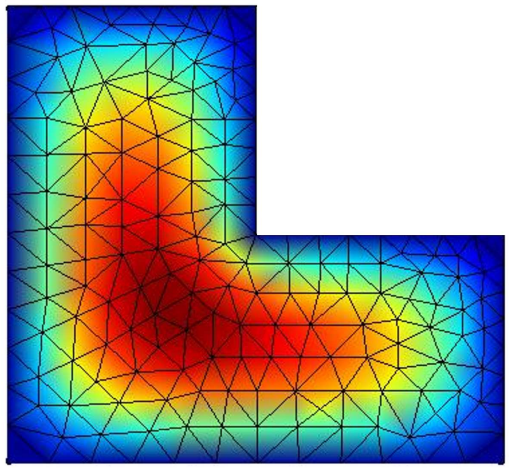
[302.044] Numerical Methods in Fluid Dynamics

Diffusion phenomena in weak formulation

F. Romanò & P.-E. des Boscs
Institute of Fluid Mechanics and Heat Transfer, TU Wien



December 7th 2017, Vienna, Austria

francesco.romano@tuwien.ac.at
pierre-emmanuel.boscs@tuwien.ac.at

Preliminary Information

FreeFEM++: Preliminary Information

Preliminary Information

FreeFEM++: Preliminary Information

- FreeFEM++ is written in C++

Preliminary Information

FreeFEM++: Preliminary Information

- FreeFEM++ is written in C++
- It is a fully integrated software

Preliminary Information

FreeFEM++: Preliminary Information

- FreeFEM++ is written in C++
- It is a fully integrated software
- Its syntax is very close to the mathematical weak form

Preliminary Information

FreeFEM++: Preliminary Information

- FreeFEM++ is written in C++
- It is a fully integrated software
- Its syntax is very close to the mathematical weak form
- Even complicated geometries are meshed

Preliminary Information

FreeFEM++: Preliminary Information

- FreeFEM++ is written in C++
- It is a fully integrated software
- Its syntax is very close to the mathematical weak form
- Even complicated geometries are meshed
- Interpolation algorithms for multiple grids

Preliminary Information

FreeFEM++: Preliminary Information

- FreeFEM++ is written in C++
- It is a fully integrated software
- Its syntax is very close to the mathematical weak form
- Even complicated geometries are meshed
- Interpolation algorithms for multiple grids
- Adapting meshing tools

Preliminary Information

FreeFEM++: Preliminary Information

- FreeFEM++ is written in C++
- It is a fully integrated software
- Its syntax is very close to the mathematical weak form
- Even complicated geometries are meshed
- Interpolation algorithms for multiple grids
- Adapting meshing tools
- Different functional basis ($\mathbb{P}^0$, $\mathbb{P}^1$, $\mathbb{P}^2$, $\mathbb{P}^{1d}$, etc...)

Preliminary Information

FreeFEM++: Preliminary Information

- FreeFEM++ is written in C++
- It is a fully integrated software
- Its syntax is very close to the mathematical weak form
- Even complicated geometries are meshed
- Interpolation algorithms for multiple grids
- Adapting meshing tools
- Different functional basis ($\mathbb{P}^0$, $\mathbb{P}^1$, $\mathbb{P}^2$, $\mathbb{P}^{1d}$, etc...)
- Integrated viewer to visualize the results

Preliminary Information

FreeFEM++: Preliminary Information

- FreeFEM++ is written in C++
- It is a fully integrated software
- Its syntax is very close to the mathematical weak form
- Even complicated geometries are meshed
- Interpolation algorithms for multiple grids
- Adapting meshing tools
- Different functional basis ($\mathbb{P}^0$, $\mathbb{P}^1$, $\mathbb{P}^2$, $\mathbb{P}^{1d}$, etc...)
- Integrated viewer to visualize the results
- Object-Oriented Programming (OOP) is easily achievable

Poisson Equation

Elliptic PDEs in FreeFEM++

Poisson Equation

Poisson Equation

Elliptic PDEs in FreeFEM++

Poisson Equation

- Strong form:

Poisson Equation

Elliptic PDEs in FreeFEM++

Poisson Equation

- Strong form:

$$\begin{cases} -\Delta u(\mathbf{x}) = f(\mathbf{x}) \quad , \quad \mathbf{x} \in \Omega \\ \\ +\text{Homogeneous Dirichlet's BCs} \end{cases}$$

Poisson Equation

Elliptic PDEs in FreeFEM++

Poisson Equation

- Strong form:

$$\begin{cases} -\Delta u(\mathbf{x}) = f(\mathbf{x}) \quad , \quad \mathbf{x} \in \Omega \\ \\ +\text{Homogeneous Dirichlet's BCs} \end{cases}$$

- Weak form:

Poisson Equation

Elliptic PDEs in FreeFEM++

Poisson Equation

- Strong form:

$$\begin{cases} -\Delta u(\mathbf{x}) = f(\mathbf{x}) \quad , \quad \mathbf{x} \in \Omega \\ \\ +\text{Homogeneous Dirichlet's BCs} \end{cases}$$

- Weak form:

$$\boxed{\begin{cases} \int_\Omega \nabla u \cdot \nabla v d\Omega = \int_\Omega f v d\Omega \quad , \quad \mathbf{x} \in \Omega \\ \\ u|_{\Gamma=\partial\Omega} = 0 \end{cases}}$$

Poisson Equation

Elliptic PDEs in FreeFEM++

2-D Poisson Equation

Poisson Equation

Elliptic PDEs in FreeFEM++

2-D Poisson Equation

- Weak form: FEM discretization

$$\begin{cases} \displaystyle\int_{\Omega_h} \frac{\partial u_h}{\partial x}\frac{\partial v_h}{\partial x} + \frac{\partial u_h}{\partial y}\frac{\partial v_h}{\partial y} d\Omega_h = \int_{\Omega_h} f v_h d\Omega_h \quad , \quad \mathbf{x} \in \Omega_h \\ \\ u|_{\Gamma_h = \partial\Omega_h} = 0 \end{cases}$$

Poisson Equation

Elliptic PDEs in FreeFEM++

2-D Poisson Equation

- Fist case: $\Omega = [-1, 1] \times [-1, 1]$
  - definition of the discrete domain via edges and relative orientation:

```
1  border B(t=0,1){x=-1+2*t; y=-1; label=G;}
2  border R(t=0,1){x=1; y=-1+2*t; label=G;}
3  border T(t=0,1){x=1-2*t; y=1; label=G;}
4  border L(t=0,1){x=-1; y=1-2*t; label=G;}
```

  - plot of the nodes distribution along the edges:

```
1  plot (B(50)+R(26)+T(31)+L(41), wait=true, ps="square
       .eps");
```

Poisson Equation

Elliptic PDEs in FreeFEM++

2-D Poisson Equation

- First case: $\Omega = [-1, 1] \times [-1, 1]$
  - triangular mesh built via Delaunay-Voronoi algorithm:

```
1  mesh Omega = buildmesh (B(50)+R(26)+T(31)+L(41));
```

  - plot of the mesh:

```
1  plot (Omega, wait = 1);
```

  - definition of the projection space for solution and test functions:

```
1  fespace Vh(Omega,P1); Vh u, v;
```

    both u_h and v_h belong to V_h

Poisson Equation

Elliptic PDEs in FreeFEM++

2-D Poisson Equation

- First case: $\Omega = [-1, 1] \times [-1, 1]$
  - triangular mesh built via Delaunay-Voronoi algorithm:

```
1  mesh Omega = buildmesh (B(50)+R(26)+T(31)+L(41));
```

  - plot of the mesh:

```
1  plot (Omega, wait = 1);
```

  - definition of the projection space for solution and test functions:

```
1  fespace Vh(Omega,P1); Vh u, v;
```

both u_h and v_h belong to V_h
V_h is chosen as the continuous piecewise-linear functional space $\mathbb{P}^1$

Poisson Equation

Elliptic PDEs in FreeFEM++

2-D Poisson Equation

- First case: $\Omega = [-1, 1] \times [-1, 1]$
  - set-up of the known term on the rhs:

```
1  func f = x*sin(y);
```

  - definition of the Poisson equation solver:

```
1  solve Poisson(u, v, solver = LU) =
2  + int2d(Omega)(dx(u)*dx(v) + dy(u)*dy(v))
3  - int2d(Omega)(f*v)
4  + on(G,u=0);
```

  - plot the solution:

```
1  plot(u);
```

Poisson Equation

Elliptic PDEs in FreeFEM++

2-D Poisson Equation

▷ First case: $\Omega = [-1, 1] \times [-1, 1]$

```
1  border B(t=0,1){x=-1+2*t; y=-1; label=G;}
2  border R(t=0,1){x= 1 ; y=-1+2*t; label=G;}
3  border T(t=0,1){x= 1-2*t; y=1; label=G;}
4  border L(t=0,1){x=-1; y=1-2*t; label=G;}
5  mesh Omega = buildmesh(B(50)+R(26)
6                         +T(31)+L(41));
7  fespace Vh(Omega,P1); Vh u, v;
8  func f = x*sin(y);
9  solve Poisson(u, v, solver = LU) =
10 + int2d(Omega)(dx(u)*dx(v) + dy(u)*dy(v))
11 - int2d(Omega)(f*v)
12 + on(G,u=0);
13 plot(u);
```

Poisson Equation

Object Oriented Programming in FreeFEM++

FreeFEM++: Object Oriented Programming

- First case: $\Omega = [-1, 1] \times [-1, 1]$

  Mesh Block

```
1  border B(t=0,1){x=-1+2*t; y=-1; label=G;}
2  border R(t=0,1){x= 1; y=-1+2*t; label=G;}
3  border T(t=0,1){x= 1-2*t; y=1; label=G;}
4  border L(t=0,1){x=-1; y=1-2*t; label=G;}
5  mesh Omega = buildmesh(B(50)+R(26)
6                         +T(31)+L(41));
```

Poisson Equation

Object Oriented Programming in FreeFEM++

FreeFEM++: Object Oriented Programming

- First case: $\Omega = [-1, 1] \times [-1, 1]$

  Functional Space Block

```
1  fespace Vh(Omega,P1); Vh u, v;
```

Poisson Equation

Object Oriented Programming in FreeFEM++

FreeFEM++: Object Oriented Programming

- First case: $\Omega = [-1, 1] \times [-1, 1]$

  Solver Block

```
1  func f = x*sin(y);
2  solve Poisson(u, v, solver = LU) =
3  + int2d(Omega)(dx(u)*dx(v) + dy(u)*dy(v))
4  - int2d(Omega)(f*v)
5  + on(G,u=0);
6  plot(u);
```

Poisson Equation

Object Oriented Programming in FreeFEM++

FreeFEM++: Object Oriented Programming

- Second case: $\Omega = \left\{ x^2 + y^2 \leq 1 \right\}$

```
1  border C(t=0,2*pi){x=cos(t); y=sin(t);
2                     label=G;}
3  mesh Omega = buildmesh (C(150));
4  fespace Vh(Omega,P1); Vh u, v;
5  func f = x*sin(y);
6  solve Poisson(u, v, solver = LU) =
7  + int2d(Omega)(dx(u)*dx(v) + dy(u)*dy(v))
8  - int2d(Omega)(f*v)
9  + on(G,u=0);
10 plot(u);
```

Poisson Equation

Object Oriented Programming in FreeFEM++

FreeFEM++: Object Oriented Programming

- Second case: $\Omega = \left\{ x^2 + y^2 \leq 1 \right\}$

  Mesh Block Changed

```
1  border C(t=0,2*pi){x=cos(t); y=sin(t);
2                     label=G;}
3  mesh Omega = buildmesh (C(150));
```

  Functional Space Block: Unchanged

  Solver Block: Unchanged

Poisson Equation

Elliptic PDEs in FreeFEM++

2-D Poisson Equation: Neumann BCs

$$\begin{cases} \displaystyle\int_{\Omega_h} \frac{\partial u_h}{\partial x}\frac{\partial v_h}{\partial x} + \frac{\partial u_h}{\partial y}\frac{\partial v_h}{\partial y} d\Omega_h = \int_{\Omega_h} f v_h d\Omega_h + \int_{\Gamma_h^N} \psi v_h d\gamma_h \\ \\ + u|_{\Gamma_h^D} = g \quad , \quad u_{,n}\,|_{\Gamma_h^N} = \psi \quad , \quad \text{where } \Gamma_h^D \cup \Gamma_h^N = \partial\Omega_h \end{cases}$$

- The Solver Block includes the boundary conditions:
  - definition of the Poisson equation solver:

```
1  solve Poisson(u, v, solver = LU) =
2  + int2d(Omega)(dx(u)*dx(v) + dy(u)*dy(v))
3  - int2d(Omega)(f*v)
4  + on(GD, u=g) - int1d(Omega,GN)(psi*v);
```

Poisson Equation

Elliptic PDEs in FreeFEM++

2-D Poisson Equation: Robin BCs

$$\begin{cases} \displaystyle\int_{\Omega_h} \frac{\partial u_h}{\partial x}\frac{\partial v_h}{\partial x} + \frac{\partial u_h}{\partial y}\frac{\partial v_h}{\partial y} d\Omega_h = \int_{\Omega_h} f v_h d\Omega_h + \int_{\Gamma_h} \psi v_h d\gamma_h \\ \\ + \left(u_{,n} + \alpha u\right)|_{\Gamma_h = \partial\Omega_h} = g \end{cases}$$

- The Solver Block includes the boundary conditions:
  - definition of the Poisson equation solver:

```
1  solve Poisson(u, v, solver = LU) =
2  + int2d(Omega)(dx(u)*dx(v) + dy(u)*dy(v))
3  - int2d(Omega)(f*v)
4  - int1d(Omega,G)(g*v - alpha*u*v);
```

Poisson Equation

Elliptic PDEs in FreeFEM++

2-D Poisson Equation: Periodic BCs

$$\begin{cases} \displaystyle\int_{\Omega_h} \frac{\partial u_h}{\partial x}\frac{\partial v_h}{\partial x} + \frac{\partial u_h}{\partial y}\frac{\partial v_h}{\partial y} d\Omega_h = \int_{\Omega_h} f v_h d\Omega_h \\ \\ + \text{periodic BCs} \end{cases}$$

- The Functional Space Block includes the BCs:
  - definition of the projection space for solution and test functions:

```
1  fespace Vh(Omega,P2,
2  periodic[<edgename>,<edgecoord>]);Vh u,v;
```

- The Solver Block does not include the boundary conditions.

Poisson Equation

Elliptic PDEs in FreeFEM++

2-D Poisson Equation: Error Convergence

$$\begin{cases} \displaystyle\int_{\Omega_h} \frac{\partial u_h}{\partial x}\frac{\partial v_h}{\partial x} + \frac{\partial u_h}{\partial y}\frac{\partial v_h}{\partial y} d\Omega_h = \int_{\Omega_h} f v_h d\Omega_h \\ \\ + u|_{\Gamma_h = \partial\Omega_h} = u_{exact} \end{cases}$$

- Exact solution:

$$\boxed{u_{exact} = \sin(x - 2y^2)}$$

- Consistent source term:

$$f = (1 + 16y^2)\sin(x - 2y^2) + 4\cos(x - 2y^2)$$

Poisson Equation

Elliptic PDEs in FreeFEM++

2-D Poisson Equation: Error Convergence

- Problem domain: $\Omega = [-1, 1] \times [-1, 1]$;
- Definition of f and u_{exact};
- Initialization on the error vector for the 5 tested grids:

```
1  int G=1;
2  border B(t=0,1){x=-1+2*t; y=-1; label=G;}
3  border R(t=0,1){x= 1 ; y=-1+2*t; label=G;}
4  border T(t=0,1){x= 1-2*t; y=1; label=G;}
5  border L(t=0,1){x=-1; y=1-2*t; label=G;}
6  func f = + (1+16*y^2)*sin(x-2*y^2)
7           + 4*cos(x-2*y^2);
8  func uex = sin(x-2*y^2);
9  int Ngrid = 5;
10 real[int] L2err(Ngrid);
```

Poisson Equation

Elliptic PDEs in FreeFEM++

2-D Poisson Equation: Error Convergence

▷ Convergence loop for the 5 meshes:

```
1  for(int n=0;n<Ngrid;n++)
2  {
3      mesh Omega=buildmesh(B(3*(n+1))+
4           R(3*(n+1))+T(3*(n+1))+L(3*(n+1)));
5      fespace Vh(Omega,P2); Vh u, v;
6      solve Poisson(u, v, solver = LU) =
7      + int2d(Omega)(dx(u)*dx(v)+dy(u)*dy(v))
8      - int2d(Omega)(f*v) + on(G,u=uex);
9      plot(Omega,u,wait=1,fill=1);
10     L2err[n]=sqrt(int2d(Omega)((u-uex)^2));
11 }
12 for(int n=0;n<Ngrid;n++)
13 cout << 10*(n+1) << " Nodes: ||err||_L2 = "<< L2error[n]
       <<endl;
```

Diffusion Equation

Parabolic PDEs in FreeFEM++

Diffusion Equation

Diffusion Equation

Parabolic PDEs in FreeFEM++

Diffusion Equation

- Strong form:

Diffusion Equation

Parabolic PDEs in FreeFEM++

Diffusion Equation

- Strong form:

$$\begin{cases} \partial_t u - \nabla \cdot (\sigma \nabla u) = f(\mathbf{x}, t) \quad , \quad \mathbf{x} \in \Omega \ , \ t > 0 \\ \\ +\text{Dirichlet's BCs} \end{cases}$$

Diffusion Equation

Parabolic PDEs in FreeFEM++

Diffusion Equation

- Strong form:

$$\begin{cases} \partial_t u - \nabla \cdot (\sigma \nabla u) = f(\mathbf{x}, t) \quad , \quad \mathbf{x} \in \Omega \ , \ t > 0 \\ \\ +\text{Dirichlet's BCs} \end{cases}$$

- Weak form:

Diffusion Equation

Parabolic PDEs in FreeFEM++

Diffusion Equation

- Strong form:

$$\begin{cases} \partial_t u - \nabla \cdot (\sigma \nabla u) = f(\mathbf{x}, t) \quad , \quad \mathbf{x} \in \Omega \ , \ t > 0 \\ \\ +\text{Dirichlet's BCs} \end{cases}$$

- Weak form:

$$\begin{cases} \int_\Omega v \partial_t u d\Omega + \int_\Omega \sigma \nabla u \cdot \nabla v d\Omega = \int_\Omega f v d\Omega \quad , \quad \mathbf{x} \in \Omega \ , \ t > 0 \\ \\ u|_{\Gamma = \partial\Omega} = g \end{cases}$$

Diffusion Equation

Parabolic PDEs in FreeFEM++

2-D Diffusion Equation

Diffusion Equation

Parabolic PDEs in FreeFEM++

2-D Diffusion Equation

- Weak form: FEM discretization

$$\begin{cases} \displaystyle\int_{\Omega_h} v_h \partial_t u_h d\Omega_h + \int_{\Omega_h} \sigma \left(\frac{\partial u_h}{\partial x}\frac{\partial v_h}{\partial x} + \frac{\partial u_h}{\partial y}\frac{\partial v_h}{\partial y} \right) d\Omega_h = \\ \qquad\qquad \displaystyle\int_{\Omega_h} f v_h d\Omega_h \quad , \quad \mathbf{x} \in \Omega_h \; t > 0 \\ u|_{\Gamma_h = \partial\Omega_h} = g \end{cases}$$

Diffusion Equation

Parabolic PDEs in FreeFEM++

2-D Diffusion Equation

- Weak form: FEM discretization

$$\begin{cases} \displaystyle\int_{\Omega_h} v_h \partial_t u_h d\Omega_h + \int_{\Omega_h} \sigma \left(\frac{\partial u_h}{\partial x}\frac{\partial v_h}{\partial x} + \frac{\partial u_h}{\partial y}\frac{\partial v_h}{\partial y} \right) d\Omega_h = \\ \qquad\qquad \displaystyle\int_{\Omega_h} f v_h d\Omega_h \quad , \quad \mathbf{x} \in \Omega_h \; t > 0 \\ u|_{\Gamma_h = \partial\Omega_h} = g \end{cases}$$

- Introducing the FT Implicit Euler scheme:

$$\begin{cases} \displaystyle\int_{\Omega_h} v_h \frac{u_h^{n+1} - u_h^n}{\Delta t} + \sigma \left(\frac{\partial u_h^{n+1}}{\partial x}\frac{\partial v_h}{\partial x} + \frac{\partial u_h^{n+1}}{\partial y}\frac{\partial v_h}{\partial y} \right) d\Omega_h = \\ \qquad\qquad \displaystyle\int_{\Omega_h} f^{n+1} v_h d\Omega_h \quad , \quad \mathbf{x} \in \Omega_h \; t > 0 \\ u|_{\Gamma_h = \partial\Omega_h} = g^{n+1} \end{cases}$$

Diffusion Equation

Parabolic PDEs in FreeFEM++

2-D Diffusion Equation: IEFT

- First case: $\Omega = [0,1] \times [0,1]$

```
1  mesh Omega = square(60,20);
2  func u0 =10+90*x/6;
3  func sigma = 0.1*(y<0.5)+sin(x);
4  fespace Vh(Omega,P1); Vh u = u0,v,uOld;
5  func f = x*sin(y); real uL = 25, uR = 35, uD = 15, uU =
       15, dt=0.1 ;
6  problem Diff(u, v) = + int2d(Omega)(u*v/dt+sigma*(dx(u)*
       dx(v)+dy(u)*dy(v)))
7  - int2d(Omega)(uOld*v/dt)-int2d(Omega)(f*v)
8  + on(1,u=uD)+on(2,u=uR)+on(3,u=uU)+on(4,u=uL);
9  for(real t=0; t<2; t+=dt){
10 uOld=u; Diff; endl;
11 plot(u);}
```

Diffusion Equation

[302.044] Numerical Methods in Fluid Dynamics

Object Oriented Programming in FreeFEM++

FreeFEM++: Object Oriented Programming

- First case: $\Omega = [0,1] \times [0,1]$

  Mesh Block

```
1  mesh Omega = square(60,20);
```

Diffusion Equation

[302.044] Numerical Methods in Fluid Dynamics

Object Oriented Programming in FreeFEM++

FreeFEM++: Object Oriented Programming

- First case: $\Omega = [0,1] \times [0,1]$

  Functional Space Block

```
1  func u0 =10+90*x/6;
2  func sigma = 0.1*(y<0.5)+sin(x);
3  fespace Vh(Omega,P1); Vh u = u0,v,uOld;
```

Diffusion Equation

[302.044] Numerical Methods in Fluid Dynamics

Object Oriented Programming in FreeFEM++

FreeFEM++: Object Oriented Programming

- First case: $\Omega = [0,1] \times [0,1]$

  Solver Block

```
1  func f = x*sin(y); real uL = 25, uR = 35, uD = 15,
       uU = 15, dt=0.1 ;
2  problem Diff(u, v) = + int2d(Omega)(u*v/dt+sigma*(dx
       (u)*dx(v)+dy(u)*dy(v)))
3  - int2d(Omega)(uOld*v/dt)
4  -int2d(Omega)(f*v)
5  + on(1,u=uD)+on(2,u=uR)+on(3,u=uU)+on(4,u=uL);
6  for(real t=0; t<2; t+=dt){
7  uOld=u; Diff; endl;
8  plot(u);}
```

Diffusion Equation

Object Oriented Programming in FreeFEM++

FreeFEM++: Object Oriented Programming

- Second case: $\Omega = [0, 6] \times [0, 1]$

  Mesh Block Changed

```
1  mesh Omega = square(60,20,[6*x,y]);
```

  Functional Space Block: Unchanged

  Solver Block: Unchanged