_h = \{v_h \in (C)^0, v_{h|_{T_K}} \in P_3, \forall T_K \in \mathcal{T}\}$.
The elements of V_h *are determined uniquely by their values at the vertices of the triangles, the thirds of the edges, and the center of gravity.*



Example Let us construct the polynomial of P_3 such that

$$P_0(A^0) = 1 \text{ and } P_0(A^i) = P_0(A^{ii,j}) = 0 \tag{3}$$

Outline

Outline

FreeFem++

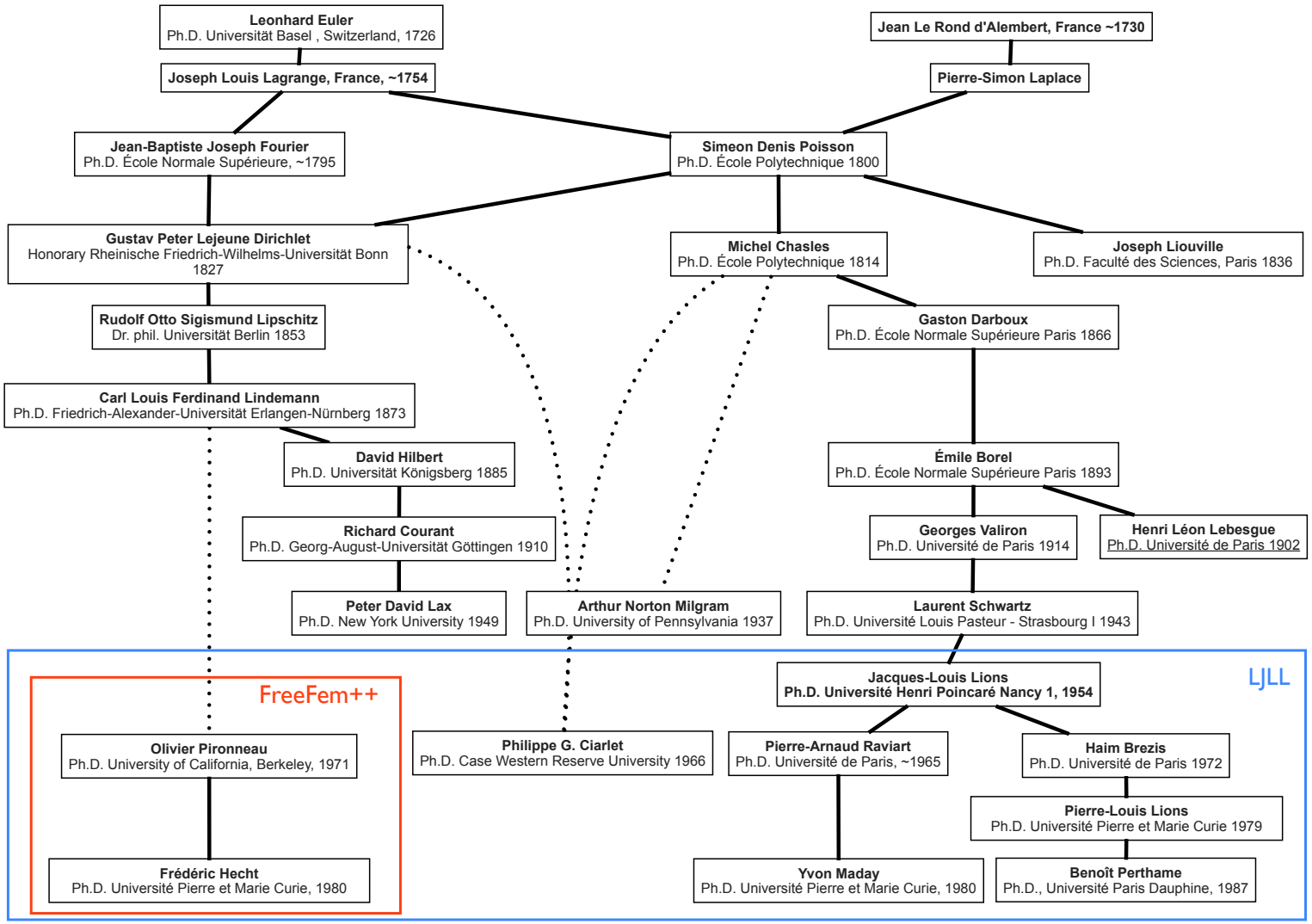
FreeFem++ is an open source partial differential equation solver. It is developed in the Laboratory Jacques-Louis Lions (LJLL) of University Pierre et Marie Curie (Paris, France) to solve PDEs.

- finite elements method, including discontinuous FE spaces;
- 2D/3D
- built on C++, syntax from C++ adapted to mathematics
- automatic mesh generator (2D/3D);
- load, save mesh;
- unstructured meshes;
- mesh adaptation (2D);
- problem definition using variational formulation
- fast linear solvers;
- mpi tools for parallel computing.

Goal

We presented a basic introduction to FreeFem++ for beginners with FreeFem++.

History++ Genealogy



Source : Mathematics Genealogy Project , http://www.genealogy.ams.org

History++

History

- 1985: MacFEM PCFEM (Pironneau)
- 1990: syntax analyzer (+ D. Bernardi) freefem
- 1995: Freefem+ (+ Hecht)
- 2000: FreeFem++ (Hecht alone)
- 2000: Freefem3D (DelPino, Hav , Pironneau)
- 2003: an integrated environment + web (Lerouzic)
- 2004 FreeFem++-cs (A. Le Hyaric)
- 2005: a new documentation (+ Ohtsuka)
- 2009: Freefem++3D
- A web site : `www.freefem.org`
- 3D Visualisation by medit (Pascal Frey)
- 2011: Parallel version with MPI

Source : O. Pironneau, F. Hecht, A. Le Hyaric

FreeFem++

- Download the current FreeFem++ from Frédéric . Hecht website : `http://www.freefem.org/ff++/`
- Full documentation and examples at `http://www.freefem.org/ff++/ftp/freefem++doc.pdf`
- FreeFem++-cs, an integrated environment with graphical interface , from Antoine Le Hyaric website : `http://www.ann.jussieu.fr/~lehyaric/ffcs/index.htm`
- Taiwan Society for Industrial and Applied Mathematics (TWSIAM) FreeFem++ activity group : `http://homepage.ntu.edu.tw/~twhsheu/twsiamff++/freefem.html`

Start using FreeFem++

Installation of FreeFem++ is really easy on Windows and MacOS platforms. On Linux, the software needs to be compiled from source code.

FreeFem++ scripts can be written in any text editor and saved to a ".edp" file. Lauching an '.edp' file depends on your operating system.

I recommend to start using FreeFem++ with FreeFem++-cs, an integrated environment for FreeFem++ (Windows, OS X, Linux). It provides a text editor with FreeFem++ syntax and highlighting of FreeFem++ compilation errors, a text output and visualisation. Developed by Antoine Lehyaric (LJLL). `http://www.ann.jussieu.fr/~lehyaric/ffcs/index.htm`

Main idea of a FE code using FreeFem++

Main steps to a PDE using the FEM

- define the domain and generate the mesh.
- define the finite element space (basis functions) for the unknowns and test functions.
- set the problem : in FreeFem++, a problem must be given in it's variational form. We need to provide a bilinear form a(u, v) , a linear form l(f,v), and possibly a Dirichlet boundary conditions.
- solve the problem.
- visualise the solution.

Meshing

It order to solve the problem on a computer, we need to discretize it. The first step of the discretization is to triangulate the physical domain Ω into the computational domain Th. We can use the keyword square to triangulate the domain and generate the mesh. We need to provide a number N of vertex placed on the borders.

square

```
mesh Th = square (N,N);
plot(Th, wait=1, ps="meshsquare.eps");
```

Note that we use the keyword plot to plot the mesh and the keyword wait to stop the program at each plot.

Finite Element Spaces

The FEM approximates all function $u(x,y) \simeq u_0\Phi_0(x,y) + ... + u_{M-1}\Phi_{M-1}(x,y)$ with finite element basis functions $\Phi_k(x,y)$.
FreeFem++ handles Lagrangian finite elements (P0,P1,P2,P3,P4), bubble elements (P1b,P2b), discontinuous P1, Raviart-Thomas, ... (see full documentation).

fespace defines the the discrete FE space for the unknowns and test functions;

$$Vh(Th, P1) = \left\{ w(x,y) | w(x,y) = \Sigma_{k=1}^{M} w_k \Phi_k(x,y), w_k \in \mathbb{R} \right\} \tag{4}$$

Declare velocity and pressure spaces

```
fespace Vh(Th,P1);
```

Declare the unknown velocity and pressure functions and the test functions

```
Vh u,v;          // unknown and test function.
```

Finite Element Spaces

- **P0** piecewise constant discontinuous finite element. The degrees of freedom are the barycenter element value.
- **P1** piecewise linear continuous finite element. The degrees of freedom are the vertices values.
- **P2** piecewise continuous quadratic finite element. The degrees of freedom are the vertices values and the middle point of each edge.
- **P3** piecewise continuous cubic finite element. (need : load "Element_P3")
- **P4** piecewise continuous quartic finite element. (need : load "Element_P4")
- **P1b** piecewise linear continuous plus bubble. [1]
- **P2b** piecewise quadratic continuous plus bubble. [1]
- **P1dc** piecewise linear discontinuous finite element.
- **P2dc** piecewise P2 discontinuous finite element.

For full list, refer to FreeFem++ documentation.

[1] Brezzi, F. & Russo, A. Choosing Bubbles for Advection-Diffusion Problems. Mathematical Models and Methods in Applied Sciences 04, 571587 (1994).

(The residual-free bubbles technique interprets the stabilization parameter as the mean value of the solution of a differential equation defined at the element level.)

Outline

P0-element

For each triangle (d=2) or tetrahedron (d=3) T_k, the basis function ϕ_k in $\mathrm{Vh}(\mathrm{Th}, \mathrm{P0})$ is given by

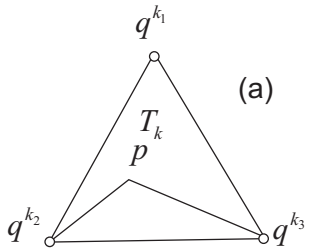
$$\phi_k(x) = 1 if (x) \in T_k, \qquad \phi_k(x) = 0 if (x) \notin T_k \tag{5}$$

If we write

```
Vh(Th,P0);
Vh fh=f(x, y);
```

then for vertices q^{k_i}, $i = 1, 2, ..d$, fh is built as

$$\mathrm{fh} = fh(x, y) = \sum_k f \left(\frac{\sum_i q^{k_i}}{d+1} \right) \phi_K \tag{6}$$



P0-element

- Download the FreeFem++ script "basis.edp" from the website `http://homepage.ntu.edu.tw/~ydeleuze/ff2016/`
- Open the script with FreeFem++-cs
- Run the script

Exercice

- plot the P0 basis function
- plot the projection of a continuous function in the P0 space.

P1-element

For each triangle (d=2) or tetrahedron (d=3) T_k, the basis function ϕ_i in $\mathrm{Vh}(\mathrm{Th}, \mathrm{P1})$ is given by

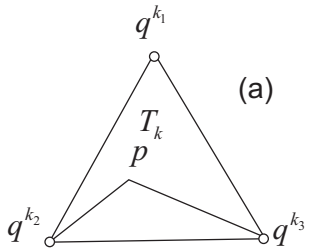
$$\phi_i(q^j) = 1, \qquad \phi_i(q^j) = 0 if i \neq j \tag{7}$$

If we write

```
Vh(Th,P1); Vh fh=g(x, y);
```

then for vertices q^{k_i}, $i = 1, 2, ..d$, fh is built as

$$\mathrm{fh} = fh(x, y) = \sum_k f\left(q^i\right) \phi_K \tag{8}$$



P1,P2,P3 elements

Exercice

- plot the P1, P2, P3, ... basis function
- plot the projection of a continuous function in the P1, P2, P3,... spaces.

Homework

Write a short report

1. compare the P0, P1, P2, P3, ... basis function and give their formulation in barycentric coordinates;
2. how many triangles (elements), vertices, degree of freedoms in a mesh ?
3. how many vertices, degree of freedoms in a triangle (element) ?
4. plot the projection of continuous functions in the P0, P1, P2, P3,... spaces;
5. Compute $\int_{T_K} \lambda_i(M)\,\lambda_j(M) dy$, $i \neq j$.

Outline

Model problem

Let us consider the elliptic problem with Dirichlet boundary condition

$$\begin{aligned} -\nabla^2 u + c\, u &= f \qquad \text{in } \Omega \\ u &= g \qquad \text{on } \partial\Omega, \end{aligned} \tag{9}$$

with c,f,g given function on Ω.

Figure: n is the unit outward normal to $\partial\Omega$

Variational formulation

To derive the variational formulation, also called weak form, of the equation, one multiplies the problem equation (9) by a smooth test function v and integrate over the bounded domain. Applying Green's identity it yields

$$\int_\Omega \nabla u \cdot \nabla v \, \mathrm{dx} - \int_{\partial\Omega} (\nabla u \cdot n)\, v \, \mathrm{d}\Gamma + \int_\Omega c\, u\, v \, \mathrm{dx} = \int_\Omega f\, v \, \mathrm{dx}. \tag{10}$$

Due to the Dirichlet boundary conditions, the finite element space to consider for u here is the H^1 Sobolev space with trace equal to g on the boundary $\partial\Omega$. Let, for any g in $H^{1/2}(\partial\Omega)$

$$V_g = \{v \in H^1(\Omega), v = g \text{ on } \partial\Omega\}. \tag{11}$$

Then, if u is a solution of the problem (9), then

$\forall v \in V_0$

$$\int_\Omega \nabla u \cdot \nabla v + c\, u\, v \, \mathrm{dx} - \int_\Omega f v \mathrm{dx} = 0 \tag{12}$$

Definition

Let the bilinear form

$$a(u, v) = \int_\Omega \nabla u \cdot \nabla v + c\, u\, v \, \mathrm{dx} \tag{13}$$

and the linear form

$$\ell(v) = \int_\Omega f v \mathrm{dx}. \tag{14}$$

Then the problem

find $u \in V_g$ such that $\forall v \in V_0$

$$a(u, v) - \ell(v) = 0$$

is called the *variational formulation* of the problem (9).

Integration by parts

Let Ω be an open compact domain of $\mathbb{R}^d$ with a smooth boundary $\partial\Omega$. Let $u, v \in \mathcal{C}^2(\bar{\Omega})$. Then

1. Gauss-Ostrogradsky theorem (divergence theorem)

$$\int_\Omega \nabla \cdot U \, \mathrm{dx} = \int_{\partial\Omega} U.n \, \mathrm{d}\Gamma \tag{15}$$

2. Integration by parts

$$\int_\Omega u \frac{\partial v}{\partial x_i} \, \mathrm{dx} = -\int_\Omega \frac{\partial u}{\partial x_i} v \, \mathrm{dx} + \int_{\partial\Omega} u \, v \, \mathrm{d}\Gamma \tag{16}$$

3. Green's first identity

$$\int_\Omega \nabla^2 u \, v \, \mathrm{dx} = -\int_\Omega \nabla u \cdot \nabla v \, \mathrm{dx} + \int_{\partial\Omega} (\nabla u \cdot n) \, v \, \mathrm{d}\Gamma \tag{17}$$

4. Green's second identity

$$\int_\Omega \nabla^2 u \, v \, \mathrm{dx} - \int_\Omega u \, \nabla^2 v \, \mathrm{dx} = \int_{\partial\Omega} ((\nabla u \cdot n) \, v - (\nabla v \cdot n) \, u) \, \mathrm{d}\Gamma \tag{18}$$

Finite element discretization

FreeFem++ uses the most common finite elements for the discretization of the continuous problem.
Consider the continuous piecewise polynomial of degree one (P_1) finite element discretization on the triangulation T_h of Ω_h

$$V_g^h = \left\{ v_h \in H^1(\Omega_h),\ v_h \in \mathcal{C}^0(\Omega_h),\ v_{h|_K} \in P_1(K), \forall K \in T_h, v_h = g \text{ on } \partial\Omega_h \right\} \tag{19}$$

The discrete problem becomes

find $u_h \in V_h$ such that $\forall v_h \in V_h^0$

$$\int_\Omega \nabla u_h \cdot \nabla v_h + \int_\Omega c\, u_h\, v_h - \int_\Omega f\, v_h = 0 \tag{20}$$

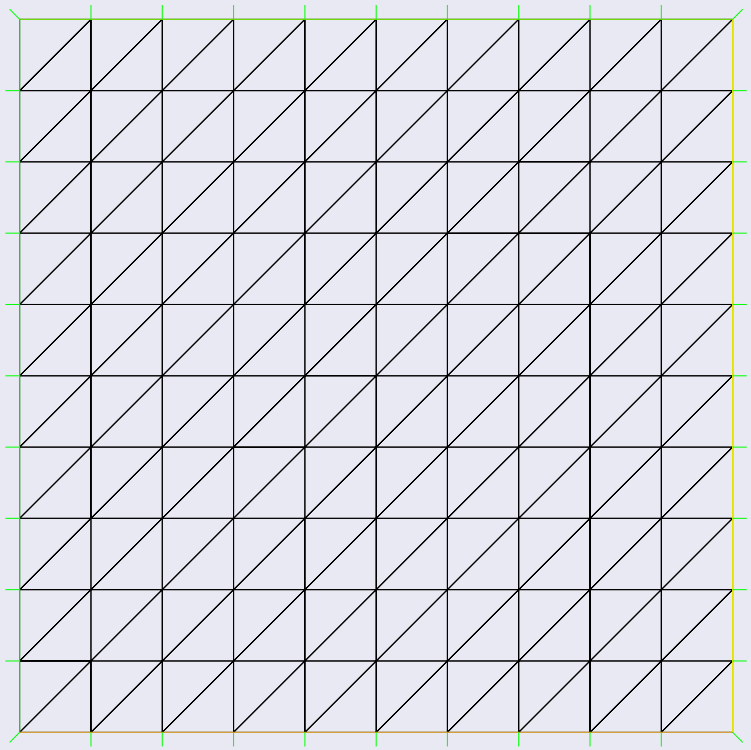
Outline

Meshing

It order to solve the problem on a computer, we need to discretize it. The first step of the discretization is to triangulate the physical domain Ω into the computational domain T_h. We can use the keyword square to triangulate the domain and generate the mesh. We need to provide a number N of vertex placed on the borders.

square

```
int N=10;
mesh Th = square (N,N);
```

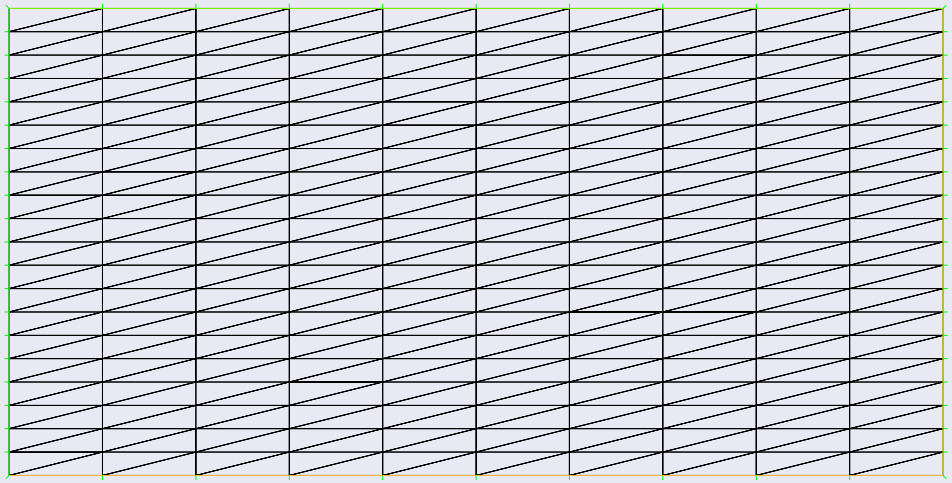


Meshing

It order to solve the problem on a computer, we need to discretize it. The first step of the discretization is to triangulate the physical domain Ω into the computational domain T_h. We can use the keyword square to triangulate the domain and generate the mesh. One needs to provide a number N of vertex placed on the borders.

rectangle

```
real x0=0, x1=10;
real y0=0, y1=5;
mesh Th=square(N,2*N,[x0+(x1-x0)*x,y0+(y1-y0)
     *y]);
plot(Th);
```



The keyword plot is used to print on screen the mesh.

Meshing : borders

In FreeFem++, the geometry of the domain can be easily defined by an analytic description of boundaries by pieces and the mesh is automatically generated. Each piece of boundary is described by it's parametric equation.
The borders are declared with the type border. The keyword label defines a group of boundaries. The label type can be either an integer or a name. The mesh is then generated by the keyword buildmesh. For is border, we need to provide a number of vertex to be placed on it.

- The border has to be well defined in the counterclockwise direction.
- The borders can not cross each others.

```
border c1(t=x0,x1){ x=t; y=y0; label=1;};
border c2(t=y0,y1){ x=x1; y=t; label=2;};
border c3(t=x1,x0){ x=t; y=y1; label=3;};
border c4(t=y1,y0){ x=x0; y=t; label=4;};
mesh Th= buildmesh(c1(N1)+c2(N2)+c3(N3)+c4(N4));
plot(Th, wait=1, ps="mesh.eps");
```

We use the keyword ps to save the plot in an eps file.

Meshing : borders

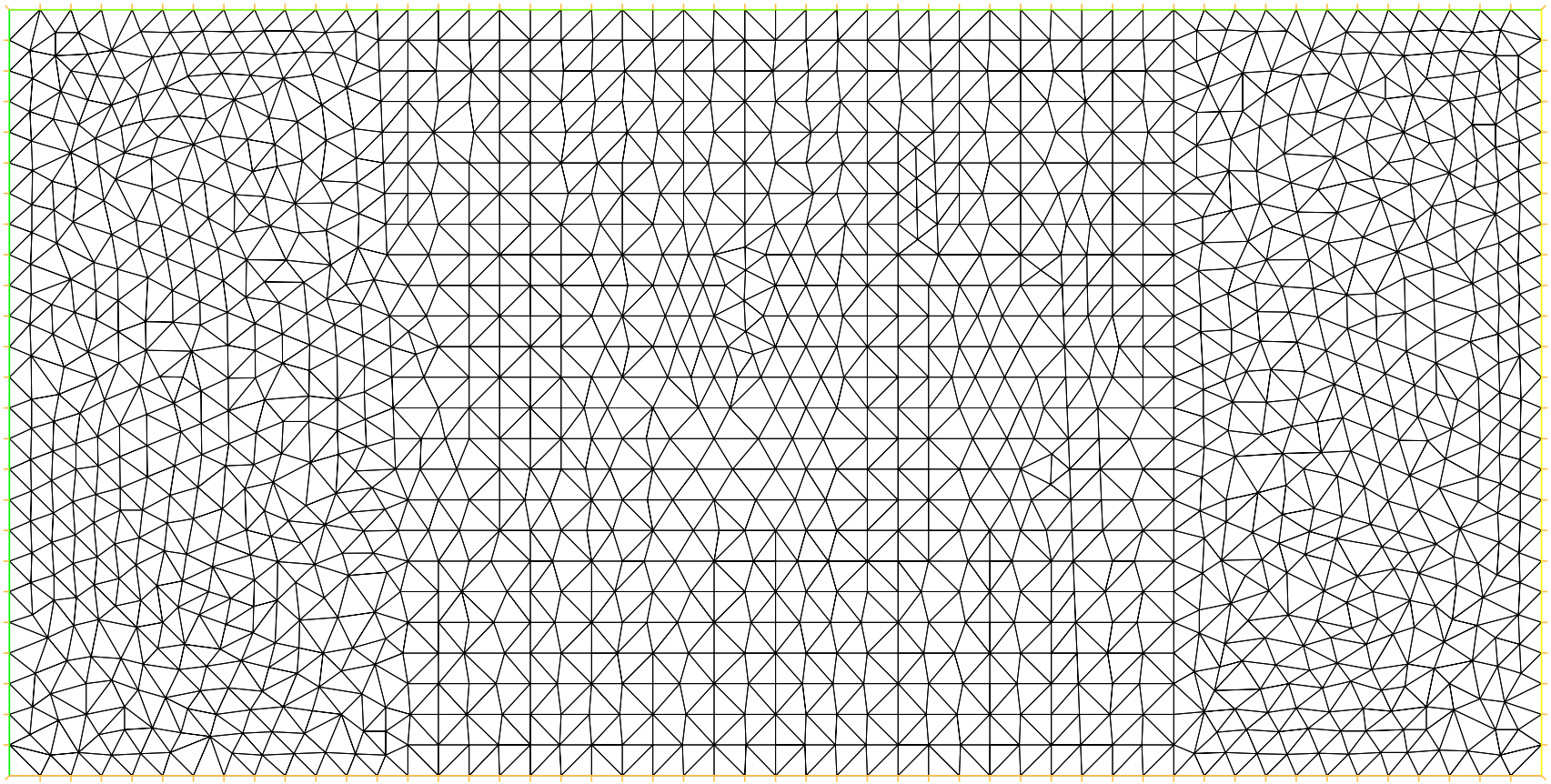


Figure: Mesh generated with border and buildmesh

Meshing : borders

```
int circle=1;
border c(t=0,2*pi){x=1+3*cos(t);y=-1+3*sin(t);label=circle;}
mesh Th=buildmesh(c(100));
```

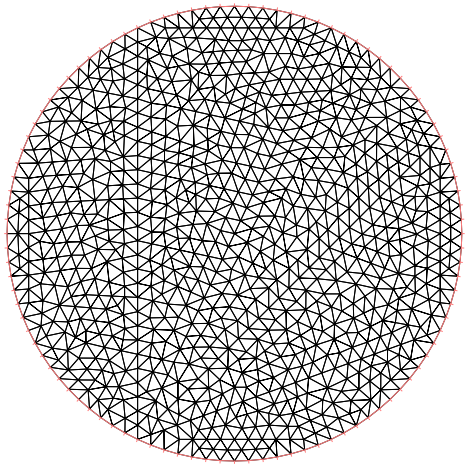


Figure: Mesh generated with border and buildmesh

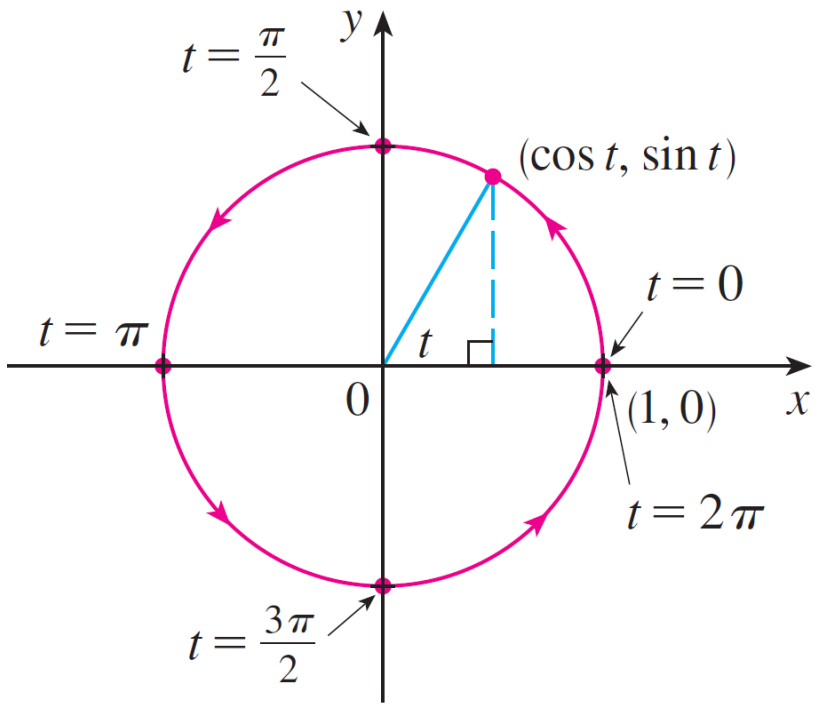
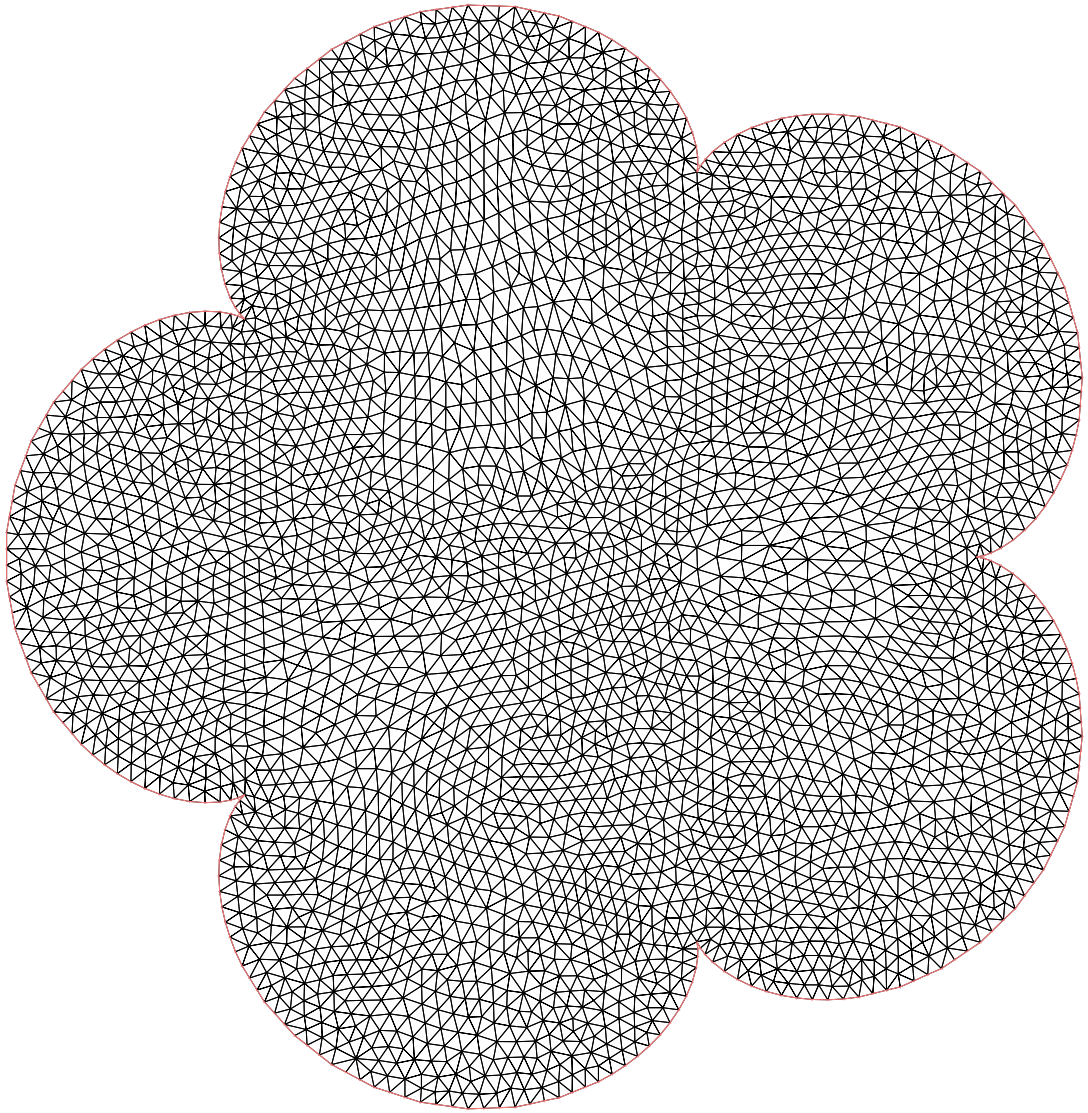


Figure: http://cims.nyu.edu/~kiryl/Calculus/Section_9.1--Parametric_Curves/Parametric_Curves.pdf

Meshing : ranunculoid

Ranunculoid mesh

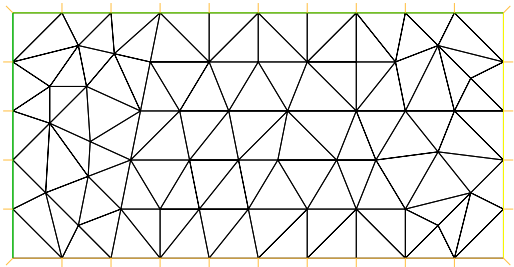


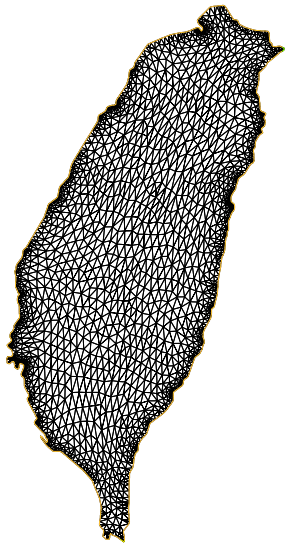
Meshing : save & load

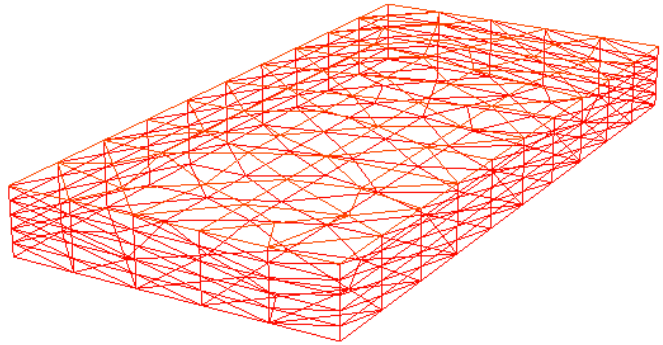
Use savemesh and readmesh to save or load a mesh file ".msh"

```
savemesh(Th,"meshfile.msh");
```

```
mesh Th = readmesh("meshfile.msh");
```





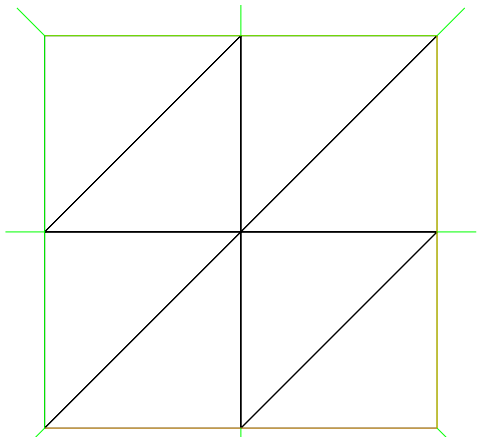


Outline

Mesh file

```
9 8 8
0 0 4
0.5 0 1
1 0 2
0 0.5 4
0.5 0.5 0
1 0.5 2
0 1 4
0.5 1 3
1 1 3
1 2 5 0
1 5 4 0
2 3 6 0
2 6 5 0
4 5 8 0
4 8 7 0
5 6 9 0
5 9 8 0
1 2 1
2 3 1
3 6 2
6 9 2
8 7 3
9 8 3
4 1 4
7 4 4
```

```
1   NbVertices NbElements NbBorderElem
2   //coordinates of vertices
3   x1 y1 label //1srt vertice
4   x2 y2 label //2nd vertice
5   ...
6   //element with 3 vertices 1,2,3
7   v1 v2 v3 region
8   v1 v2 v4 region
9   v3 v4 v5 region
10  ...
11  // boundary element is a segment of 2 vertices
12  v1 v2 label
13  v2 v3 label
14  ...
```



Mesh Connectivity

For each element i, the local vertex j corresponds to the global vertex Th[i][j] :

```
int nbtriangles=Th.nt; //total number of elements
cout << "nb of Elements = " << nbtriangles<< endl;
for (int i=0;i<nbtriangles;i++)
   for (int j=0; j <3; j++)
      cout << i << " " << j << " Th[i][j] = "
         << Th[i][j]
         << "  x = "<< Th[i][j].x
         << "  , y= "<< Th[i][j].y
         << ",  label= " << Th[i][j].label << endl;
```

Mesh Connectivity

For each vertex Th(i), access its coordinates and label :

```
int nbvertices=Th.nv; //total number of vertices
cout << " nb of vertices = " << nbvertices<< endl;
for (int i=0;i<nbvertices;i++)
        cout << "Th(" <<i << ") : "
                << Th(i).x << " " << Th(i).y
                << " " << Th(i).label
                << endl;
```

Mesh Connectivity

For each degree of freedom of the finite element space Vh, access its coordinates and value of the finite element function :

```
Vh xx = x; //get the x-coordinate of eaf dof
Vh yy = y; //get the y-coordinate of eaf dof
int nbdof=xx.n; //total numer of DOF
cout << " number of degree of freedom =" << nbdof << endl;
for (int k =0;k<nbdof;k++)     // loop on all degree of freedom
        cout << xx[][k] << " " << yy[][k] << " " << u[][k] << endl;
```

Mesh Connectivity

For each boundary element Th.be(k), the local vertex i $\in \{0, 1\}$, corresponds to the global vertex Th.be(k)[i]

```
int nbelement = Th.nbe;
cout << " nb of boundary elements = " << nbelement << endl;
for (int k=0;k<nbelement;++k)
   cout << k << " : "
   << Th.be(k)[0] << " "
   << Th.be(k)[1] << "   "
   << Th.be(k).Element //Element containing Th.be(k)
   << " , label " << Th.be(k).label << endl;
```

mesh.edp

- Download the script file `mesh.edp` from the website `http://homepage.ntu.edu.tw/~ydeleuze/ff2016/`
- Open the script with FreeFem++-cs
- Run the script with FreeFem++

Exercice

- Save a mesh in a file and read the file to understand it structure.
- Create a mesh file by hand and load it in FreeFem++ (be creative).
- Generate a circular mesh.
- Generate your own mesh using `borders` and `labels`.

Discrete variational form

In FreeFem++, the variational formulation can be declared with the keyword varf.

$$\text{varf } a(u,v) = \int_\Omega \nabla u \cdot \nabla v + c\,u\,v + (\text{boundary condition});$$
$$\text{varf } \ell(v) = \int_\Omega f\,v\ ;$$

The bilinear form and the linear form should not be written under the same integral.

$$\int_\Omega \nabla u \cdot \nabla v \rightarrow \text{int2d(Th)(dx(u)*dx(v) + dy(u)*dy(v))} \tag{21}$$

$$\int_\Omega c\,u\,v \rightarrow \text{int2d(Th)(c*u*v)} \tag{22}$$

$$\int_\Omega fv \rightarrow \text{int2d(Th)(f*v)} \tag{23}$$

where dx(u) $= \frac{\partial u}{\partial x}$, dy(u) $= \frac{\partial u}{\partial y}$. The keyword on defines the Dirichlet boundary condition on the borders 1, 2, 3 and 4.

```
// definion of the problem
varf bilinear(u,v) = int2d(Th)( grad(u)'*grad(v) + c*u*v) '//bilinear form
      + on(1,2,3,4,u=g) ; // boundary condition
varf linear(u,v) = int2d(Th)( f*v ) ; //linear form
```

Penalty method to solve Dirichlet problem

Dirichlet boundary condition

The Dirichlet boundary condition should be prescribed together with the bilinear form as in is enforced by a penalty method.

```
varf bilinear(u,v) = int2d(Th)( grad(u)'*grad(v) + c*u*v) '//
    + on(1,2,3,4,u=g) ; // boundary condition
```

modification of diagonal entries of A where index k belong to the boundary

FreeFem++ solves the linear problem of the form

$$\sum_{j=0}^{M-1} A_{ij} u_j = f_i, \quad i = 1, ..., M-1, \quad A_{ij} = a(\Phi_j, \Phi_i), f_i = \ell(\Phi_i)$$

$$\tau u_k + \sum_{k \neq j} A_{ij} u_j = f_k + \tau g_k \Leftrightarrow u_k - g_k = \frac{1}{\tau}(f_k - \sum_{k \neq j} A_{kj} u_j)$$

where τ is the penalization parameter defined in FreeFem++ as `tgv`.

Solve the problem

FreeFem++ is an elliptic solver. The user should specify it's own algorithms to solve hyperbolic and parabolic problems.

```
 Vh u;   // unknown FE function.
matrix A = bilinear(Vh,Vh,solver = LU, factorize=1); // assemble the FE matrix
real[int] bc = bilinear(0,Vh);
real[int] b = linear(0,Vh); // assemble the right hand side vector
b += bc;
u[] = A^-1*b; //solve the problem
```

The user should specify the solver to solve the linear problem. FreeFem++ provides a large variety of linear direct and iterative solvers (LU, Cholesky, Crout, CG, GMRES, multi-frontal method UMFPACK, MUMPS (MUltifrontal Massively Parallel sparse direct Solver), SuperLU, ...). Possible parameters are, for example,

solver= `LU`, `CG`, `Crout`, `Cholesky`, `GMRES`, `sparsesolver` (for MUMPS, SuperLU), `UMFPACK`. The default solver is GMRES.
The storage mode of the matrix of the underlying linear system depends on the type of solver chosen (see documentation of the use of each solver).

factorize = `false`, by default; `true` needed for the matrix factorization for LU, Cholesky or Crout.

elliptic.edp

```
mesh Th=square(2,2);

fespace Vh(Th,P1);       // P1 FE space

func f=-1;                  // right hand side function
func g=0;          // boundary condition function
func c=1;
macro grad(u) [dx(u),dy(u)] //eom - IMPORTANT : macro ends with a comments

varf bilinear(u,v) =  int2d(Th)( grad(u)'*grad(v) + c*u*v) //variational formulation
       + on(1,2,3,4,u=g) ; // boundary condition
varf linear(u,v) =  int2d(Th)( f*v ) ;

matrix A = bilinear(Vh,Vh,solver = LU, factorize=1); // assemble the FE matrix
real[int] bc = bilinear(0,Vh);
real[int] b = linear(0,Vh); // assemble the right hand side vector
Vh u;
b += bc;
u[] = A^-1*b; //solve the problem

plot(u,ps="solution.eps",value=1, wait=1, fill=1, cmm="Solution u in \Omega");
```

Outline

Workflow of a FreeFem++ script

Generate the mesh Th

```
mesh Th = ... ;
```

Define the finite element space Vh

```
fespace Vh (Th, P1);
```

Define the variational problem P

```
varf B(u,v) = a(u,v) + (boundary condition);
varf L(u,v) = l(f,v) ;
```

Solve the problem P

```
matrix A = P(Vh,Vh);
matrix bc = P(0,Vh);
real[int] b = P(0,Vh); b += bc;
u[]= A^-1*b;
```

Visualize the solution u

```
plot(u);
```

FreeFem++ syntax

```
x,y,z   // Cartesian coordinates
N.x, N.y, N.z //Normal vector components
int k = 10; // integer
real a=2.5; // real
bool b=(a<3.); /boolean
real [int] array(k); // array of k elements
array[][5]; //6th value of the array
mesh Th; //2d mesh
mesh3 Th3 //3d mesh
fespace Vh(Th,P1); //finite element space
Vh u=x; //finite element function
Vh3<complex> uc = x+ 1.i *y; //complex finite element function
fespace Xh(Th,[P2,P2,P1]);
Xh [u1,u2,p]; // a vectorial finite element function or array
u[]; //the array associated to FE function u
u(1.,0.1,3.); //value of u at point (1.,0.1,3.)
u[].max; // max of the array u
u[].min; // max of the array u
u[].sum; //sum of all elements of u
u[].l1; // l1 norm of u
u[].l2; // l2 norm of u
u[].linfty; // linfinity norm of u
macro div(u,v) (dx(u)+dy(v))// EOM
macro Grad(u) [dx(u),dy(u)]// EOM
func f=x+y; //function
func real f(int i, real a) { .....; return i+a;}
varf a([u1,u2,p],[v1,v2,q])= int2d(...) + on(..)
matrix A;
```

Operators

FreeFem++ provides many operators

- dx(u), dy(u), dxx(u), dyy(u), dxy(u)
- convect(a,dt,u)
- sin(u), exp(u), ...
- int1d(u), int2d(u)

but you can make your own

- macro div(u,v) (dx(u)+dy(v)) //EOM

Exercise : Write macros to enhance your codes.

Input-Output in FreeFem++

The user can import/export data from FreeFem++ such as meshes files, postscript images, text files. Other format are handle with FreeFem++ to interact with third parties softwares.

Terminal

The syntax write/read in the terminal is similar to C++ with the keywords cin, cout, <<, >>, endl.

```
int nn ;
cout << "number of nodes on the borders nn= " << endl; //writing in the terminal
cin >> nn; //read value from the terminal
mesh Tht=square(nn,nn);
```

Files

To write/read a file, first the user need to declare a variable with ofstream (output) or ifstream (input). Then, the syntax for input and output in a file remains the same.

```
ifstream fin("data.txt"); //declare an input from file
real ff, fg;
fin >> ff; //reads the 1srt value from the file
fin >> fg; //reads the 2nd value from the file
ofstream fout("solution.txt");
fout << ff << " "<< fg << endl; //write in file
```

Mesh

To import/export mesh the function readmesh and savemesh are available. Refer to the documentation for full list of format one can import in FreeFem++.

Finite element function (array of value)

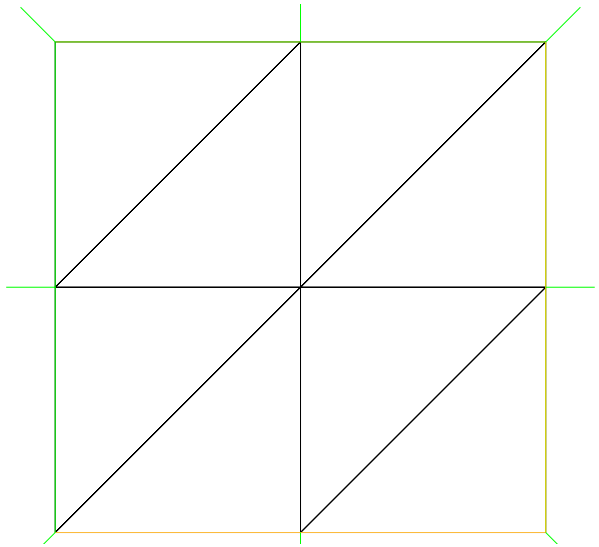
To import/export the solution of a finite element function we use the following syntax

```
ofstream fout("solution.txt");
fout << u[]; //export FE solution
ifstream fu("solution.txt");
fu >> u4plot[];} //import a FE solution
```

FE solution file

```
9
        6.25e-62        6.25e-32        6.25e-62        6.25e-32        0.0625
        6.25e-32        6.25e-62        6.25e-32        6.25e-62
```

```
NbVertices
        u(0)  u(1)  u(2)
        u(3)  u(4)  ...
        ...         u(NbVertices-1)
```



Visualization

Use the keyword plot to plot the solution with useful options :

- ps= string to save the plot in a eps file;
- cmm = string to add comment
- if value= 1 plot the value of isolines
- if fill=1 color fill between isolines
- if wait=1 stop the program at this plot

```
plot(u,ps="Laplace.eps",value=1, wait=1, fill=1, cmm="Solution u of the Laplace equation"
    );
```

elliptic.edp

- Download the script file `elliptic.edp` from the website `http://homepage.ntu.edu.tw/~ydeleuze/ff2016/`
- Open the script with FreeFem++-cs
- Run the script with FreeFem++

Exercice

- Change the mesh with your own mesh and edit the script accordingly.
- Edit the functions `f`, `g`, and `c`.
- Change the solver.

Outline

Error estimate

Let $\Omega \subset \mathbb{R}^n$, $n = 2, 3$.

Definition

$$H^m(\Omega) = \{v \in L^2(\Omega),\, \partial^\alpha v \in L^2(\Omega), |\alpha| \leq m\}$$

- scalar product: $(u, v)_{m,\Omega} = \int_\Omega \sum_{|\alpha| \leq m} \partial^\alpha \, \partial^\alpha v \, \mathrm{dx}$
- norm : $||u||_{m,\Omega} = (u, u)_{m,\Omega}^{\frac{1}{2}}$

Definition

$$H^1(\Omega) = \{v \in L^2(\Omega),\, \frac{\partial v}{\partial x_i} \in L^2(\Omega), 1 \leq i \leq n\}$$

- scalar product: $(u, v)_{1,\Omega} = \int_\Omega u\, v + \sum_{1 \leq i \leq n} \frac{\partial u}{\partial x_i} \, \frac{\partial v}{\partial x_i} \, \mathrm{dx}$
- norm : $||u||_{1,\Omega} = (u, u)_{1,\Omega}^{\frac{1}{2}}$

Error estimate

Let T_h be a triangulation of Ω.
$h_K = diam(K)$, $h = max_{K \in T_h} h_k$.
$\prod_h : C(\Omega) \to V_h$, $\prod_h u = \sum_{1 \leq i \leq N} u(P_i)\varphi_i$, where $\{\varphi_i\}_{1 \leq i \leq N}$ are the P_k finite element basis.

Theorem (Interpolation error)

$K \in T_h$, $k \geq 1$.
It exists a constant $C > 0$, that depends only on the dimension ($n = 2, 3$) and k such that for all integer m, $0 \leq m \leq k+1$, there is

$$\forall v \in H^{k+1}(K),\ |v - \prod v|_{m,K} \leq C\, h_K^{k+1}\, |v|_{k+1}, K$$

Theorem (finite element error)

Let $u \in H^{k+1}$ and u_h finite element solution with P_k element.

- *The finite element method is convergent*

$$\lim_{h \to 0} ||u - u_h||_{1,\Omega} = 0.$$

- *It exists $C > 0$, independant of h, such that*

$$||u - u_h||_{1,\Omega} \leq C\, h^k\, |u|_{k+1}, K$$

$u \in H^{k+1}$: exact solution

u_h finite element solution with P_k element.

$h_K = diam(K)$, $h = max_{K \in T_h} h_k$.

$$||u - u_h||_{1,\Omega} = c\, h^k$$

$$\frac{||u - u_{h_1}||_{1,\Omega}}{||u - u_{h_2}||_{1,\Omega}} = \frac{c\, {h_1}^k}{c\, {h_1}^k} = \left(\frac{h_1}{h_2}\right)^k$$

numerical convergence order

$$k = \log\left(\frac{||u - u_{h_1}||_{1,\Omega}}{||u - u_{h_2}||_{1,\Omega}}\right) / \log\left(\frac{h_1}{h_2}\right)$$

numerical convergence order

$$k = \log\left(\frac{||u - u_{h_1}||_{1,\Omega}}{||u - u_{h_2}||_{1,\Omega}}\right) / \log\left(\frac{h_1}{h_2}\right)$$

order.edp

- Download the script file `order.edp` from the website `http://homepage.ntu.edu.tw/~ydeleuze/ff2016/`
- Open the script with FreeFem++-cs
- Run the script with FreeFem++

$$-\nabla^2 u + c\,u = f \qquad \text{in } \Omega = [0,1] \times [0,1]$$
$$u = 0 \qquad \text{on } \partial\Omega,$$

An exact solution to this Dirichlet boundary problem is

$$u(x,y) = \sin(\pi x) * \sin(\pi y)$$

with $f = (2\pi^2 + c)\sin(\pi x) * \sin(\pi y)$.

Exercice

- Change the mesh with your own mesh and edit the script accordingly.
- Change the finite element space and compare the error estimation.

Homework

Write a short report and provide the code used.

- Elliptic problem with Dirichlet boundary condition in $\Omega = [0,1] \times [0,1]$
  1. Print the ROC curves in log-scale of $||u_h - u||_{1,\Omega})$ function of h and $|u_h - u|_{2,\Omega}$ function of h in a structured and unstructured domain.
  2. Give the order of convergence for P_1, P_2, P_3 finite elements.
- Elliptic problem with Robin boundary condition

$$\begin{aligned} -\nabla^2 u + c\, u &= f \qquad \text{in } \Omega \\ \frac{\partial u}{\partial n} + \varepsilon u &= 0 \qquad \text{on } \partial\Omega, \end{aligned} \tag{24}$$

  1. Write the variational formulation of (24).
  2. Solve (24) using the solver of your choice.
  3. Plot solutions for different values of parameters.

Outline

Outline

Heat problem

Freefem++ is able to solve time indecent problems but it is the user's responsibility to provide the algorithms.

$$\begin{cases} \frac{\partial u}{\partial t} - \nabla \cdot (\kappa \nabla u) = 0 \text{ in } (0,T) \times \Omega \\ u_{|_{t=0}} = U_0 \text{ in } \Omega \\ \nabla u \cdot n = U_N \text{ on } (0,T) \times \Gamma_N \\ u = U_D \text{ on } (0,T) \times \Gamma_D \end{cases} \tag{25}$$

Time discretization

First, let us consider a time discretization of the heat equation (25) . Time is discretize with finite difference schemes such as

θ - scheme [Raviart-Thomas 1998]

We discretize time $(0, T)$ in $M-1$ intervals $\Delta t = \frac{T}{M-1}$ and M points $(t_m)_{m=0}^{M-1}$ such that $U(t_m, x) = U^m(x)$.

$$\frac{\partial U}{\partial t} \approx \frac{U^{n+1} - U^n}{\Delta t} = \theta F(U^{n+1}) + (1-\theta) F(U^n)$$

- Euler explicit ($\theta = 0$)
- Euler implicit ($\theta = 1$)
- Crank-Nicolson ($\theta = \frac{1}{2}$)

Time discretization

Let us choose the Euler implicit quadrature. The heat problem in its semi-discrete form becomes

$$\frac{u^{n+1} - u^n}{\Delta t} - \nabla \cdot (\kappa \nabla u^{n+1}) = 0 \tag{26}$$

We multiply the Laplace's equation by a smooth test function v and integrate over the entire domain. We apply Green's identity and it gives

$$\int_\Omega \frac{1}{\Delta t} u^{n+1}\, v + \int_\Omega \kappa \nabla u^{n+1} \nabla v - \int_{\partial\Omega} \kappa (\nabla u^{n+1} \cdot n)\, v - \int_\Omega \frac{1}{\Delta t} u^n\, v = 0 \tag{27}$$

Due to the Dirichlet boundary conditions we choose v such that $v_{|\partial\Omega} = 0$ on Γ_D and we use the Neumann boundary condition on Γ_N

$$\int_\Omega \frac{1}{\Delta t} u^{n+1}\, v + \int_\Omega \kappa \nabla u^{n+1} \nabla v - \int_{\partial\Gamma_N} \kappa\, U_N\, v - \int_\Omega \frac{1}{\Delta t} u^n\, v = 0 \tag{28}$$

Spatial discretization

Let $V_g = \{v \in H^1(\Omega), v = g \text{ on } \Gamma_D\}$.
We can rewrite it, find $u^{n+1} \in V_{U_D}$ such that $\forall v \in V_0$

$$a(u, v) = \ell(v) \tag{29}$$

with

- $a(u, v) = \int_\Omega \frac{1}{\Delta t} u^{n+1} v + \int_\Omega \kappa \nabla u^{n+1} \nabla v$ the bilinear form,
- and $\ell(v) = \int_{\partial\Gamma_N} \kappa U_N v + \int_\Omega \frac{1}{\Delta t} u^n v$ the linear form.

Lets consider the continuous piecewise polynomial of degree one (P_1) finite element discretization on the triangulation T_h of Ω_h.
The discrete problem becomes

find $u_h^{n+1} \in V_h$ such that $\forall v_h \in V_h^0$

$$a(u_h^{n+1}, v_h) = l(v_h) \tag{30}$$

Heat equation on the unit square

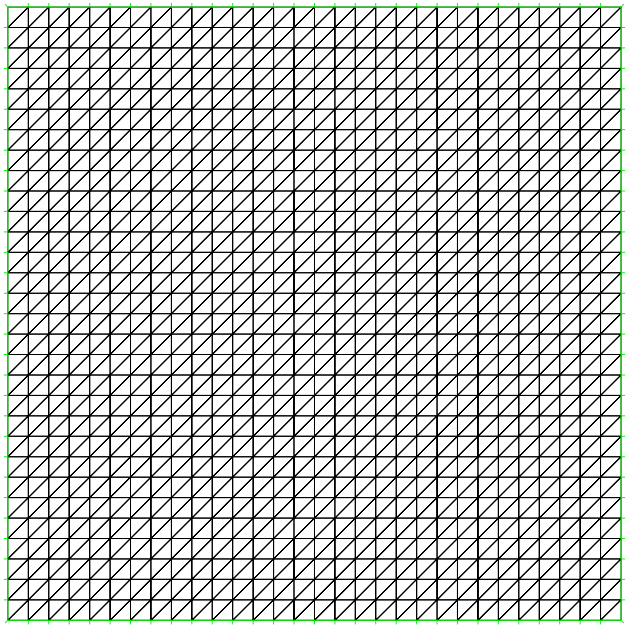
Let's consider the problem

$$\begin{cases} \frac{\partial u}{\partial t} - \nabla \cdot (\kappa \nabla u) = 0 \text{ in } (0, T) \times \Omega, \\ u_{|_{t=0}} = \sin(\pi x) \sin(\pi y) \text{ in } \Omega, \\ u = 0 \text{ on } (0, T) \times \Gamma_D = \partial\Omega. \end{cases} \tag{31}$$

Then, if $\Omega = [0, 1] \times [0, 1]$, the exact solution is

$$u(x, y, t) = \sin(\pi x) \sin(\pi y) e^{-2\pi^2 t} \tag{32}$$

Generating mesh



```
int CD=10; // could be anything
real x0=0, x1=1;
real y0=0, y1=1;
int nn=30;
mesh Ths=square(nn,nn,[x0+(x1-x0)*x,y0+(y1-y0)*y]);
int[int] labels=[1,CD,2,CD,3,CD,4,CD]; // change the labels from 1,2,3,4 to CD
Ths=change(Ths,label=labels);
plot(Ths, wait=1, ps="square.eps", cmm="square mesh");
```

Defining the problem

Parameters

```
int M=10; //time iterations
real dt = 0.01; //time step
```

Finite element space

```
fespace Vhs(Th,P1);
```

Initial condition

```
u = sin(pi*x)*sin(pi*y);
```

Variational problem

```
varf bilinear(u,v) = int2d(Th)(u*v/dt) + int2d(Th)(F(u,v))
                   + on(CD,u=0)
varf linear(unused,v) = int2d(Th)(up*v/dt) ;
```

Solving the system

Time loop

```
for(i=0;i<M; i++) {
        up=u; //update from previous step time
        ... //solve the linear system
        plot(u,wait=0,fill=1,value=1,cmm="solution at time "+i*dt, viso=viso); // plot the
            solution at each step time
}
```

Exercice

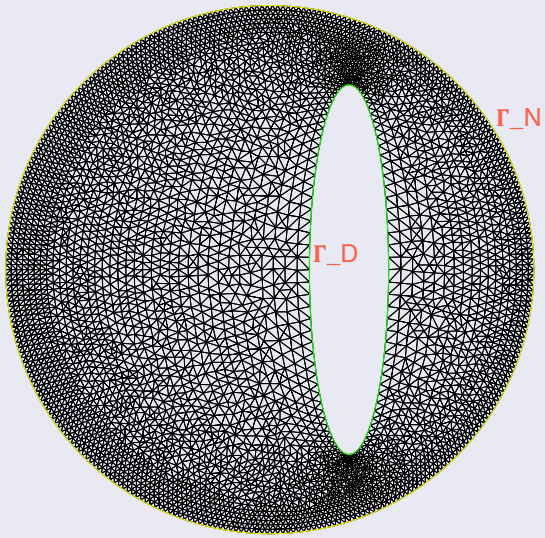
- Download the script `heat.edp` and run it with FreeFem++.
- Update the script to implement the Crank-Nicolson scheme.
- Verify your solution with the exact solution in the H^1 norm.

Homework

Write a short report and provide the code used.

- Heat equation with Dirichlet and Neumann boundary condition

$$\begin{cases} \frac{\partial u}{\partial t} - \nabla \cdot (\kappa \nabla u) = 0 \text{ in } (0, T) \times \Omega, \\ u_{|_{t=0}} = U_D = 20 \text{ in } \Omega, \\ \nabla u \cdot n = U_N = -1 \text{ on } (0, T) \times \Gamma_N, \\ u = U_D = 20 \text{ on } (0, T) \times \Gamma_D. \end{cases} \tag{33}$$



1. Write the variational formulation of (33) with the Crank-Nicolson scheme or any higher order scheme of your choice.
2. Solve (33) using the solver of your choice.
3. Plot solution at different time

Outline

Convection-Diffusion Equation

The stationary convection-diffusion equation describes the steady-state behavior of a convective-diffusive system. It describes the physical phenomena where particles or other physical quantities are transferred inside a physical system due to two processes: diffusion and convection.

$$\begin{aligned} \nabla \cdot (\mathbf{u}\phi) - \nabla \cdot (\nu \nabla \phi) &= 0 && \text{in } \Omega, \\ \phi &= \phi_D && \text{on } \Gamma_D \\ \nabla \phi \cdot \mathbf{n} &= \mathbf{u} \cdot \mathbf{n} && \text{on } \Gamma_N. \end{aligned} \tag{34}$$

where $u = (u_1, u_2)$ is a fluid velocity vector, such that $\nabla \cdot \mathbf{u} = 0$ in Ω and ν is the diffusivity.

Variational problem

To solve a problem in FreeFem++, the problem must be given in it's variational form. Let $V_\varphi = \{w \in H^1(\Omega) | w = \varphi \text{ on } \Gamma_D\}$.
We multiply (34) by a test function $v \in V_0$ and integrate over the entire domain. Then the variational formulation of the convection-diffusion problem is :

find $\phi \in V_{\phi_D}$ such that $\forall v \in V_0$:

$$a(\phi, v) = 0 \tag{35}$$

where the bilinear form

$$a(\phi, v) = \int_\Omega -\phi \mathbf{u} \cdot \nabla v + \nu \nabla \phi \cdot \nabla v.$$

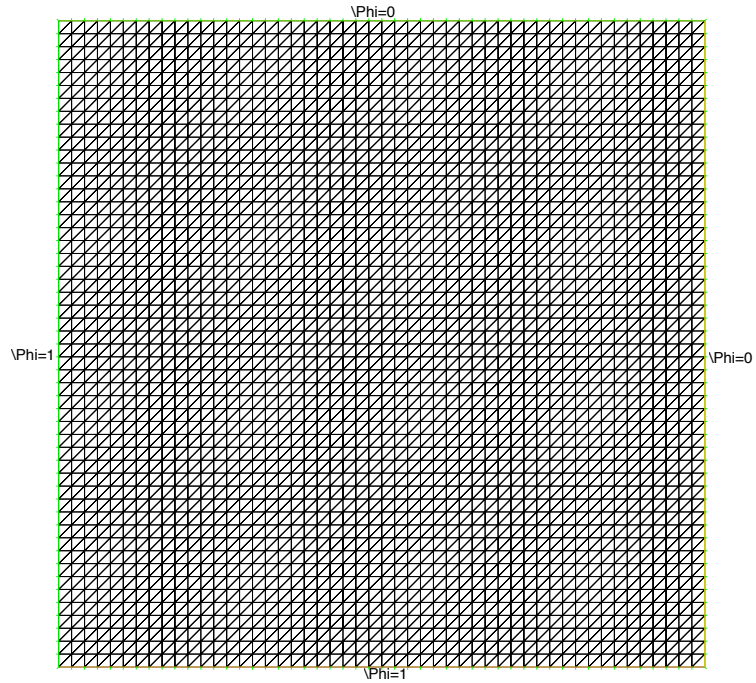


Figure: $\Omega = (0,1) \times (0,1)$, $\Gamma_D = \partial\Omega$

Numerical approximation

Finite element method requires the discretization of the problem in a finite dimensional subspace $V_h \subset V$. The problem becomes find the numerical solution $\phi_h \in V_h$ such that $\forall v_h \in V_h$:

$$\int_\Omega -(\mathbf{u} \cdot \nabla) v_h \phi_h + \nu \nabla \phi_h . \nabla v_h = 0 \tag{36}$$

```
fespace Vh(Th,P1);
varf CD(phi,v) =
        int2d(Th)  ( nu * ( dx(phi)*dx(v) + dy(phi)*dy(v) ) )
       -int2d(Th) ( (u * dx(v) + v * dy(v) ) * phi )
       +on(1,4,phi=1)+on(2,3,phi=0);
```

Convection dominated problem

When the problem is convection dominated, spurious oscillations can appear. The local Péclet number on each element K of the mesh T_h is given by

$$Pe_K = \frac{||u||_{L^\infty(K)} h_K}{2\nu}$$

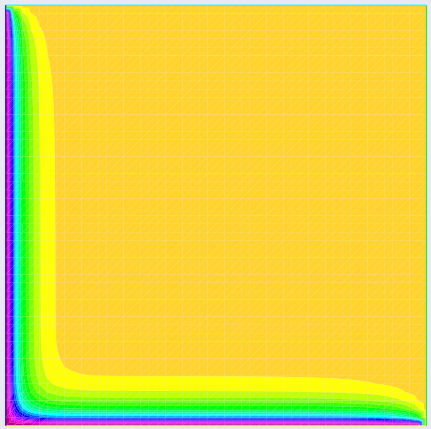


Figure: Solution Φ of (35) for $Pe_K = 0.42$. The problem is diffusion dominant and the numerical solution can be computed by the standard Galerkin linear finite element method.

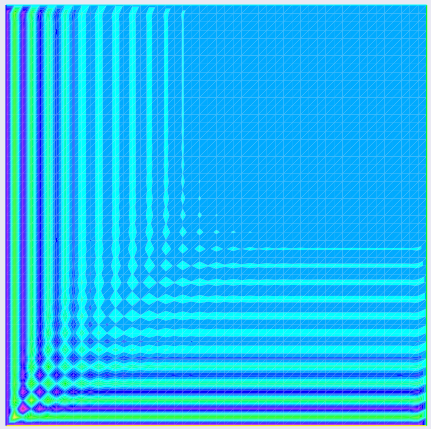


Figure: Solution Φ of (35) by a non stabilized Galerkin method when $Pe_K = 14$. The problem is convection dominant and the solution is wrong due to the spurious oscillations.

Spurious oscillations appear when $Pe_K > 1$. To avoid spurious oscillations, we adopt a stabilized Galerkin method. We will explore two methods :

- Artifical viscosity
- Streamline Upwind-Petroc Galerkin (SUPG)

Convection dominated problem

Figure: French physicist Jean Claude Eugène Péclet (10 Feb 1793 - 6 Dec 1857)

Péclet number on element K

$$Pe_K = \frac{||u||_{L^\infty(K)} h_K}{2\nu}$$

```
1  fespace Xh(Th,P0);
2  Xh hK = hTriangle;
3  Xh Pe = Uinf * hK / 2. / nu;
```

where hTriangle $= diam(K) = \{sup|x-y|; x, y \in K\}$

Method 1 : Artificial Viscosity

Artificial viscosity method introduces an isotropic numerical artificial viscosity to the problem. The bilinear form becomes

$$a_h(\phi_h, v_h) = a(\phi_h, v_h) + \sum_{K \in T_h} ||u||_{L^\infty(K)} h_K \int_K \nabla \phi_h \nabla v_h \tag{37}$$

This formulation is consistent as $h = \max_K(h_k) \to 0$.
The local Péclet number becomes

$$Pe_K = \frac{||u||_{L^\infty(K)} h_K}{2\nu + ||u||_{L^\infty(K)} h_K} \xrightarrow{||u||_{L^\infty(K)} \to \infty} 1^-$$

Exercise

- Download, read, and run `ExConvectionDiffusion.edp`.
- Implement the artificial viscosity method to deal with the spurious oscillations.

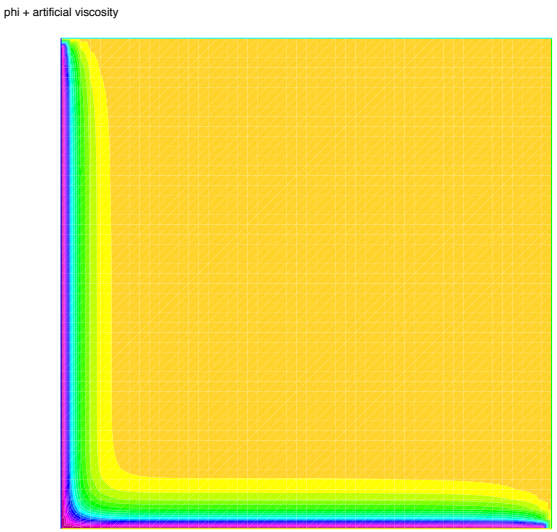


Figure: Solution Φ of (37) by a stabilized numerical artificial viscosity Galerkin method when $Pe_K = 14$. The problem is convection dominant and the solution shows no spurious oscillations.

Formaggia, L., Saleri, F. & Veneziani, A. in Solving Numerical PDEs: Problems, Applications, Exercises 147202 (Springer Milan, 2012).

Method 2 : SUPG

The streamline upwind-Petrov Galerkin method introduce an numerical artificial viscosity only in the direction of convective term.

$$a_h(\phi_h, v_h) = a(\phi_h, v_h) + \sum_{K \in T_h} \alpha_K \int_K \left(\tau_K \left(\frac{1}{2} u.\nabla v_h + \frac{1}{2} \nabla.(u v_h) \right) \right) (-\nu \Delta \phi_h + \nabla.(u \phi_h)), \quad (38)$$

with $\tau_K = \frac{h_K}{|u|_{L^2(K)}}$.

This formulation is strongly consistent because the added viscosity vanishes when applied to the exact solution since

$$-\Delta \phi + \nabla.(u\phi) = 0.$$

Exercise

- Implement the SUPG method.
- Compare with the artificial viscosity method.

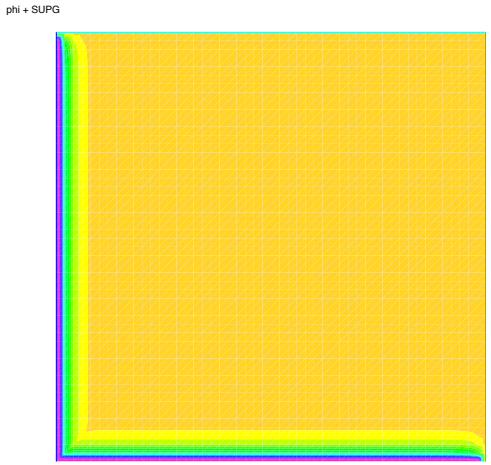


Figure: Solution Φ of (38) by a streamline upwind-Petrov Galerkin method when $Pe_K = 14$. The problem is convection dominant and the solution is shows no spurious oscillations.

Formaggia, L., Saleri, F. & Veneziani, A. in Solving Numerical PDEs: Problems, Applications, Exercises 147202 (Springer Milan, 2012).

Method 3 : SUPG (2)

The streamline upwind-Petrov Galerkin method introduce an numerical artificial viscosity only in the direction of convective term.

$$a_h(\phi_h, v_h) = a(\phi_h, v_h) + \sum_{K \in T_h} \alpha_K \int_K \left(\tau_K \left(\frac{1}{2} u.\nabla v_h + \frac{1}{2} \nabla.(u v_h) \right) \right) (-\Delta \phi_h + \nabla.(u \phi_h)) \,, \qquad (39)$$

Let us now consider another definition $\tau_K = \frac{h_K \nu}{2||u||_{L^2(K)}}$.

Exercise

- Implement the SUPG method with the new definition of τ_K.
- Compare with the previous results.

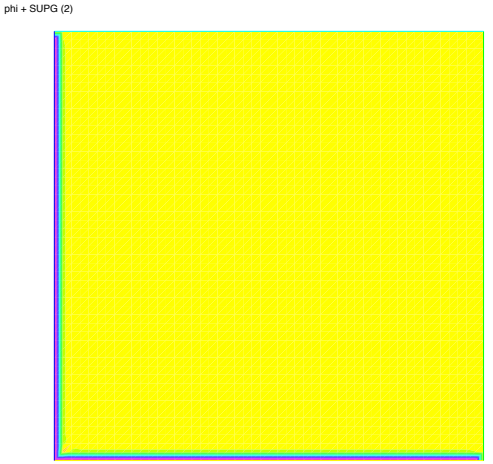


Figure: Solution Φ of (39) by a streamline upwind-Petrov Galerkin method when $Pe_K = 14$. The problem is convection dominant and the solution is shows no spurious oscillations.

Akira, M. An Implementation of the Streamline-Upwind/Petrov Galerkin Method for Linear Triangular Elements. Computer Methods in Applied Mechanics and Engineering 49, 357364 (1985).

Artificial Viscosity vs SUPG

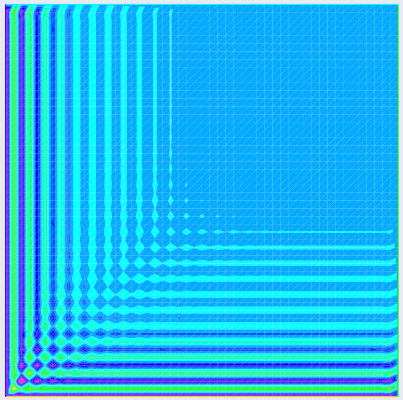


Figure: non stabilized Galerkin

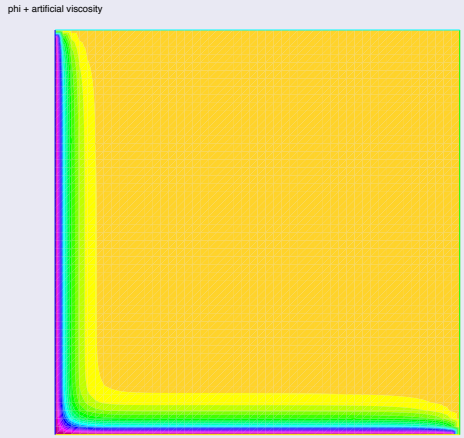


Figure: Artificial viscosity

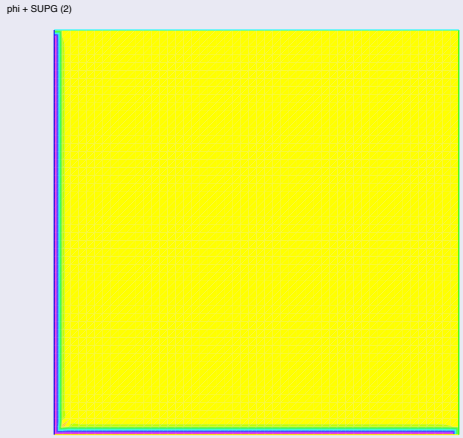


Figure: SUPG

The drawback of this artificial viscosity stabilization is to introduce viscosity in all the directions. SUPG is less diffusive. We can notice how the boundary layer is steeper.

Outline

Stationary Stokes equations

When an incompressible flow is dominated by viscous effect and is closed to a steady state one can consider the stationary Stokes equations. The dimensionless Stokes equations then write

$$\begin{aligned} -\frac{1}{\mathrm{Re}}\nabla^2\mathbf{u} + \nabla p &= \mathbf{f}, \\ \nabla \cdot \mathbf{u} &= 0, \end{aligned} \tag{40}$$

where $\mathbf{u} = (u_1, u_2)$ is the velocity vector and p the pressure, and Re is the Reynolds number. ∇^2, here, stands for the vector Laplacian. In Cartesian coordinates, $\nabla^2 u = (\nabla^2 u_1, \nabla^2 u_2)$.

Stationary Stokes equations

As an example, let's consider an approximation of the Stokes equations, namely the pseudo-compressible Stokes equations, on the smooth spatial domain $\Omega \in \mathbb{R}^2$, with a boundary split up into three borders :

$$
\begin{aligned}
-\frac{1}{\mathrm{Re}}\nabla^2\mathbf{u} + \nabla p &= \mathbf{f} && \text{in } \Omega, \\
\nabla \cdot \mathbf{u} + \varepsilon p &= 0, && \\
\mathbf{u} &= g && \text{on } \Gamma_1, && \leftarrow \text{ inlet flow Dirichlet b.c.} \\
\mathbf{u} &= 0 && \text{on } \Gamma_{2-3}, && \leftarrow \text{ wall boundary no-slip Dirichlet b.c.} \\
-\frac{1}{\mathrm{Re}}\boldsymbol{\nabla}\mathbf{u}\cdot\mathbf{n} + p\cdot\mathbf{n} &= 0 && \text{on } \Gamma_4. && \leftarrow \text{ traction-free outflow b.c.}
\end{aligned}
\tag{41}
$$

Variational problem

To solve a problem in FreeFem++, let us write the variational formulation of the problem. We multiply the first equation by a smooth test function $v = (v_1, v_2)$, we get after applying the Green's formula

$$\frac{1}{\mathrm{Re}} \int_\Omega \nabla u_i \nabla v_i - \frac{1}{\mathrm{Re}} \int_{\partial\Omega} \nabla u_i \cdot n \, v_i - \int_\Omega p \partial_i v_i + \int_{\partial\Omega} p \, n_i v_i - \int_\Omega f_i v_i = 0. \tag{42}$$

And we multiply the second equation by a smooth function q and integrate over Ω. We get

$$\int_\Omega \nabla \cdot \mathbf{u} \, q + \int_\Omega \varepsilon p q = 0. \tag{43}$$

We consider the space $V_\phi = \left\{(\mathbf{w}, r) \in [H_0^1(\Omega)]^2 \times L^2(\Omega) | w_{|\Gamma_1} = \phi, w_{|\Gamma_2} = 0\right\}$ associated with the norm

$$||(\mathbf{u}, p)||_V = \left(||u_1||^2_{H^1,\Omega} + ||u_2||^2_{H^1,\Omega} + |p|^2_{L^2,\Omega} \right)^{1/2}$$

Let $(\mathbf{u}, p) \in V_g$ and $(\mathbf{v}, q) \in V_0$, then (42) reduces to

$$\frac{1}{\mathrm{Re}} \int_\Omega \nabla u_i \nabla v_i - \int_\Omega p \partial_i v_i - \int_\Omega f_i v_i = 0, \;\; i = 1, 2, \tag{44}$$

Variational problem

The variational problem is

find $(\mathbf{u}, p) \in V_g$ such that for all $(\mathbf{v}, q) \in V_0$

$$\frac{1}{\mathrm{Re}} \int_\Omega (\nabla u_1 \nabla v_1 + \nabla u_2 \nabla v_2) - \int_\Omega p(\partial_x v_1 + \partial_y v_2) + \int_\Omega (\partial_x u_1 + \partial_y u_2)\, q + \int_\Omega \varepsilon p q = \int_\Omega (f_1 v_1 + f_2 v_2) \tag{45}$$

Mixed formulation

$(\mathbf{u}_h, p_h) \in V_g^h = V \times Q$, $V_g^h \subset V_g$.

Taylor-Hood Elements

V : P2 finite element
Q : P1 finite element

```
fespace Vh(Th,[P2,P2,P1]);
```

Bubble Elements

V : P1b finite element
Q : P1 finite element

```
fespace Vh(Th,[P1b,P1b,P1]);
```

find $(\mathbf{u}_h, p_h) \in V_g^h$ such that for all $(\mathbf{v}, q) \in V_0^h$

$$a((\mathbf{u}_h, p_h), (\mathbf{v}_h, q_h)) = \ell((\mathbf{v}_h, q_h)). \tag{46}$$

Implementation of the Stokes problem

```
real L=10;
border b1(t=1,0)          {x=0;          y=t;          label=1;}
border b2(t=0,L)          {x=t;          y=0;          label=2;}
border b3(t=0,1)          {x=L;          y=t;          label=3;}
border b4(t=L,0)          {x=t;          y=1;          label=4;}
real Dx=0.1, Dy=0.05;
mesh Th= buildmesh(b1(ceil(1./Dy))+b2(ceil(L/Dx))+b3(ceil(1./Dy))+b4(ceil(L/Dx)));

fespace Vh(Th,[P2,P2,P1]);
fespace Ph(Th,P1); // for plots
Vh [u,v,p]; //FE solution

macro grad(u) [dx(u),dy(u)] //eom ¡- IMPORTANT : macro ends with a comments

varf bilinear ([u1,u2,p],[v1,v2,q]) = ... ;
varf linear([u1,u2,p],[v1,v2,q]) =  ... ;

matrix A = bilinear(Vh,Vh,solver = CG, factorize=0); // assemble the FE matrix
real[int] bc = bilinear(0,Vh); // Dirichlet boundary condition
real[int] b = linear(0,Vh); // assemble the right hand side vector
b += bc; // modifying the entires for the Dirichlet BC.
u[] = A^-1*b; //solve the problem

Ph pu = u, pv = v; // interpolate on P1 for plots
plot(coef=0.03,cmm=" Flow field [u,v]",[pu,pv], value=1, wait=1);
plot(cmm=" Pressure p",p, value=1, fill=1, wait=1);
Ph V = sqrt(u*u+v*v);
plot(cmm=" Fluid velocity magnitude V",V, value=1, fill=1, wait=1);
```

Implementation of the Stokes problem

Exercise

1. From the variational formulation, identify the bilinear and linear parts.
2. Download the script `ExStokes.edp` and write the FF++ code to solve the Stokes problem

$$
\begin{aligned}
-\frac{1}{\mathrm{Re}}\nabla^2\mathbf{u} + \nabla p &= \mathbf{f} && \text{in } [0, L]\times[0,1],\\
\nabla\cdot\mathbf{u} + \varepsilon p &= 0, &&\\
\mathbf{u} &= (y(1-y), 0) && \text{on } \Gamma_1,\\
\mathbf{u} &= (0,0) && \text{on } \Gamma_{2-4},\\
-\frac{1}{\mathrm{Re}}\boldsymbol{\nabla}\mathbf{u}\cdot\mathbf{n} + p\cdot\mathbf{n} &= (0,0) && \text{on } \Gamma_3.
\end{aligned}
$$

3. Analytic validation in the square $[0,1]\times[0,1]$ with the $||\cdot||_V$ norm according to the analytical solution

$$u_1 = y(1-y),\ u_2 = 0,\ p = \frac{2}{\mathrm{Re}}(1-x),\ f = 0.$$

   Give the error for different values of $h = \max_K h_K$ for the P2-P1 and P1b-P1 elements.
4. Compare the time used by the solvers CG, Crout, UMFPACK using the function `real t = clock();` (Homework: plot the time as function of the number of degree of freedom).

Penalty type stabilized formulation

$(\mathbf{u}_h, p_h) \in V_g^h = V \times Q$, $V_g^h \subset V_g$.

V : P1 finite element Q : P1 finite element

```
fespace Vh(Th,[P1,P1,P1]);
```

find $(\mathbf{u}_h, p_h) \in V_g^h$ such that for all $(\mathbf{v}, q) \in V_0^h$

$$a((\mathbf{u}_h, p_h), (\mathbf{v}_h, q_h)) + \delta s(p_h, q_h) = \ell((\mathbf{v}_h, q_h)). \tag{47}$$

where $\delta > 0$ and

$$s(p, q) = \sum_{K \in T_h} h_K^2 \int_K \nabla p \cdot q. \tag{48}$$

Exercise

- Implement the P1-P1 stabilized method to solve the Stokes equations.
- Compare the solution with the analytical solution in $\Omega = [0, 1]^2$.

Homework (1)

Use the code from the exercices. Write a small report with comments and discussion on the results and provide the code used.

$$\begin{aligned} \nabla \cdot (\mathbf{u}\phi) - \nabla \cdot (\nu \nabla \phi) = 0 \quad & \text{in } [0,1] \times [0,1], \\ \phi = 1 \quad & \text{on } \Gamma_1 \cup \Gamma_4 \\ \phi = 0 \quad & \text{on } \Gamma_2 \cup \Gamma_3 \end{aligned}$$

with $\mathbf{u} = (-u_{max}, -u_{max})$.

1. Write the variational formulation of the convection-diffusion equation.
2. Plot the solution for different value of $h = \frac{1}{N_K} \sum_K h_k$ of the convection-diffusion equation using the non-stabilized method, the artificial viscosity method and the SUPG methods in the case $Pe = \max_K(Pe_K) < 1$ and $Pe > 1$.
3. In the case, $Pe < 1$, compare the solutions from each stabilized method with the non-stabilized method using the H^1 norm.

Homework (2)

Use the code from the exercices. Write a small report with comments and discussion on the results and provide the code used.

1. Write the variational formulation of the Stokes equations

$$
\begin{aligned}
-\frac{1}{\mathrm{Re}}\nabla^2\mathbf{u} + \nabla p &= \mathbf{f} && \text{in } [0, L]\times[0,1],\\
\nabla\cdot\mathbf{u} + \varepsilon p &= 0, &&\\
\mathbf{u} &= (y(1-y), 0) && \text{on } \Gamma_1,\\
\mathbf{u} &= (0,0) && \text{on } \Gamma_{2-4},\\
-\frac{1}{\mathrm{Re}}\boldsymbol{\nabla}\mathbf{u}\cdot\mathbf{n} + p\cdot\mathbf{n} &= (0,0) && \text{on } \Gamma_3.
\end{aligned}
$$

2. Analytic validation in the square $[0,1]\times[0,1]$ with the $||\cdot||_V$ norm according to the analytical solution

$$u_1 = y(1-y),\ u_2 = 0,\ p = \frac{2}{\mathrm{Re}}(1-x),\ f = 0.$$

   Plot the error for different values of $h = \max_K h_K$ for the P2-P1, P1b-P1, and P1-P1 elements.

3. Solve the Stokes equation in the domain of your choice with the boundary condition of your choice. Give the variational formulation and a plot of the solution.

Outline

Convection equation in $\Omega \subset \mathbb{R}^d$, $d = 1, 2, 3$

$$\frac{\partial u}{\partial t} + \mathbf{a} \cdot \nabla u = f, \quad \text{on } \Omega \times (0, T) \tag{49}$$

The convection equation can be discretized in time with the one step of backward convection by the method of Characteristics-Galerkin

$$\frac{u^{n+1}(\mathbf{x}) - u^n(\mathbf{X}^n)}{\Delta t} = f^{n+1}(\mathbf{x}). \tag{50}$$

where $\mathbf{X}^n$ is an approximation, at time t^n of the ODE

$$\begin{cases} \dfrac{d\mathbf{X}}{dt} = u(\mathbf{X}(t)) \\ \mathbf{X}(t^{n+1}) = \mathbf{x} \end{cases} \tag{51}$$

Convection

Let $t^{n+1} = t^n + \Delta t$, $u^n(\mathbf{x}) = u(\mathbf{x}, t^n)$, and $\mathbf{x} = \mathbf{X}(t^{n+1})$. On the one hand, applying Taylor's expansion for $\tau \mapsto u^n(\mathbf{X}(\tau))$, then

$$u^n(\mathbf{X}(t^n)) = u^n(\mathbf{x}) - \Delta t\, \mathbf{a}(\mathbf{x}) \cdot \nabla u^n(\mathbf{x}) + \circ(\Delta t) \tag{52}$$

And on the other hand, applying Taylor's expansion for $\tau \mapsto u^n(\mathbf{x} - \mathbf{a}^n(\mathbf{x})\,\tau)$, $0 \leq \tau \leq \Delta t$ then

$$u^n(\mathbf{x} - \mathbf{a}^n(\mathbf{x})\,\Delta t) = u^n(\mathbf{x}) - \Delta t\, \mathbf{a}^n(\mathbf{x}) \cdot \nabla u^n(\mathbf{x}) + \circ(\Delta t). \tag{53}$$

FreeFem++ provides an interpolation operator convect$(a^n, -\Delta t, u^n) \approx u^n(\mathbf{X}^n(\mathbf{x}))$ where

- Euler scheme: $\mathbf{X}^n = \mathbf{x} - \mathbf{a}^n(\mathbf{x})\,\Delta t$

Characteristics-Galerkin method

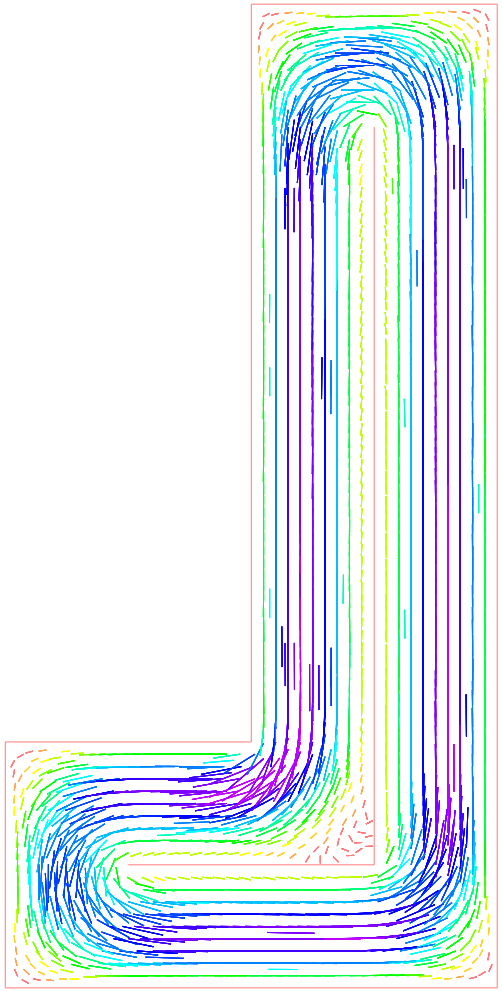
Using the Euler scheme, for $f = 0$, the discretized problem becomes

$$u^{n+1} = u^n(\mathbf{X}^n) \tag{54}$$

FreeFem++

```
u = convect([a1,a2],-dt,u)
```

Pure convection



Generating the mesh

```
real Lx = 10;
real Ly = 20;

real[int] A(2), B(2), C(2), D(2), E(2), F(2), G(2), H(2), I(2);

A=[0,0]; B=[Lx,0]; C=[Lx,Ly]; D=[Lx/2.,Ly];
E=[Lx/2.,Ly/4.]; F= [0,Ly/4.];
G=[Lx/4.,Ly/8.]; H=[3*Lx/4.,Ly/8.]; I=[3*Lx/4.,7*Ly/8.];

border l1(t=0,1){x=(1-t)*A[0]+t*B[0]; y=(1-t)*A[1]+t*B[1]; label=0;}
border l2(t=0,1){x=(1-t)*B[0]+t*C[0]; y=(1-t)*B[1]+t*C[1]; label=0;}
border l3(t=0,1){x=(1-t)*C[0]+t*D[0]; y=(1-t)*C[1]+t*D[1]; label=0;}
border l4(t=0,1){x=(1-t)*D[0]+t*E[0]; y=(1-t)*D[1]+t*E[1]; label=0;}
border l5(t=0,1){x=(1-t)*E[0]+t*F[0]; y=(1-t)*E[1]+t*F[1]; label=0;}
border l6(t=0,1){x=(1-t)*F[0]+t*A[0]; y=(1-t)*F[1]+t*A[1]; label=0;}

border i1(t=0,1){x=(1-t)*G[0]+t*H[0]; y=(1-t)*G[1]+t*H[1]; label=0;}
border i2(t=0,1){x=(1-t)*H[0]+t*I[0]; y=(1-t)*H[1]+t*I[1]; label=0;}

int nn=4;
mesh Th=buildmesh(l1(nn*Lx)+l2(nn*Ly)+l3(Lx/2.*nn)+l4(3.*Ly/4.*nn)+l5(Lx/2.*nn)+l6(Ly/4.*
    nn)+i1(Lx/2.*nn)+i2(Ly*6./8.*nn));
```

Generating the velocity field

$$
\begin{aligned}
-\frac{1}{\mathrm{Re}}\nabla^2\mathbf{u} + \nabla p &= (0, x - \frac{Lx}{2}) \qquad \text{in } \Omega, \\
\nabla \cdot \mathbf{u} + \varepsilon p &= 0, \\
\mathbf{u} &= (0,0), \qquad \text{on } \partial\Omega,
\end{aligned}
$$

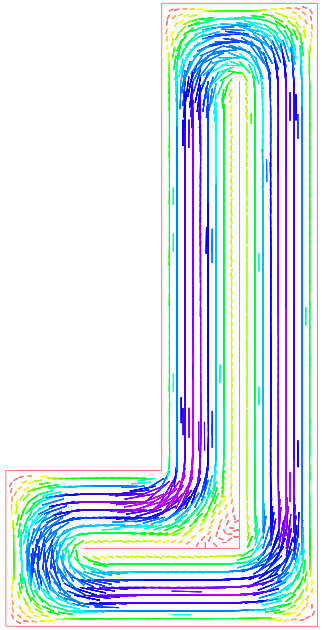


Figure: Velocity field $a = [a_2, a_2]$

Pure convection

```
Uh u = (sqrt((x-Lx/2.)^2+(y-Ly/16.)^2)<Ly/32.); //initial solution
real masse0 = int2d(Th)(u); //initial mass
cout << "initial mass M(0)=" << masse0 << endl;
real [int] viso(0.:0.025:1.); //define fixed isolines
plot(u, viso=viso, fill=1, wait=1,value=1);

real time = 0; //current time
real dt=0.2; // time step
real itmax=500; // max iterations

for (int it=0; it<itmax; it++){
        u = convect([a1,a2], -dt, u);
        time += dt;
        plot(u, viso=viso, value=1, fill=1, cmm="u("+time+")");
}
```

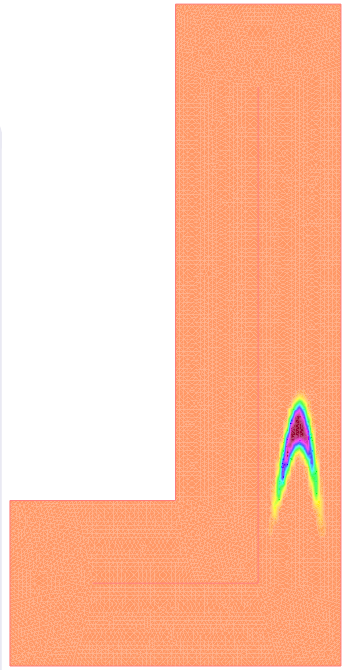


Figure: Velocity field $a = [a_2, a_2]$

Exercise

$$\frac{\partial c}{\partial t} + \mathbf{u} \cdot \nabla c = f, \qquad \text{on } \Omega \times (0, T) \tag{55}$$

- Download, read, and run `Exconvect.edp`.
- Generate the velocity field solving the Stokes equations

$$\begin{aligned} -\frac{1}{\mathrm{Re}} \nabla^2 \mathbf{u} + \nabla p &= (0, x - \frac{Lx}{2}) && \text{in } \Omega, \\ \nabla \cdot \mathbf{u} + \varepsilon p &= 0, \\ \mathbf{u} &= (0, 0) && \text{on } \partial\Omega. \end{aligned}$$

- Check the evolution of the total mass of the system $M(t) = \int_\Omega c \, \mathrm{dx}$.
- Try to decrease the spatial resolution h, the time stp Δt, or increase the order of the method.

The method is not conservative. Refer to the FreeFem++ documentation for other methods (Discontinuous Galerkin, finite volume method).

Outline

Navier-Stokes equations

An incompressible viscous fluid satisfies in Ω the Navier-Stokes equations:

$$\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p - \nu \nabla^2 \mathbf{u} = 0, \tag{56}$$

$$\boldsymbol{\nabla} \cdot \mathbf{u} = 0, \tag{57}$$

$$\mathbf{u}_{|_{t=0}} = \mathbf{u}^0, \tag{58}$$

$$\mathbf{u}_{|_{\Gamma}} = u_{\Gamma}. \tag{59}$$

where $\mathbf{u} = (u_1, u_2)$ is the velocity vector and p the pressure. $\nu = \frac{1}{Re}$.

Variational problem

To solve a problem in FreeFem++, the problem must be given in it's variational form. We multiply by a test function $v \in V = \{(\mathbf{u}, p) \in H_0^1(\Omega)^2 \times L^2(\Omega) | div(w) = 0\}$ and and integrate over the entire domain. Then the weak form of the NS problem is

find $(\mathbf{u}, p) \in V$ such that $\forall (\mathbf{v}, q) \in V$

$$\int_\Omega \partial_t \mathbf{u}.\mathbf{v} + ((\mathbf{u}.\nabla)\mathbf{u}).\mathbf{v} + \nu \nabla \mathbf{u} : \nabla \mathbf{u} - p\nabla \cdot \mathbf{v} + q\nabla \cdot \mathbf{u} = 0 \tag{60}$$

Discretization

First, we generate the mesh Th associated to the physical domain Ω.

square

```
mesh Th=square(25,25);
```

Then, we define the FE spaces. The velocity is approximated with the P_2 FE space, and the the pressure is approximated with the P_1 FE space.

```
fespace Vh(Th,[P2,P2,P1]);
```

up1 and up2 are used to save the solution u at the previous time step.

Discrete variational form

FreeFem++ provides an interpolation operator convect for the $\partial_t u + ((u.\nabla)u)$. We can rewrite the problem giving the discretization in time :

find $(u^{n+1}, p^{n+1}) \in (Vh, Ph)$ such that $\forall (v_h, q_h) \in (Vh, Ph)$:

$$\int_\Omega \frac{1}{dt}(\mathbf{u}^{n+1} - \mathbf{u}^n(\mathbf{X}^n))\mathbf{v} + \nu \nabla \mathbf{u} : \nabla \mathbf{u} - p\nabla \cdot \mathbf{v} + q\nabla \cdot \mathbf{u} + \varepsilon p\, q = 0 \tag{61}$$

The term $u^n(X^n)$ will be computed by the operator convect. The term $\varepsilon p\, q$ is added for the stabilization.

The user can provide the solver to solve the linear problem associated. FreeFem++ provides a large variety of linear direct and iterative solvers (CG, GMRES, UMFPACK, MUMPS, SuperLU, ...).

Remark: To save computational time, the mass matrix associated with the problem should not be assembled at each time step.

Solving and visualization

The problem of the driven cavity is time dependent. We implement a loop in time to solve the problem at each step time i. We store the solution [u1,u2] from the previous step time in the variables [up1,up2].

We can plot the solution at each time step using wait=0 giving the impression of a video. The keyword coef allows to resize the vector size in the plot.

```
/*       Solving the problem      */
matrix A = bilinear(Vh,Vh,solver = UMFPACK, factorize=0); // assemble the FE matrix
real[int] bc = bilinear(0,Vh); // Dirichlet boundary condition

/*       time iterations */
for (int ite=0;ite<Nite;ite++) {
        [up1,up2,upp] = [u1,u2,p];
        real[int] b = linear(0,Vh); // assemble the right hand side vector
        b += bc; // modifying the entires for the Dirichlet BC.
        u1[] = A^-1*b; //solve the problem
        plot(coef=0.2, wait=0, cmm=" [u1,u2] and p ",p,[u1,u2],value=1);
};
```

Solving and visualization

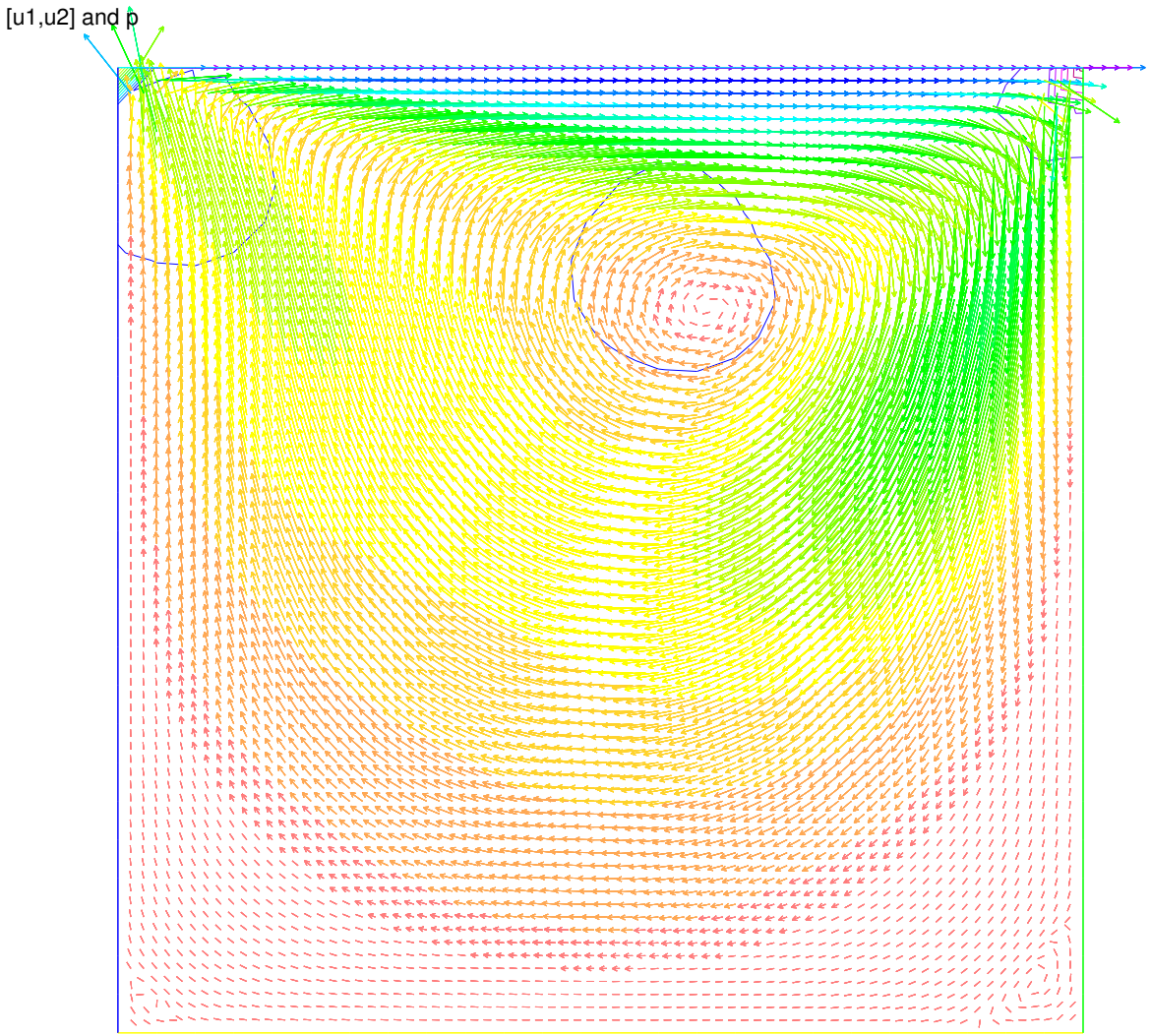


Figure: Solution $u = (u_1, u_2)$ and p at a given time t.

Exercise : Navier-Stokes

- Dowload and complete the FreeFem++ script
  - `ExNScavity.edp`
- Solve the Navier-Stokes equation and visualize and describe the velocity field and the pressure.
- Run the simulations for different Reynolds number ($Re = 1$ to $Re = 1500$) and mesh size.

Exercise : Navier-Stokes

- Explore and run the FreeFem++ scripts
  - `MeshAneurysm.edp`
  - `MeshBifurcation.edp`
  - `MeshStenose.edp`
  - `MeshStair.edp`
  - `MeshTube.edp`

  to create meshes of 5 study case : aneurysm, stenosis, blood vessel bifurcation, bend, stair case. Meshes are saved in a file with the command savemesh.
- Solve the Navier-Stokes equation on each of the 5 generated meshes. Use the readmesh function to load the meshes in your main script. Adapt you variational formulation with inlet and outlet boundary conditions.
- Run the simulations for different Reynolds number ($Re = 1$ to $Re = 1500$) and mesh size.
- Verify the incompressible fluid condition: how much fluid is going in and out of the domain ?

Temporary end of the story

Yannick Deleuze
http://homepage.ntu.edu.tw/~ydeleuze/
@thechicanosweb