Éléments Finis: applications (FreeFem++)

M. Ersoy

Ecole d'ingénieurs de l'Université de Toulon
MOdélisation et CAlculs Fluides et Structures (MOCA)
2A

OUTLINE OF THE TALK

Outline

1. Introduction
2. Examples of linear and non linear PDEs
   - Poisson equation
   - Heat equation
   - Convection problem
   - Advection-diffusion problem
   - Incompressible Navier-Stokes equations
   - many among others
3. Conclusion

A software for solving PDEs

FreeFem++ for 2D-3D [1] PDEs

- Finite element method

1. in progress

A software for solving PDEs

FreeFem++ for 2D-3D[1] PDEs

- Finite element method
- It is an easy install Free software with a well documented, see `http://www.freefem.org/ff++/ftp/freefem++doc.pdf` based on command line interface available for
  - Linux
  - Windows
  - Mac

  operating system

1. in progress

A software for solving PDEs

FreeFem++ for 2D-3D [1] PDEs

- Finite element method
- It is an easy install Free software with a well documented, see `http://www.freefem.org/ff++/ftp/freefem++doc.pdf` based on command line interface available for
  - Linux
  - Windows
  - Mac

  operating system
- ++ means extension of FreeFem and FreeFem+ (see Historic `http://www.freefem.org/ff++/ftp/HISTORY`) initially designed by Olivier Pironneau and Frédéric Hecht

1. in progress

A software for solving PDEs

FreeFem++ for 2D-3D [1] PDEs

- Finite element method
- It is an easy install Free software with a well documented, see http://www.freefem.org/ff++/ftp/freefem++doc.pdf based on command line interface available for
  - Linux
  - Windows
  - Mac

  operating system
- ++ means extension of FreeFem and FreeFem+ (see Historic http://www.freefem.org/ff++/ftp/HISTORY) initially designed by Olivier Pironneau and Frédéric Hecht
  To install it, please visit http://www.freefem.org/ff++/

1. in progress

Characteristics of Freefem++

- Several finite element : linear, quadratic, discontinuous P1, . . .

Characteristics of Freefem++

- Several finite element : linear, quadratic, discontinuous P1, ...
- Automatic interpolation from one mesh to an other

Characteristics of Freefem++

- Several finite element : linear, quadratic, discontinuous P1, ...
- Automatic interpolation from one mesh to an other
- Able to read/save mesh

Characteristics of Freefem++

- Several finite element : linear, quadratic, discontinuous P1, ...
- Automatic interpolation from one mesh to an other
- Able to read/save mesh
- Automatic mesh refinement tools isotropic as well as anisotropic

Characteristics of Freefem++

- Several finite element : linear, quadratic, discontinuous P1, ...
- Automatic interpolation from one mesh to an other
- Able to read/save mesh
- Automatic mesh refinement tools isotropic as well as anisotropic
- Analytic description of the boundary

Characteristics of Freefem++

- Several finite element : linear, quadratic, discontinuous P1, ...
- Automatic interpolation from one mesh to an other
- Able to read/save mesh
- Automatic mesh refinement tools isotropic as well as anisotropic
- Analytic description of the boundary
- Problem definition based on the weak variational form of the PDEs

Characteristics of Freefem++

- Several finite element : linear, quadratic, discontinuous P1, . . .
- Automatic interpolation from one mesh to an other
- Able to read/save mesh
- Automatic mesh refinement tools isotropic as well as anisotropic
- Analytic description of the boundary
- Problem definition based on the weak variational form of the PDEs
- Efficient use of linear algebra : LU, CG, GMRES, UMFPACK, ARPACK, . . .

Characteristics of Freefem++

- Several finite element : linear, quadratic, discontinuous P1, ...
- Automatic interpolation from one mesh to an other
- Able to read/save mesh
- Automatic mesh refinement tools isotropic as well as anisotropic
- Analytic description of the boundary
- Problem definition based on the weak variational form of the PDEs
- Efficient use of linear algebra : LU, CG, GMRES, UMFPACK, ARPACK, ...
- Export data result to post-treatment with medit, gnuplot, tecplot, ...
- Written and use the same C++ syntaxes

How to run a Freefem++ file

1. Edit a file with the extension ".edp" or ".pde" or ".c" with your favorite text editor

How to run a Freefem++ file

1. Edit a file with the extension ".edp" or ".pde" or ".c" with your favorite text editor
2. To run the code on command line :
   `FreeFem++ file.edp`

How to run a Freefem++ file

1. Edit a file with the extension ".edp" or ".pde" or ".c" with your favorite text editor
2. To run the code on command line :
   `FreeFem++ file.edp`
3. Or with the X-version and click
   `run`
4. Each line end with ;

Data

Data

- `x`, `y` and `z` : devoted to the point (x, y, z)

Data

Data

- `x`, `y` and `z` : devoted to the point (x, y, z)
- `label` : devoted to the part of the boundary

Data

Data

- `x`, `y` and `z` : devoted to the point (x, y, z)
- `label` : devoted to the part of the boundary
- `N` : devoted to the Normal to a boundary point

Data

Data

- `x`, `y` and `z` : devoted to the point (x, y, z)
- `label` : devoted to the part of the boundary
- `N` : devoted to the Normal to a boundary point
- `cin`, `cout ,` `endl` : devoted to get, print and end line used with $\ll$ and $\gg$

Data

Data

- `x`, `y` and `z` : devoted to the point (x, y, z)
- `label` : devoted to the part of the boundary
- `N` : devoted to the Normal to a boundary point
- `cin`, `cout`, `endl` : devoted to get, print and end line used with $\ll$ and $\gg$
- `pi`, `true`, `false`, `i` are explicit

Data& Types

Data

- x, y and z : devoted to the point (x, y, z)
- label : devoted to the part of the boundary
- N : devoted to the Normal to a boundary point
- cin, cout, endl : devoted to get, print and end line used with $\ll$ and $\gg$
- pi, true, false, i are explicit

Type

- Operators as in C langage : + - * ^ == < > <= >= & = +=

Data& Types

Data

- `x`, `y` and `z` : devoted to the point (x, y, z)
- `label` : devoted to the part of the boundary
- `N` : devoted to the Normal to a boundary point
- `cin`, `cout`, `endl` : devoted to get, print and end line used with $\ll$ and $\gg$
- `pi`, `true`, `false`, `i` are explicit

Type

- Operators as in C langage : `+ - * ^ == < > <= >= & = +=`
- `int`, `real`, `complex`, `string`, ...

Data& Types

Data

- x, y and z : devoted to the point (x, y, z)
- label : devoted to the part of the boundary
- N : devoted to the Normal to a boundary point
- cin, cout, endl : devoted to get, print and end line used with $\ll$ and $\gg$
- pi, true, false, i are explicit

Type

- Operators as in C langage : `+ - * ^ == < > <= >= & = +=`
- int, real, complex, string, ...
- `real[int] a(n);`
  `a[3]=2;`
- `real[int,int] a(n,m);`
  `a[1][2]=0; ...`

Functions

- `cos`, `sin`, `tan`, `acos`, `asin`, `atan`, `cosh`, `sinh`, `acosh`, `asinh`, `log`, `log10`, `exp`, `sqrt`

Functions

- `cos, sin, tan, acos, asin, atan, cosh, sinh, acosh, asinh, log, log10, exp, sqrt`
- Function definition :

```
func type func_name(type & var)
{
instruction 1;
instruction n;
return outvar;
}
```

Functions

- `cos, sin, tan, acos, asin, atan, cosh, sinh, acosh, asinh, log, log10, exp, sqrt`
- Function definition :

```
func type func_name(type & var)
{
instruction 1;
instruction n ;
return outvar ;
}
```

- Simple function :

```
func outvar = expression of x and y and classical functions;
func f = exp(x)*y+sqrt(x)*cos(pi*x);
```

Control instruction

- loop for

```
for (int,cond,incr)
{
...
};
```

Control instruction

- loop for

```
for (int,cond,incr)
{
...
};
```

- loop while

```
while (cond)
{
...
};
```

Control instruction

- loop `for`

```
for (int,cond,incr)
{
...
};
```

- loop `while`

```
while (cond)
{
...
};
```

- control `if`

```
if (cond)
{
...
}
else {
...
}
;
```

Input/Output

- Open to read a file `ifstream`
  ```
  ifstream name(file_name);
  ```

Input/Output

- Open to read a file `ifstream`
  `ifstream name(file_name);`
- Open to write a file `ofstream`
  `ofstream name(file_name);`

Input/Output

- Open to read a file `ifstream`
  `ifstream name(file_name);`
- Open to write a file `ofstream`
  `ofstream name(file_name);`
- Read/Write in a file $\ll$ $\gg$

Input/Output

- Open to read a file `ifstream`
  `ifstream name(file_name);`
- Open to write a file `ofstream`
  `ofstream name(file_name);`
- Read/Write in a file $\ll \quad \gg$
- ```
  ofstream data("result.dat");
  for (int i=0,i<=100,i++)
  {
  data << " x = " << x[i] << endl
  };
  ```
  ```

How to define the mesh

Let Ω be an arbitrary domain :

- If the domain Ω is regular, for instance rectangle of size $[a, b] \times [c, d]$, then the command
  `mesh mesh_name = square(n,m,[a+(b-a)*x,c+(d-c)*y];`
  generates a regular mesh with triangles of size $n \times m$
  n means that the segment $[a, b]$ is divided into $n + 1$ points.

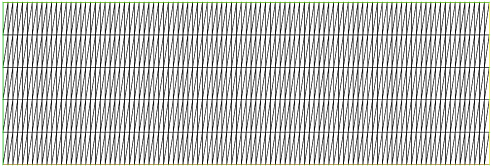
How to define the mesh

Let Ω be an arbitrary domain :

- If the domain Ω is regular, for instance rectangle of size $[a, b] \times [c, d]$, then the command
  `mesh mesh_name = square(n,m,[a+(b-a)*x,c+(d-c)*y]` ;
  generates a regular mesh with triangles of size $n \times m$
  n means that the segment $[a, b]$ is divided into $n+1$ points.
- Otherwise, the domain Ω has to be defined by a parametrization of its boundary with the following command
  `border border_name(t=beg,end) {x=x(t) ; y = y(t); label = num_label};`
  then, the mesh of Ω is obtained with
  `mesh mesh_name =buildmesh(b1(n1)+b2(n2)+...+bk(nk));` .
  n_i means that the border b_i is divided into $n_i + 1$ points.

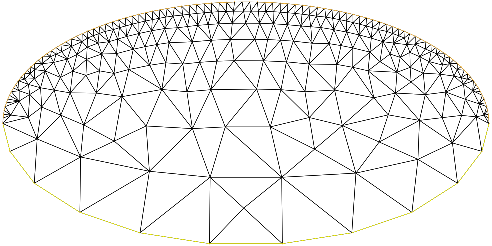
Examples

The code see the numerical code generates the mesh



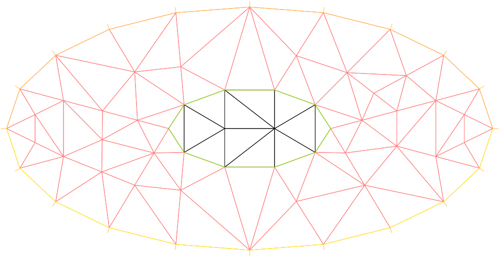
Examples

The code see the numerical code generates the mesh



EXAMPLES

The code see the numerical code generates the mesh



Others mesh functions

- `savemesh(mesh_name,file_name);` save the mesh and generates a file `file_name.msh`

Others mesh functions

- `savemesh(mesh_name,file_name);` save the mesh and generates a file `file_name.msh`
- `readmesh(file_name);` read the mesh file `file_name`

Others mesh functions

- savemesh(mesh_name,file_name); save the mesh and generates a file file_name.msh
- readmesh(file_name); read the mesh file file_name
- mesh newmesh = movemesh(oldmesh,[f1(x,y),f2(x,y)]);

Others mesh functions

- `savemesh(mesh_name,file_name);` save the mesh and generates a file `file_name.msh`
- `readmesh(file_name);` read the mesh file `file_name`
- `mesh newmesh = movemesh(oldmesh,[f1(x,y),f2(x,y)]);`
- `mesh newmesh = adaptmesh(oldmesh,crit);` adapt the mest w.r.t. one or more criterions `crit`

Solve a PDE with Freefem++

- Define the approximation space V_h with the command
  `fespace space_name(mesh_name,FE_type);`
  where `FE_type` is `P0, P1, ...`

Solve a PDE with Freefem++

- Define the approximation space V_h with the command
  `fespace space_name(mesh_name,FE_type);`
  where FE_type is `P0, P1, ...`
- Define the variational problem
  `problem problem_name(u,v,solver)=`
  `a(u,v)-l(v)`
  `+ (boundary conditions);` and add the command
  `problem_name;`
  or replace simply `problem` with `solve`.

Bilinear, linear form and boundary conditions

- Bilinear form

  `int2d(dx(u)*dx(v)+dy(u)*dy(v))` $\Longleftrightarrow \displaystyle\int_\Omega \nabla u(x,y) \cdot \nabla v(x,y)\, dx\, dy$

Bilinear, linear form and boundary conditions

- Bilinear form

  `int2d(dx(u)*dx(v)+dy(u)*dy(v))` $\Longleftrightarrow \displaystyle\int_\Omega \nabla u(x,y)\cdot\nabla v(x,y)\;dx\,dy$

- Linear form

  `int2d(f*v)` $\Longleftrightarrow \displaystyle\int_\Omega f(x,y)v(x,y)\;dx\,dy$

Bilinear, linear form and boundary conditions

- Bilinear form
  `int2d(dx(u)*dx(v)+dy(u)*dy(v))` $\Longleftrightarrow \displaystyle\int_\Omega \nabla u(x,y)\cdot\nabla v(x,y)\;dx\,dy$
- Linear form
  `int2d(f*v)` $\Longleftrightarrow \displaystyle\int_\Omega f(x,y)v(x,y)\;dx\,dy$
- Dirichlet boundary conditions
  `+on(border_name,u=g)`

Bilinear, linear form and boundary conditions

- Bilinear form

  `int2d(dx(u)*dx(v)+dy(u)*dy(v))` $\iff \int_\Omega \nabla u(x,y) \cdot \nabla v(x,y)\; dx\, dy$

- Linear form

  `int2d(f*v)` $\iff \int_\Omega f(x,y) v(x,y)\; dx\, dy$

- Dirichlet boundary conditions

  `+on(border_name,u=g)`

- Neumann boundary conditions

  `-int1d(mesh_name,border_name)(b*v)` $\iff \int_{\partial\Omega} \nabla u(x,y) \cdot n\; v\; dx\, dy$

  where $\nabla u(x,y) \cdot n = b$

OUTLINE

1. INTRODUCTION

2. EXAMPLES OF LINEAR AND NON LINEAR PDES
   - Poisson equation
   - Heat equation
   - Convection problem
   - Advection-diffusion problem
   - Incompressible Navier-Stokes equations
   - many among others

3. CONCLUSION

Outline

The problem

Let us consider the following Poisson problem on the unit square

$$\begin{cases} -\Delta\varphi = f & \text{in } \Omega \\ \varphi(x_1, x_2) = 0 & \text{on } \Gamma_2 \cup \Gamma_3 \\ \partial_n \varphi := \nabla\varphi \cdot n = 0 & \text{on } \Gamma_1 \cup \Gamma_4 \end{cases}$$

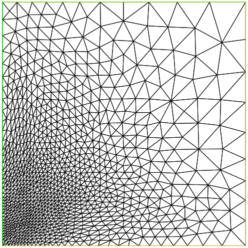


FIGURE: The mesh of Ω. The borders Γ_1 is by default $\{(x,y); y=0\}$, Γ_2 is by default $\{(x,y); x=1\}$, Γ_3 is by default $\{(x,y); y=1\}$, and Γ_4 is by default $\{(x,y); x=0\}$. The label for Γ_i is i.

The problem

Let us consider the following Poisson problem on the unit square

$$\begin{cases} -\Delta\varphi = f & \text{in } \Omega \\ \varphi(x_1, x_2) = 0 & \text{on } \Gamma_2 \cup \Gamma_3 \\ \partial_n \varphi := \nabla\varphi \cdot n = 0 & \text{on } \Gamma_1 \cup \Gamma_4 \end{cases}$$

Thus, the weak form is for any w test function :

$$\int_\Omega \nabla\varphi \cdot \nabla w \, dx = \int_\Omega f w \, dx + \int_{\Gamma_1 \cup \Gamma_4} 0 w \, ds$$

$$A(\varphi, w) = l(w)$$

with

$$A(\varphi, w) = \int_\Omega \nabla\varphi \cdot \nabla w \, dx$$

and

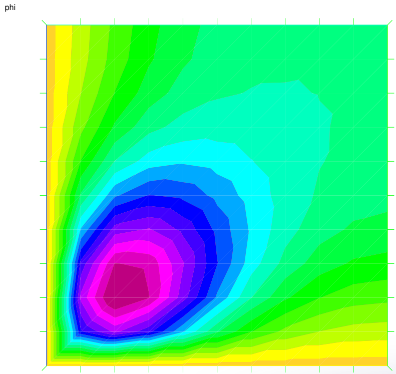
$$l(w) = \int_\Omega f w \, dx$$

The problem

Let us consider the following Poisson problem on the unit square

$$\begin{cases} -\Delta\varphi = f & \text{in } \Omega \\ \varphi(x_1, x_2) = 0 & \text{on } \Gamma_2 \cup \Gamma_3 \\ \partial_n\varphi := \nabla\varphi \cdot n = 0 & \text{on } \Gamma_1 \cup \Gamma_4 \end{cases}$$

Then,



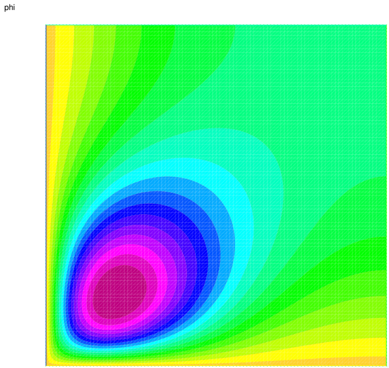
(a) 200 triangles

The problem

Let us consider the following Poisson problem on the unit square

$$\begin{cases} -\Delta\varphi = f & \text{in } \Omega \\ \varphi(x_1, x_2) = 0 & \text{on } \Gamma_2 \cup \Gamma_3 \\ \partial_n \varphi := \nabla\varphi \cdot n = 0 & \text{on } \Gamma_1 \cup \Gamma_4 \end{cases}$$

Then,



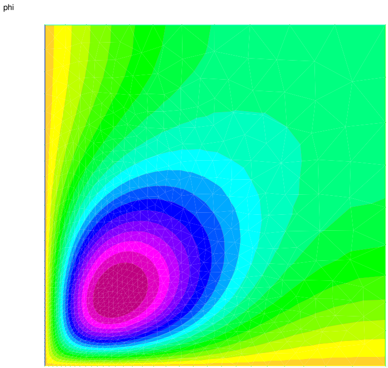
(b) 20000 triangles

The problem

Let us consider the following Poisson problem on the unit square

$$\begin{cases} -\Delta\varphi = f & \text{in } \Omega \\ \varphi(x_1, x_2) = 0 & \text{on } \Gamma_2 \cup \Gamma_3 \\ \partial_n \varphi := \nabla\varphi \cdot n = 0 & \text{on } \Gamma_1 \cup \Gamma_4 \end{cases}$$

Then,



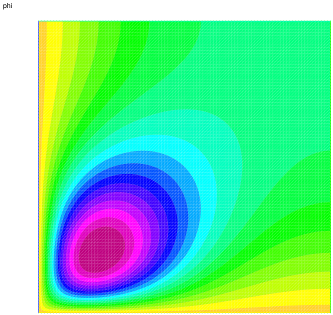
(c) Adaptive mesh (based on $\nabla\phi$) with 1798 triangles (initial mesh 200 triangles)

The problem

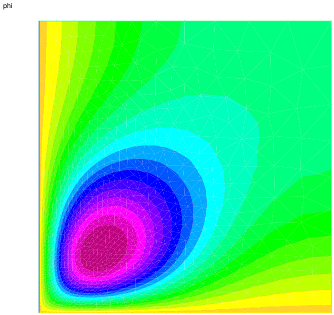
Let us consider the following Poisson problem on the unit square

$$\begin{cases} -\Delta\varphi = f & \text{in } \Omega \\ \varphi(x_1, x_2) = 0 & \text{on } \Gamma_2 \cup \Gamma_3 \\ \partial_n\varphi := \nabla\varphi \cdot n = 0 & \text{on } \Gamma_1 \cup \Gamma_4 \end{cases}$$

Then,



(d) 20000 triangles



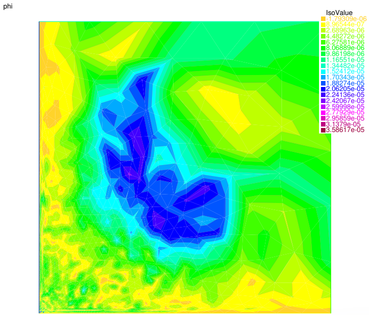
(e) Adaptive mesh (based on $\nabla\phi$) with 1798 triangles (initial mesh 200 triangles)

The problem

Let us consider the following Poisson problem on the unit square

$$\begin{cases} -\Delta\varphi = f & \text{in } \Omega \\ \varphi(x_1, x_2) = 0 & \text{on } \Gamma_2 \cup \Gamma_3 \\ \partial_n \varphi := \nabla\varphi \cdot n = 0 & \text{on } \Gamma_1 \cup \Gamma_4 \end{cases}$$

Then,



(f) local error : $|\varphi_{20000} - \varphi_{\text{adapt}}|$

The problem

Let us consider the Poisson problem on a ……. . .Poisson domain with homogenous Dirichlet boundary conditions with $f = 1$.

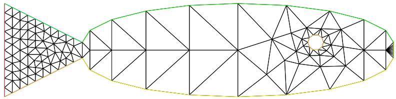
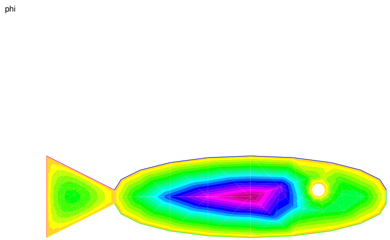


FIGURE: The mesh of Ω with 217 triangles

The problem

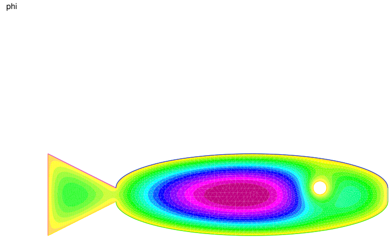
Let us consider the Poisson problem on a ....Poisson domain with homogenous Dirichlet boundary conditions with $f = 1$. Then,



(a) 217 triangles

THE PROBLEM

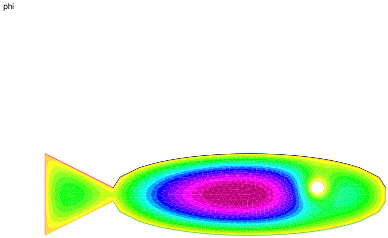
Let us consider the Poisson problem on a ....Poisson domain with homogenous Dirichlet boundary conditions with $f = 1$. Then,



(b) 21125 triangles

THE PROBLEM

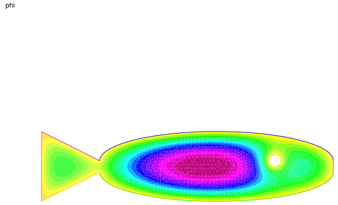
Let us consider the Poisson problem on a ....Poisson domain with homogenous Dirichlet boundary conditions with $f = 1$. Then,



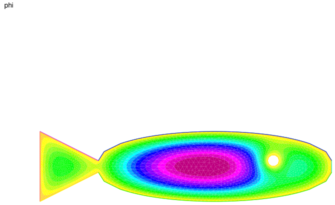
(c) Adaptive mesh (based on $\nabla\phi$) with 5155 triangles (initial mesh 217 triangles)

The problem

Let us consider the Poisson problem on a ....Poisson domain with homogenous Dirichlet boundary conditions with $f = 1$. Then,

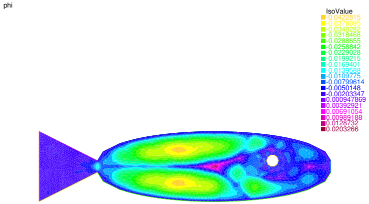


(d) 21125 triangles



(e) Adaptive mesh (based on $\nabla\phi$) with 5155 triangles (initial mesh 217 triangles)

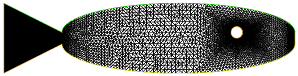
Let us consider the Poisson problem on a ....Poisson domain with homogenous Dirichlet boundary conditions with $f = 1$. Then,



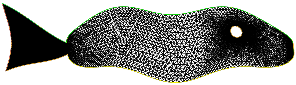
(f) local error

The problem

Let us consider the Poisson problem on a ....Poisson domain with homogenous Dirichlet boundary conditions with $f = 1$. One can also apply a deformation to the domain with `movemesh`, here $(x + sin(y\pi)/10, y + cos(x\pi)/10)$:



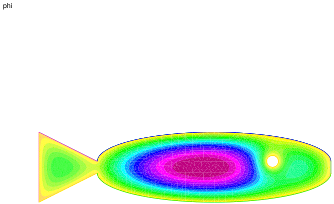
(g) Initial mesh with 21125 triangles



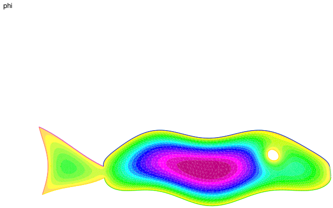
(h) Deformed mesh with 21125 triangles

THE PROBLEM

Let us consider the Poisson problem on a ....Poisson domain with homogenous Dirichlet boundary conditions with $f = 1$. One can also apply a deformation to the domain with movemesh, here $(x + sin(y\pi)/10, y + cos(x\pi)/10)$:



(i) Initial mesh with 21125 triangles



(j) Deformed mesh with 21125 triangles

Display results

- To display the result, we use the command `plot`

Display results

- To display the result, we use the command `plot`
  - for instance, `plot(phi);` display the two dimensional result.

Display results

- To display the result, we use the command `plot`
  - for instance, `plot(phi);` display the two dimensional result.
  - for instance, `plot(Omega,phi);` display the two dimensional result and the mesh `Omega`.

Display results

- To display the result, we use the command `plot`
  - for instance, `plot(phi);` display the two dimensional result.
  - for instance, `plot(Omega,phi);` display the two dimensional result and the mesh `Omega`.
  - for instance, `plot(Omega,phi,ps="name_of_figure.eps");` display the two dimensional result, the mesh `Omega` and save the plot in a file `name_of_figure.eps`.

Display results

- To display the result, we use the command `plot`
  - for instance, `plot(phi);` display the two dimensional result.
  - for instance, `plot(Omega,phi);` display the two dimensional result and the mesh `Omega`.
  - for instance, `plot(Omega,phi,ps="name_of_figure.eps");` display the two dimensional result, the mesh `Omega` and save the plot in a file `name_of_figure.eps`.
  - for instance, `plot(phi,dim=3);` display the isolines of $\phi(x, y)$

Display results

- To display the result, we use the command `plot`
  - for instance, `plot(phi);` display the two dimensional result.
  - for instance, `plot(Omega,phi);` display the two dimensional result and the mesh `Omega`.
  - for instance, `plot(Omega,phi,ps="name_of_figure.eps");` display the two dimensional result, the mesh `Omega` and save the plot in a file `name_of_figure.eps`.
  - for instance, `plot(phi,dim=3);` display the isolines of $\phi(x,y)$
  - for instance, `plot(phi,fill=1,dim=3);` display and fill between isolines

Display results

- To display the result, we use the command `plot`
  - for instance, `plot(phi);` display the two dimensional result.
  - for instance, `plot(Omega,phi);` display the two dimensional result and the mesh `Omega`.
  - for instance, `plot(Omega,phi,ps="name_of_figure.eps");` display the two dimensional result, the mesh `Omega` and save the plot in a file `name_of_figure.eps`.
  - for instance, `plot(phi,dim=3);` display the isolines of $\phi(x, y)$
  - for instance, `plot(phi,fill=1,dim=3);` display and fill between isolines
  - for instance, `plot(phi,value=1);` display an array of values for ϕ.

Display results

- To display the result, we use the command `plot`
  - for instance, `plot(phi);` display the two dimensional result.
  - for instance, `plot(Omega,phi);` display the two dimensional result and the mesh `Omega`.
  - for instance, `plot(Omega,phi,ps="name_of_figure.eps");` display the two dimensional result, the mesh `Omega` and save the plot in a file `name_of_figure.eps`.
  - for instance, `plot(phi,dim=3);` display the isolines of $\phi(x, y)$
  - for instance, `plot(phi,fill=1,dim=3);` display and fill between isolines
  - for instance, `plot(phi,value=1);` display an array of values for ϕ.
  - the command `plot` has several options ...

Display results

- To display the result, we use the command `plot`
  - for instance, `plot(phi);` display the two dimensional result.
  - for instance, `plot(Omega,phi);` display the two dimensional result and the mesh `Omega`.
  - for instance, `plot(Omega,phi,ps="name_of_figure.eps");` display the two dimensional result, the mesh `Omega` and save the plot in a file `name_of_figure.eps`.
  - for instance, `plot(phi,dim=3);` display the isolines of $\phi(x,y)$
  - for instance, `plot(phi,fill=1,dim=3);` display and fill between isolines
  - for instance, `plot(phi,value=1);` display an array of values for ϕ.
  - the command `plot` has several options ...
- It is also possible to export data to use paraview, tecplot, visit or other visualisation software

OUTLINE

The problem

Given f and $\boldsymbol{A}$ p.s.d, let us consider the following heat equation :

$$\begin{cases} \partial_t \varphi - \mathsf{div}(\boldsymbol{A}(t,x)\nabla\varphi) = f & \text{in } t \in (0,T],\ x \in \Omega \\ \varphi(t,x) = z(t,x) & \text{for } t > 0,\ x \in \Gamma_1 \\ \partial_n \varphi := \boldsymbol{A}\nabla\varphi \cdot n = 0 & \text{for } t > 0,\ x \in \Gamma_2 \\ \varphi(0,x) = \varphi_0(x) & \text{for } x \in \Omega \end{cases}$$

The problem

Given f and $\boldsymbol{A}$ p.s.d, let us consider the following heat equation :

$$\left\{\begin{array}{ll} \partial_t \varphi - \mathsf{div}(\boldsymbol{A}(t,x)\nabla\varphi) = f & \text{in } t \in (0,T],\ x \in \Omega \\ \varphi(t,x) = z(t,x) & \text{for } t > 0,\ x \in \Gamma_1 \\ \partial_n \varphi := \boldsymbol{A}\nabla\varphi \cdot n = 0 & \text{for } t > 0,\ x \in \Gamma_2 \\ \varphi(0,x) = \varphi_0(x) & \text{for } x \in \Omega \end{array}\right.$$

Thus, the weak form of the equation for any "suitable" w test function is :

$$\int_\Omega \partial_t \varphi(t,x) w + \boldsymbol{A}\nabla\varphi(t,x) \cdot \nabla w \, dx = \int_\Omega f(t,x) w \, dx + \text{ BC}$$

The problem

Given f and $\boldsymbol{A}$ p.s.d, let us consider the following heat equation :

$$\left\{\begin{array}{ll} \partial_t\varphi - \mathsf{div}(\boldsymbol{A}(t,x)\nabla\varphi) = f & \text{in } t \in (0,T],\ x \in \Omega \\ \varphi(t,x) = z(t,x) & \text{for } t > 0,\ x \in \Gamma_1 \\ \partial_n\varphi := \boldsymbol{A}\nabla\varphi\cdot n = 0 & \text{for } t > 0,\ x \in \Gamma_2 \\ \varphi(0,x) = \varphi_0(x) & \text{for } x \in \Omega \end{array}\right.$$

Thus, the weak form of the equation for any "suitable" w test function is :

$$\int_\Omega \partial_t\varphi(t,x)w + \boldsymbol{A}\nabla\varphi(t,x)\cdot\nabla w\,dx = \int_\Omega f(t,x)w\,dx + \text{ BC}$$

Noting $\delta t = T/N$ for some $N \in \mathbb{N}_+$, the evolution problem can be therefore approximated by :

$$\int_\Omega \frac{\varphi^{n+1} - \varphi^n}{\delta t}w + \boldsymbol{A}\nabla\varphi^{n+1}\cdot\nabla w\,dx = \int_\Omega f^n w\,dx + \text{ BC}$$

The problem

Given f and $\boldsymbol{A}$ p.s.d, let us consider the following heat equation :

$$\begin{cases} \partial_t \varphi - \mathrm{div}(\boldsymbol{A}(t,x)\nabla\varphi) = f & \text{in } t \in (0,T],\ x \in \Omega \\ \varphi(t,x) = z(t,x) & \text{for } t > 0,\ x \in \Gamma_1 \\ \partial_n \varphi := \boldsymbol{A}\nabla\varphi \cdot n = 0 & \text{for } t > 0,\ x \in \Gamma_2 \\ \varphi(0,x) = \varphi_0(x) & \text{for } x \in \Omega \end{cases}$$

Thus, the weak form of the equation for any "suitable" w test function is :

$$\int_\Omega \partial_t \varphi(t,x) w + \boldsymbol{A}\nabla\varphi(t,x) \cdot \nabla w \, dx = \int_\Omega f(t,x) w \, dx + \text{ BC}$$

Noting $\delta t = T/N$ for some $N \in \mathbb{N}_+$, the evolution problem can be therefore approximated by :

$$\int_\Omega \frac{\varphi^{n+1}}{\delta t} w + \boldsymbol{A}^{n+1}\nabla\varphi^{n+1} \cdot \nabla w \, dx = \int_\Omega \left(\frac{\varphi^n}{\delta t} + f^n \right) w \, dx + \text{ BC}$$

where φ^n is supposed to be an approximation of φ at time $t_n = n\delta t$.

The problem

Given f and $\boldsymbol{A}$ p.s.d, let us consider the following heat equation :

$$\begin{cases} \partial_t \varphi - \mathsf{div}(\boldsymbol{A}(t,x)\nabla\varphi) = f & \text{in } t \in (0,T],\ x \in \Omega \\ \varphi(t,x) = z(t,x) & \text{for } t > 0,\ x \in \Gamma_1 \\ \partial_n \varphi := \boldsymbol{A}\nabla\varphi \cdot n = 0 & \text{for } t > 0,\ x \in \Gamma_2 \\ \varphi(0,x) = \varphi_0(x) & \text{for } x \in \Omega \end{cases}$$

Thus, the weak form of the equation for any "suitable" w test function is :

$$\int_\Omega \partial_t \varphi(t,x) w + \boldsymbol{A}\nabla\varphi(t,x) \cdot \nabla w \, dx = \int_\Omega f(t,x) w \, dx + \text{ BC}$$

As a consequence, noting

$$a(t_{n+1}, \varphi, w) = \int_\Omega \frac{\varphi^{n+1}}{\delta t} w + \boldsymbol{A}^{n+1} \nabla \varphi^{n+1} \cdot \nabla w \, dx$$

and

$$l(t_n, w) = \int_\Omega \left(\frac{\varphi^n}{\delta t} + f^n \right) w \, dx + \text{ BC}$$

one has to solve

$$a(t_{n+1}, \varphi, w) = l(t_n, w),\ 0 \leqslant n < N-1.$$

Thus, the Freefem++ code is

```
//Parameters
int N = ...;
real T = ...,dt = ...;
//Define Omega
mesh Th = ...;
//Define FE space and all required functions (especially phi0)
fespace Vh ...;
```

Evolution problems with Freefem++

Thus, the Freefem++ code is

```
//Parameters
int N = ...;
real T = ...,dt = ...;
//Define Omega
mesh Th = ...;
//Define FE space and all required functions (especially phi0)
fespace Vh ...;
//Time loop
for(real t=0;t<=T;t=t+dt)
{
  solve Evolution_Problem(phi,w) =
    ...
  ;
  phi0 = phi;
  plot(...);
}
```

if the stiffness matrix depend on t otherwise

Thus, the Freefem++ code is

```
//Parameters
int N = ...;
real T = ...,dt = ...;
//Define Omega
mesh Th = ...;
//Define FE space and all required functions (especially phi0)
fespace Vh ...;
//Define the problem
problem Evolution_Problem(phi,w) =
//Time loop
for(real t=0;t<=T;t=t+dt)
{
  Evolution_Problem;
  phi0 = phi;
  plot(...);
}
```

An exemple

Let us consider the heat equation with $\boldsymbol{A}$ the identity matrix with homogenous Dirichlet boundary conditions and $f = \exp\left(-\sin(t)(x^2+y^2)\right)$ on the Poisson domain.

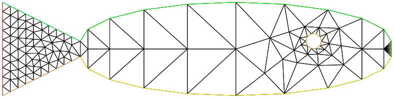
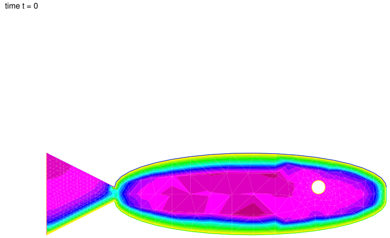


Figure: The mesh of Ω with 217 triangles

An exemple

Let us consider the heat equation with $\boldsymbol{A}$ the identity matrix with homogenous Dirichlet boundary conditions and $f = \exp\left(-\sin(t)(x^2+y^2)\right)$ on the Poisson domain.

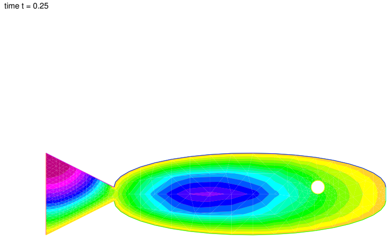
Then,



(a) t=0.00

An exemple

Let us consider the heat equation with $\boldsymbol{A}$ the identity matrix with homogenous Dirichlet boundary conditions and $f = \exp\left(-\sin(t)(x^2+y^2)\right)$ on the Poisson domain.
Then,

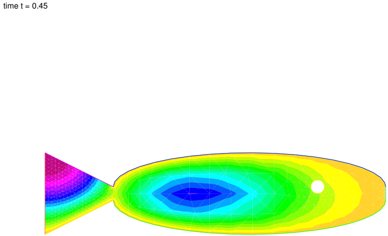


(b) t=0.25

An exemple

Let us consider the heat equation with $\boldsymbol{A}$ the identity matrix with homogenous Dirichlet boundary conditions and $f = \exp\left(-\sin(t)(x^2+y^2)\right)$ on the Poisson domain.

Then,

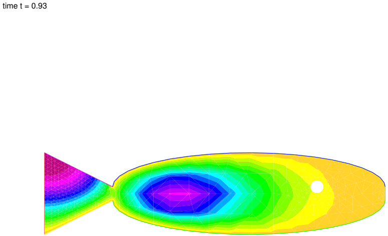


(c) t=0.45

An exemple

Let us consider the heat equation with $\boldsymbol{A}$ the identity matrix with homogenous Dirichlet boundary conditions and $f = \exp\left(-\sin(t)(x^2+y^2)\right)$ on the Poisson domain.

Then,

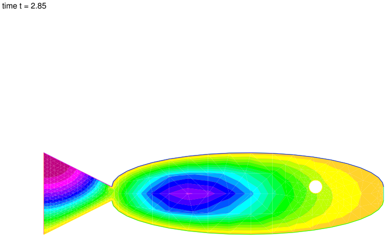


(d) t=0.93

An exemple

Let us consider the heat equation with $\boldsymbol{A}$ the identity matrix with homogenous Dirichlet boundary conditions and $f = \exp\left(-\sin(t)(x^2+y^2)\right)$ on the Poisson domain.

Then,



(e) t=2.85

How to visualize ?

As mentionned before one can save the figures. To make a video, one can save at each time step the figure through the command

```
plot(func,cmm="t= "+(t),ps="Folder_Name/File_Name"+num+".eps");
```

How to visualize ?

As mentionned before one can save the figures. To make a video, one can save at each time step the figure through the command

```
plot(func,cmm="t= "+(t),ps="Folder_Name/File_Name"+num+".eps");
```

into a file `File_Name"+num+".eps` contained in the folder `Folder_Name`.

How to visualize ?

As mentionned before one can save the figures. To make a video, one can save at each time step the figure through the command

```
plot(func,cmm="t= "+(t),ps="Folder_Name/File_Name"+num+".eps");
```

into a file `File_Name"+num+".eps` contained in the folder `Folder_Name`.
Then on unix operating system, one can convert all eps file into png file, for instance using `ImageMagick` and then concatenate to construct an animation using for instance `mencoder`.

How to visualize ?

As mentionned before one can save the figures. To make a video, one can save at each time step the figure through the command

```
plot(func,cmm="t= "+(t),ps="Folder_Name/File_Name"+num+".eps");
```

into a file `File_Name"+num+".eps` contained in the folder `Folder_Name`.
Then on unix operating system, one can convert all eps file into png file, for instance using `ImageMagick` and then concatenate to construct an animation using for instance `mencoder`.
Here, an example of bash script to do that

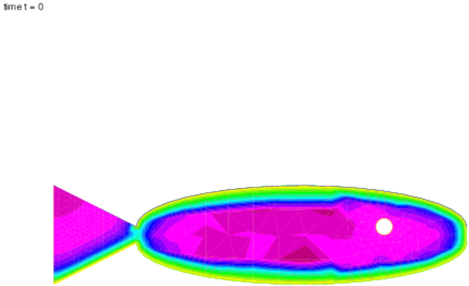
```
#!/bin/bash
#Convert eps file to png file
for file in *.eps; do
   convert ./"$file" ./"$file%.eps.png"
done
#Create a movie
mencoder mf:// -mf fps=25:type=png -ovc lavc -oac copy -o
../movie.avi
```

How to visualize ?

As mentionned before one can save the figures. To make a video, one can save at each time step the figure through the command

```
plot(func,cmm="t= "+(t),ps="Folder_Name/File_Name"+num+".eps");
```

into a file `File_Name"+num+".eps` contained in the folder `Folder_Name`.
Then on unix operating system, one can convert all eps file into png file, for instance using `ImageMagick` and then concatenate to construct an animation using for instance `mencoder`.



OUTLINE

1. INTRODUCTION

2. EXAMPLES OF LINEAR AND NON LINEAR PDES
   - Poisson equation
   - Heat equation
   - Convection problem
   - Advection-diffusion problem
   - Incompressible Navier-Stokes equations
   - many among others

3. CONCLUSION

A "characteristic Galerkin" method

Let us consider the following convection problem

$$\partial_t u + \boldsymbol{c}(x) \cdot \nabla u = 0, \ (t, x) \in (0, T) \times \Omega$$

with the initial data $u(0, x) = u_0(x), x \in \Omega$ and c assumed to be a regular function.

A "characteristic Galerkin" method

Let us consider the following convection problem

$$\partial_t u + \boldsymbol{c}(x) \cdot \nabla u = 0,\ (t, x) \in (0, T) \times \Omega$$

with the initial data $u(0, x) = u_0(x), x \in \Omega$ and c assumed to be a regular function.
Then, the exact solution is $u(t, x) = u_0(X(0; t, x))$ where X solve the ODE

$$X'(s; t, x) = c(X(s)),\ X(t; t, x) = x\ .$$

A "characteristic Galerkin" method

Let us consider the following convection problem

$$\partial_t u + \boldsymbol{c}(x) \cdot \nabla u = 0,\ (t, x) \in (0, T) \times \Omega$$

with the initial data $u(0, x) = u_0(x), x \in \Omega$ and c assumed to be a regular function.
Then, the exact solution is $u(t, x) = u_0(X(0; t, x))$ where X solve the ODE

$$X'(s; t, x) = c(X(s)),\ X(t; t, x) = x\ .$$

Therefore, one can compute the solution at point (t, x) with the initial guess $X(-t, t; x)$.

A "characteristic Galerkin" method

Let us consider the following convection problem

$$\partial_t u + \boldsymbol{c}(x) \cdot \nabla u = 0, \ (t, x) \in (0, T) \times \Omega$$

with the initial data $u(0, x) = u_0(x), x \in \Omega$ and c assumed to be a regular function.

Then, the exact solution is $u(t, x) = u_0(X(0; t, x))$ where X solve the ODE

$$X'(s; t, x) = c(X(s)), \ X(t; t, x) = x \ .$$

Therefore, one can compute the solution at point (t, x) with the initial guess $X(-t, t; x)$.

We perform this at each t_n. Noting $c = (c_1, c_2)$, the command is simply

```
u = convect([c1,c2],-dt,uold)
```

where convect returns $u \circ X$(t) see the numerical code.

Outline

Applications 1 : advection-diffusion problem

Let f and $c(t,x) \in \mathbb{R}^2$ for all $(t,x) \in [0,T] \times \Omega$ be given functions. Let us consider the following advection-diffusion problem

$$\partial_t u + c \cdot \nabla u - \Delta u = f, x \in \Omega, t > 0$$

with $u(0,x) = u_0(x)$.

Applications 1 : advection-diffusion problem

Let f and $c(t,x) \in \mathbb{R}^2$ for all $(t,x) \in [0,T] \times \Omega$ be given functions. Let us consider the following advection-diffusion problem

$$\partial_t u + c \cdot \nabla u - \Delta u = f, x \in \Omega, t > 0$$

with $u(0,x) = u_0(x)$. It can be written as

$$D_t u - \Delta u = f, x \in \Omega, t > 0$$

with $u(0,x) = u_0(x)$ where $D_t u$ is the convection derivative.

Applications 1 : advection-diffusion problem

Let f and $c(t, x) \in \mathbb{R}^2$ for all $(t, x) \in [0, T] \times \Omega$ be given functions. Let us consider the following advection-diffusion problem

$$\partial_t u + c \cdot \nabla u - \Delta u = f, x \in \Omega, t > 0$$

with $u(0, x) = u_0(x)$. It can be written as

$$D_t u - \Delta u = f, x \in \Omega, t > 0$$

with $u(0, x) = u_0(x)$ where $D_t u$ is the convection derivative.
Therefore, one can, for instance, use an implicit Euler scheme in time with the characteristic method :

$$\int_\Omega \frac{u^{n+1} - u \circ X^n}{\delta t} w + \nabla u^{n+1} \cdot \nabla w \; dx = \int_\Omega f w \; dx$$

with $\dfrac{d}{dt} X = c$.

Applications 1 : advection-diffusion problem

Let f and $c(t,x) \in \mathbb{R}^2$ for all $(t,x) \in [0,T] \times \Omega$ be given functions. Let us consider the following advection-diffusion problem

$$\partial_t u + c \cdot \nabla u - \Delta u = f, x \in \Omega, t > 0$$

with $u(0,x) = u_0(x)$. It can be written as

$$D_t u - \Delta u = f, x \in \Omega, t > 0$$

with $u(0,x) = u_0(x)$ where $D_t u$ is the convection derivative.
Therefore, one can, for instance, use an implicit Euler scheme in time with the characteristic method :

$$\int_\Omega \frac{u^{n+1} - u \circ X^n}{\delta t} w + \nabla u^{n+1} \cdot \nabla w \, dx = \int_\Omega f w \, dx$$

with $\dfrac{d}{dt} X = c$. see the numerical code

Exercice : system of advection-diffusion problem

Let $f(t,x) \in \mathbb{R}^2$ for all $(t,x) \in [0,T] \times \Omega$ be a given function. Let us consider the following coupled advection-diffusion problem of a species i

$$\partial_t u_i + u \cdot \nabla u_i - \Delta u_i = f_i, x \in \Omega, t > 0$$

with $u(0,x) = u_0(x)$

Exercice : system of advection-diffusion problem

Let $f(t,x) \in \mathbb{R}^2$ for all $(t,x) \in [0,T] \times \Omega$ be a given function. Let us consider the following coupled advection-diffusion problem of a species i

$$\partial_t u_i + u \cdot \nabla u_i - \Delta u_i = f_i, x \in \Omega, t > 0$$

with $u(0,x) = u_0(x)$ It can be written as

$$D_t u_i - \Delta u_i = f_i, x \in \Omega, t > 0$$

with $u(0,x) = u_0(x)$ where $D_t u$ is the convection derivative.

Exercice : system of advection-diffusion problem

Let $f(t,x) \in \mathbb{R}^2$ for all $(t,x) \in [0,T] \times \Omega$ be a given function. Let us consider the following coupled advection-diffusion problem of a species i

$$\partial_t u_i + u \cdot \nabla u_i - \Delta u_i = f_i, x \in \Omega, t > 0$$

with $u(0,x) = u_0(x)$ It can be written as

$$D_t u_i - \Delta u_i = f_i, x \in \Omega, t > 0$$

with $u(0,x) = u_0(x)$ where $D_t u$ is the convection derivative.
Therefore, as done before, using an implicit Euler scheme in time with the characteristic method, we get

$$\int_\Omega \frac{u_1^{n+1} - u_1 \circ X^n}{\delta t} w_1 + \nabla u_1^{n+1} \cdot \nabla w_1 \; dx - \int_\Omega f w_1 \; dx +$$

$$\int_\Omega \frac{u_2^{n+1} - u_2 \circ X^n}{\delta t} w_2 + \nabla u_2^{n+1} \cdot \nabla w_2 \; dx - \int_\Omega f w_2 \; dx = 0$$

with $\dfrac{d}{dt} X = u$.

Exercice : system of advection-diffusion problem

Let $f(t,x) \in \mathbb{R}^2$ for all $(t,x) \in [0,T] \times \Omega$ be a given function. Let us consider the following coupled advection-diffusion problem of a species i

$$\partial_t u_i + u \cdot \nabla u_i - \Delta u_i = f_i, x \in \Omega, t > 0$$

with $u(0,x) = u_0(x)$ It can be written as

$$D_t u_i - \Delta u_i = f_i, x \in \Omega, t > 0$$

with $u(0,x) = u_0(x)$ where $D_t u$ is the convection derivative.
Therefore, as done before, using an implicit Euler scheme in time with the characteristic method, we get

$$\int_\Omega \frac{u_1^{n+1} - u_1 \circ X^n}{\delta t} w_1 + \nabla u_1^{n+1} \cdot \nabla w_1 \; dx - \int_\Omega f w_1 \; dx +$$

$$\int_\Omega \frac{u_2^{n+1} - u_2 \circ X^n}{\delta t} w_2 + \nabla u_2^{n+1} \cdot \nabla w_2 \; dx - \int_\Omega f w_2 \; dx = 0$$

with $\frac{d}{dt} X = u$. see the numerical code

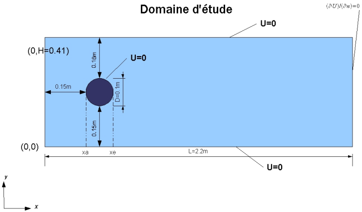
Outline

THE PROBLEM

Let us consider the incompressible Navier-Stokes equation

$$(NSI)\begin{cases} \rho(\partial_t u + (u.\nabla)u) - \rho\nu\Delta u + \nabla p &= 0 \\ \mathrm{div}(u) &= 0 \\ u(x,0) &= u_0(x) \\ + \text{ boundary conditions} & \end{cases}$$

on the domain Ω



with the fluid velocity, $\rho = 1.0$ the density, the viscosity $\nu = 10^{-3}\ m^2/s$ and p the pressure.

VF

Let V be the functional space for u and M the one for p. Let us note the discrete spaces as follows

$$V_h = \{v_h \in V;\, v_{h|K} \in \mathbb{P}_k,\, \forall K \in \tau_h\}$$

and

$$M_h = \{q_h \in M;\, q_{h|K} \in \mathbb{P}_l,\, \forall K \in \tau_h\}$$

where τ_h stands for the mesh and K a given finite element. We fix $k = 2$ and $l = 1$.
Noting $v \in V_h$ and $p \in M_h$ the test functions, one can perform the following implicit scheme

$$\left\{ \begin{array}{rcl} \rho \int_\Omega \frac{u^{n+1}}{\delta t} v\, dx + \rho\nu \int_\Omega \nabla u^{n+1} : \nabla v\, dx - \int_\Omega \mathrm{div}(v) p^{n+1}\, dx & = & \\ \rho \int_\Omega \frac{u^n \circ X^n(x)}{\delta t} v\, dx & & \\ \int_\Omega \mathrm{div}(u^{n+1}) q\, dx & = & 0 \end{array} \right.$$

where the characteristic method is used as in the previous example.

The code is

```
problem pbNSI2D2(u1,u2,p,v1,v2,q,solver=UMFPACK)
  = int2d(Th)( rho/dt*(u1*v1+u2*v2)
             + rho*nu*( dx(u1)*dx(v1)+ dy(u1)*dy(v1)+
                        dx(u2)*dx(v2)+ dy(u2)*dy(v2)
                       )
             -p*dx(v1)-p*dy(v2)
             -q*dx(u1)-q*dy(u2) + perturb*p*q
     )
  -int2d(Th) (rho/dt*convect([u1car,u2car],-dt,u1car)*v1+
              rho/dt*convect([u1car,u2car],-dt,u2car)*v2
      )
  -int1d(Th,3)( g1*v1 +g2*v2 ) // Condition de Neumann
  +on( 3, u2 = 0 )
  +on( 1 , u1 = u0 , u2 = v0 )
  +on( 2 , 4 , 5 , u1 = 0 , u2 = 0 )
  ;
```

see the full numerical code.

Outline

1. Introduction

2. Examples of linear and non linear PDEs
   - Poisson equation
   - Heat equation
   - Convection problem
   - Advection-diffusion problem
   - Incompressible Navier-Stokes equations
   - many among others

3. Conclusion

- The convection-diffusion problem see the numerical code
- The shallow water equations on fixed and moving bottom see the numerical code
- The shallow water and Exner equations see the numerical code
- see the section "Learning by examples" of the freefem++ pdf file.

1. Introduction

2. Examples of linear and non linear PDEs
   - Poisson equation
   - Heat equation
   - Convection problem
   - Advection-diffusion problem
   - Incompressible Navier-Stokes equations
   - many among others

3. Conclusion

And a

Finally a lot of equations can be quickly solved with freefem++.


Enjoy
yourself