OFELI Example Codes

Rachid Touzani

Laboratoire de Mathématiques
Université Blaise Pascal
Clermont-Ferrand, France

Outlook

Example 1 : A 1-D Problem

```
double Lmin=0, Lmax=1;
int N = 20;
double f(double x);
Mesh ms(Lmin,Lmax,N);

TrMatrix<double> A(N-1);
Vect<double> b(N-1);
double h = (Lmax-Lmin)/double(N);

for (int i=2; i<N-1; i++) {
   double x = ms.getPtrNode(i)->getCoord(1);
   A(i,i) = 2./h;
   A(i,i+1) = -1./h;
   A(i,i-1) = -1./h;
   b(i) = f(x)*h;
}
A(1,1) = 2./h;
A(1,2) = -1./h;
A(N-1,N-2) = -1./h;
A(N-1,N-1) = 2./h;

A.Solve(b);
```

Example 1 : A 1-D Problem

Example 1

```
double Lmin=0, Lmax=1;
int N = 20;
double f(double x);
Mesh ms(Lmin,Lmax,N);

TrMatrix<double> A(N-1);
Vect<double> b(N-1);
double h = (Lmax-Lmin)/double(N);

for (int i=2; i<N-1; i++) {
   double x = ms.getPtrNode(i)->getCoord(1);
   A(i,i) = 2./h;
   A(i,i+1) = -1./h;
   A(i,i-1) = -1./h;
   b(i) = f(x)*h;
}
A(1,1) = 2./h;
A(1,2) = -1./h;
A(N-1,N-2) = -1./h;
A(N-1,N-1) = 2./h;

A.Solve(b);
```

Example 1 : A 1-D Problem

Example 1

```
double Lmin=0, Lmax=1;
int N = 20;
double f(double x);
Mesh ms(Lmin,Lmax,N);

TrMatrix<double> A(N-1);
Vect<double> b(N-1);
double h = (Lmax-Lmin)/double(N);

for (int i=2; i<N-1; i++) {
   double x = ms.getPtrNode(i)->getCoord(1);
   A(i,i) = 2./h;
   A(i,i+1) = -1./h;
   A(i,i-1) = -1./h;
   b(i) = f(x)*h;
}
A(1,1) = 2./h;
A(1,2) = -1./h;
A(N-1,N-2) = -1./h;
A(N-1,N-1) = 2./h;

A.Solve(b);
```

Example 1 : A 1-D Problem

Example 1

```
double Lmin=0, Lmax=1;
int N = 20;
double f(double x);
Mesh ms(Lmin,Lmax,N);

TrMatrix<double> A(N-1);
Vect<double> b(N-1);
double h = (Lmax-Lmin)/double(N);

for (int i=2; i<N-1; i++) {
   double x = ms.getPtrNode(i)->getCoord(1);
   A(i,i) = 2./h;
   A(i,i+1) = -1./h;
   A(i,i-1) = -1./h;
   b(i) = f(x)*h;
}
A(1,1) = 2./h;
A(1,2) = -1./h;
A(N-1,N-2) = -1./h;
A(N-1,N-1) = 2./h;

A.Solve(b);
```

Example 1 : A 1-D Problem

Example 1

```
double Lmin=0, Lmax=1;
int N = 20;
double f(double x);
Mesh ms(Lmin,Lmax,N);

TrMatrix<double> A(N-1);
Vect<double> b(N-1);
double h = (Lmax-Lmin)/double(N);

for (int i=2; i<N-1; i++) {
   double x = ms.getPtrNode(i)->getCoord(1);
   A(i,i) = 2./h;
   A(i,i+1) = -1./h;
   A(i,i-1) = -1./h;
   b(i) = f(x)*h;
}
A(1,1) = 2./h;
A(1,2) = -1./h;
A(N-1,N-2) = -1./h;
A(N-1,N-1) = 2./h;

A.Solve(b);
```

Example 1 : A 1-D Problem

Example 1

```
double Lmin=0, Lmax=1;
int N = 20;
double f(double x);
Mesh ms(Lmin,Lmax,N);

TrMatrix<double> A(N-1);
Vect<double> b(N-1);
double h = (Lmax-Lmin)/double(N);

for (int i=2; i<N-1; i++) {
   double x = ms.getPtrNode(i)->getCoord(1);
   A(i,i) = 2./h;
   A(i,i+1) = -1./h;
   A(i,i-1) = -1./h;
   b(i) = f(x)*h;
}
A(1,1) = 2./h;
A(1,2) = -1./h;
A(N-1,N-2) = -1./h;
A(N-1,N-1) = 2./h;

A.Solve(b);
```

Example 2: A *Black Box* Code (1/2)

Example 2

A Black Box Finite Element Code:
Diffusion-Convection Equation.

```
// Instantiate mesh and prescription
   Mesh ms(data.getMeshFile(),true);
   Prescription p(ms,data.getPrescriptionFile());

// Declare problem data (matrix, rhs, boundary conditions, body forces)
   NodeVect<double> u(ms,1,"Temperature");
   NodeVect<double> bc(ms), body_f(ms);
   p.get(BOUNDARY_CONDITION,bc);
   p.get(SOURCE,body_f);

// Read velocity for convection
   NodeVect<double> v(ms.getDim());
   IOField ff(data.getMeshFile(),data.getAuxFile(1),ms,XML_READ);
   ff.get(v);
```

Example 2: A *Black Box* Code (1/2)

Example 2

A Black Box Finite Element Code:
Diffusion-Convection Equation.

```
// Instantiate mesh and prescription
   Mesh ms(data.getMeshFile(),true);
   Prescription p(ms,data.getPrescriptionFile());

// Declare problem data (matrix, rhs, boundary conditions, body forces)
   NodeVect<double> u(ms,1,"Temperature");
   NodeVect<double> bc(ms), body_f(ms);
   p.get(BOUNDARY_CONDITION,bc);
   p.get(SOURCE,body_f);

// Read velocity for convection
   NodeVect<double> v(ms.getDim());
   IOField ff(data.getMeshFile(),data.getAuxFile(1),ms,XML_READ);
   ff.get(v);
```

Example 2: A *Black Box* Code (1/2)

Example 2

A Black Box Finite Element Code:
Diffusion-Convection Equation.

```
// Instantiate mesh and prescription
   Mesh ms(data.getMeshFile(),true);
   Prescription p(ms,data.getPrescriptionFile());

// Declare problem data (matrix, rhs, boundary conditions, body forces)
   NodeVect<double> u(ms,1,"Temperature");
   NodeVect<double> bc(ms), body_f(ms);
   p.get(BOUNDARY_CONDITION,bc);
   p.get(SOURCE,body_f);

// Read velocity for convection
   NodeVect<double> v(ms.getDim());
   IOField ff(data.getMeshFile(),data.getAuxFile(1),ms,XML_READ);
   ff.get(v);
```

Example 2: A *Black Box* Code (1/2)

Example 2

A Black Box Finite Element Code:
Diffusion-Convection Equation.

```
// Instantiate mesh and prescription
   Mesh ms(data.getMeshFile(),true);
   Prescription p(ms,data.getPrescriptionFile());

// Declare problem data (matrix, rhs, boundary conditions, body forces)
   NodeVect<double> u(ms,1,"Temperature");
   NodeVect<double> bc(ms), body_f(ms);
   p.get(BOUNDARY_CONDITION,bc);
   p.get(SOURCE,body_f);

// Read velocity for convection
   NodeVect<double> v(ms.getDim());
   IOField ff(data.getMeshFile(),data.getAuxFile(1),ms,XML_READ);
   ff.get(v);
```

Example 2: A *Black Box* Code (2/2)

```
// Set equation features and choose solver
   DC2DT3 eq(ms,u);
   eq.setInput(BOUNDARY_CONDITION,bc.getVect());
   eq.setInput(SOURCE,body_f.getVect());
   eq.setInput(VELOCITY_FIELD,v.getVect());
   eq.setTerms(DIFFUSION|CONVECTION);
   eq.setSolver(GMRES_SOLVER,ILU_PREC);

// Formation and solution of the linear system
   int it = eq.run();

// Output and save solution
   cout << u;
   if (data.getSave()) {
      IOField pf(data.getPlotFile(),XML_WRITE);
      pf.put(u);
   }
```

Example 2: A *Black Box* Code (2/2)

```
// Set equation features and choose solver
   DC2DT3 eq(ms,u);
   eq.setInput(BOUNDARY_CONDITION,bc.getVect());
   eq.setInput(SOURCE,body_f.getVect());
   eq.setInput(VELOCITY_FIELD,v.getVect());
   eq.setTerms(DIFFUSION|CONVECTION);
   eq.setSolver(GMRES_SOLVER,ILU_PREC);

// Formation and solution of the linear system
   int it = eq.run();

// Output and save solution
   cout << u;
   if (data.getSave()) {
      IOField pf(data.getPlotFile(),XML_WRITE);
      pf.put(u);
   }
```

Example 2: A *Black Box* Code (2/2)

```
// Set equation features and choose solver
   DC2DT3 eq(ms,u);
   eq.setInput(BOUNDARY_CONDITION,bc.getVect());
   eq.setInput(SOURCE,body_f.getVect());
   eq.setInput(VELOCITY_FIELD,v.getVect());
   eq.setTerms(DIFFUSION|CONVECTION);
   eq.setSolver(GMRES_SOLVER,ILU_PREC);

// Formation and solution of the linear system
   int it = eq.run();

// Output and save solution
   cout << u;
   if (data.getSave()) {
      IOField pf(data.getPlotFile(),XML_WRITE);
      pf.put(u);
   }
```

Example 2: A *Black Box* Code (2/2)

```
// Set equation features and choose solver
   DC2DT3 eq(ms,u);
   eq.setInput(BOUNDARY_CONDITION,bc.getVect());
   eq.setInput(SOURCE,body_f.getVect());
   eq.setInput(VELOCITY_FIELD,v.getVect());
   eq.setTerms(DIFFUSION|CONVECTION);
   eq.setSolver(GMRES_SOLVER,ILU_PREC);

// Formation and solution of the linear system
   int it = eq.run();

// Output and save solution
   cout << u;
   if (data.getSave()) {
      IOField pf(data.getPlotFile(),XML_WRITE);
      pf.put(u);
   }
```

Example 3: An optimization Problem (1/3)

Consider the following problem:

$$\mathbf{u} \in \mathscr{V};\ W(\mathbf{u}) = \inf_{\mathbf{v} \in \mathscr{V}} W(\mathbf{v})$$

where

$$\mathscr{V} := \{\mathbf{v} \in \mathbb{R}^N;\ a_i \leq v_i \leq b_i,\ 1 \leq i \leq N\}$$

To solve this problem with **OFELI**, we write a C++ code:

Example 3: An optimization Problem (1/3)

Consider the following problem:

$$\mathbf{u} \in \mathscr{V};\ W(\mathbf{u}) = \inf_{\mathbf{v} \in \mathscr{V}} W(\mathbf{v})$$

where

$$\mathscr{V} := \{\mathbf{v} \in \mathbb{R}^N;\ a_i \le v_i \le b_i,\ 1 \le i \le N\}$$

To solve this problem with **OFELI**, we write a C++ code:

Example 3: An optimization Problem (1/3)

Example 3

Consider the following problem:

$$\mathbf{u} \in \mathscr{V};\ W(\mathbf{u}) = \inf_{\mathbf{v}\in\mathscr{V}} W(\mathbf{v})$$

where

$$\mathscr{V} := \{\mathbf{v} \in \mathbb{R}^N;\ a_i \le v_i \le b_i,\ 1 \le i \le N\}$$

To solve this problem with OFELI, we write a C++ code:

Example 3: An optimization Problem (1/3)

Example 3

Consider the following problem:

$$\mathbf{u} \in \mathscr{V}; \;\; W(\mathbf{u}) = \inf_{\mathbf{v} \in \mathscr{V}} W(\mathbf{v})$$

where

$$\mathscr{V} := \{\mathbf{v} \in \mathbb{R}^N; \;\; a_i \leq v_i \leq b_i, \;\; 1 \leq i \leq N\}$$

To solve this problem with OFELI, we write a C++ code:

Example 3: An optimization Problem (1/3)

Example 3

Consider the following problem:

$$\mathbf{u} \in \mathscr{V}; \;\; W(\mathbf{u}) = \inf_{\mathbf{v} \in \mathscr{V}} W(\mathbf{v})$$

where

$$\mathscr{V} := \{\mathbf{v} \in \mathbb{R}^N; \;\; a_i \leq v_i \leq b_i, \;\; 1 \leq i \leq N\}$$

To solve this problem with OFELI, we write a C++ code:

Example 3: An optimization Problem (1/3)

Example 3

Consider the following problem:

$$\mathbf{u} \in \mathscr{V};\;\; W(\mathbf{u}) = \inf_{\mathbf{v}\in\mathscr{V}} W(\mathbf{v})$$

where

$$\mathscr{V} := \{\mathbf{v} \in \mathbb{R}^N;\;\; a_i \leq v_i \leq b_i,\;\; 1 \leq i \leq N\}$$

To solve this problem with **OFELI**, we write a C++ code:

Example 3: An optimization Problem (2/3)

```
Mesh ms("test.m");
User ud(ms);
Vect<double> x(ms.getNbDOF());
Vect<double> low(ms.getNbDOF()), up(ms.getNbDOF());
Vect<int> pivot(ms.getNbDOF());

Vect<double> bc(ms.getNbDOF());
ud.setDBC(bc);

Opt theOpt(ms,ud);
BCAsConstraint(ms,bc,up,low);

OptimTN<Opt>(theOpt,x,low,up,pivot,100,1.e-8,1);
```

Example 3: An optimization Problem (3/3)

The class `Opt` is defined as follows:

```
class Opt {

   public:
      Opt(Mesh &ms, User &ud);
      void Objective(const Vect<double> &x, double &f, Vect<double> &g);

   private:
      Mesh *_ms;
      User *_ud;
};
```

Example 3: An optimization Problem (3/3)

The class `Opt` is defined as follows:

```
class Opt {

  public:
    Opt(Mesh &ms, User &ud);
    void Objective(const Vect<double> &x, double &f, Vect<double> &g);

  private:
    Mesh *_ms;
    User *_ud;
};
```

Example 4: Mixed Hybrid Finite Elements (1/6)

This example illustrates the use of non standard methods in **OFELI** (Mixed Elements, Finite Volumes, . . .) Consider the problem

$$\begin{aligned} \Delta u = 0 \qquad & \text{in } \Omega \subset \mathbb{R}^2 \\ u = g \qquad & \text{on } \partial\Omega \end{aligned}$$

This problem is equivalent to:

$$\mathbf{p} - \nabla u = 0, \; -\nabla \cdot \mathbf{p} = f \qquad \text{in } \Omega \subset \mathbb{R}^2, \qquad u = g \qquad \text{on } \partial\Omega$$

The approximation by mixed hybrid finite elements consists in defining the spaces:

$$\begin{aligned} &\mathscr{V} = \{v \in L^2(\Omega);\; v_{|T} = \text{Const.} \quad \forall\, T \in \mathscr{T}\}, \\ &\mathscr{Q} = \{\mathbf{q} \in L^2(\Omega)^2;\; q_{|T} = \mathbf{a}_T + b_T\mathbf{x},\; \mathbf{a}_T \in \mathbb{R}^2,\; b_T \in \mathbb{R}\; \forall\, T \in \mathscr{T}\}, \\ &\mathscr{M} = \{\mu;\; \mu_{|e} = \text{Const.} \quad \forall\, e \in \mathscr{E}\}. \end{aligned}$$

where $\mathscr{T}$: triangles, $\mathscr{E}$: edges.

Example 4: Mixed Hybrid Finite Elements (1/6)

This example illustrates the use of non standard methods in **OFELI** (Mixed Elements, Finite Volumes, ...) Consider the problem

$$\begin{aligned} \Delta u = 0 \qquad & \text{in } \Omega \subset \mathbb{R}^2 \\ u = g \qquad & \text{on } \partial\Omega \end{aligned}$$

This problem is equivalent to:

$$\mathbf{p} - \nabla u = 0, \ -\nabla \cdot \mathbf{p} = f \qquad \text{in } \Omega \subset \mathbb{R}^2, \qquad u = g \qquad \text{on } \partial\Omega$$

The approximation by mixed hybrid finite elements consists in defining the spaces:

$$\begin{aligned} &\mathscr{V} = \{v \in L^2(\Omega);\ v_{|T} = \text{Const.} \quad \forall\ T \in \mathscr{T}\}, \\ &\mathscr{Q} = \{\mathbf{q} \in L^2(\Omega)^2;\ q_{|T} = \mathbf{a}_T + b_T\mathbf{x},\ \mathbf{a}_T \in \mathbb{R}^2,\ b_T \in \mathbb{R}\ \forall\ T \in \mathscr{T}\}, \\ &\mathscr{M} = \{\mu;\ \mu_{|e} = \text{Const.} \quad \forall\ e \in \mathscr{E}\}. \end{aligned}$$

where $\mathscr{T}$: triangles, $\mathscr{E}$: edges.

Example 4: Mixed Hybrid Finite Elements (1/6)

This example illustrates the use of non standard methods in **OFELI** (Mixed Elements, Finite Volumes, ...) Consider the problem

$$\Delta u = 0 \qquad \text{in } \Omega \subset \mathbb{R}^2$$
$$u = g \qquad \text{on } \partial\Omega$$

This problem is equivalent to:

$$\mathbf{p} - \nabla u = 0, \ -\nabla \cdot \mathbf{p} = f \qquad \text{in } \Omega \subset \mathbb{R}^2, \qquad u = g \qquad \text{on } \partial\Omega$$

The approximation by mixed hybrid finite elements consists in defining the spaces:

$$\mathscr{V} = \{v \in L^2(\Omega);\ v_{|T} = \text{Const.} \quad \forall\ T \in \mathscr{T}\},$$
$$\mathscr{Q} = \{\mathbf{q} \in L^2(\Omega)^2;\ q_{|T} = \mathbf{a}_T + b_T\mathbf{x},\ \mathbf{a}_T \in \mathbb{R}^2,\ b_T \in \mathbb{R}\ \forall\ T \in \mathscr{T}\},$$
$$\mathscr{M} = \{\mu;\ \mu_{|e} = \text{Const.} \quad \forall\ e \in \mathscr{E}\}.$$

where $\mathscr{T}$: triangles, $\mathscr{E}$: edges.

Example 4: Mixed Hybrid Finite Elements (1/6)

This example illustrates the use of non standard methods in **OFELI** (Mixed Elements, Finite Volumes, ...) Consider the problem

$$
\begin{aligned}
\Delta u = 0 &\qquad \text{in } \Omega \subset \mathbb{R}^2 \\
u = g &\qquad \text{on } \partial\Omega
\end{aligned}
$$

This problem is equivalent to:

$$
\mathbf{p} - \nabla u = 0, \; -\nabla \cdot \mathbf{p} = f \qquad \text{in } \Omega \subset \mathbb{R}^2, \qquad u = g \qquad \text{on } \partial\Omega
$$

The approximation by mixed hybrid finite elements consists in defining the spaces:

$$
\begin{aligned}
&\mathscr{V} = \{v \in L^2(\Omega); \; v_{|T} = \text{Const.} \quad \forall \; T \in \mathscr{T}\}, \\
&\mathscr{Q} = \{\mathbf{q} \in L^2(\Omega)^2; \; q_{|T} = \mathbf{a}_T + b_T\mathbf{x}, \; \mathbf{a}_T \in \mathbb{R}^2, \; b_T \in \mathbb{R} \; \forall \; T \in \mathscr{T}\}, \\
&\mathscr{M} = \{\mu; \; \mu_{|e} = \text{Const.} \quad \forall \; e \in \mathscr{E}\}.
\end{aligned}
$$

where $\mathscr{T}$: triangles, $\mathscr{E}$: edges.

Mixed Hybrid Finite Elements (2/6)

We then look for a triple $(u, \mathbf{p}, \lambda) \in \mathscr{V} \times \mathscr{Q} \times \mathscr{M}$ such that:

$$\begin{aligned}
&\int_T \mathbf{p}\cdot\mathbf{q}\,d\mathbf{x} + \int_T u\,\nabla\cdot\mathbf{q}\,d\mathbf{x} - \sum_{e\in\mathscr{E}_T}\int_e \lambda\mathbf{q}\cdot\mathbf{n}\,ds = \sum_{e\in\mathscr{E}_T^D}\int_e g\mathbf{q}\cdot\mathbf{n}\,ds && \forall\ \mathbf{q}\in\mathscr{Q},\ T\in\mathscr{T},\\
&\int_T \nabla\cdot\mathbf{p}\,d\mathbf{x} = -\int_T f\,d\mathbf{x}, && \forall\ T\in\mathscr{T},\\
&\sum_{T\in\mathscr{T}}\sum_{e\in\mathscr{E}_T}\int_e \mu\,\mathbf{p}\cdot\mathbf{n}\,ds = 0 && \forall\ \mu\in\mathscr{M}
\end{aligned}$$

After some calculus, we obtain for λ the linear system

$$\sum_{T\in\mathscr{T}_e}\Big(\frac{1}{|T|}\sum_{e'\in\mathscr{E}_T}\ell_e\ell_{e'}\mathbf{n}_T^e\cdot\mathbf{n}_T^{e'}\Big)\lambda_{e'} = -\sum_{T\in\mathscr{T}_e}\ell_e\mathbf{n}_T^e\cdot\Big(\frac{1}{2}f_T\,(\mathbf{c}_e-\mathbf{c}_T)+\sum_{e'\in\mathscr{E}_T^D} g_{e'}\ell_{e'}\mathbf{n}_T^{e'}\Big) \qquad e\in\mathscr{E}$$

Mixed Hybrid Finite Elements (2/6)

We then look for a triple $(u, \mathbf{p}, \lambda) \in \mathscr{V} \times \mathscr{Q} \times \mathscr{M}$ such that:

$$\int_T \mathbf{p} \cdot \mathbf{q}\, dx + \int_T u\, \nabla \cdot \mathbf{q}\, dx - \sum_{e \in \mathscr{E}_T} \int_e \lambda \mathbf{q} \cdot \mathbf{n}\, ds = \sum_{e \in \mathscr{E}_T^D} \int_e g \mathbf{q} \cdot \mathbf{n}\, ds \qquad \forall\ \mathbf{q} \in \mathscr{Q},\ T \in \mathscr{T},$$

$$\int_T \nabla \cdot \mathbf{p}\, dx = -\int_T f\, dx, \qquad \forall\ T \in \mathscr{T},$$

$$\sum_{T \in \mathscr{T}} \sum_{e \in \mathscr{E}_T} \int_e \mu\, \mathbf{p} \cdot \mathbf{n}\, ds = 0 \qquad \forall\ \mu \in \mathscr{M}$$

After some calculus, we obtain for λ the linear system

$$\sum_{T \in \mathscr{T}_e} \Big(\frac{1}{|T|} \sum_{e' \in \mathscr{E}_T} \ell_e \ell_{e'} \mathbf{n}_T^e \cdot \mathbf{n}_T^{e'} \Big) \lambda_{e'} = -\sum_{T \in \mathscr{T}_e} \ell_e \mathbf{n}_T^e \cdot \Big(\frac{1}{2} f_T\, (\mathbf{c}_e - \mathbf{c}_T) + \sum_{e' \in \mathscr{E}_T^D} g_{e'} \ell_{e'} \mathbf{n}_T^{e'} \Big) \quad e \in \mathscr{E}$$

Mixed Hybrid Finite Elements (3/6)

Implementation: The main program

```
   Mesh ms("test.m");
   ms.setDOFSupport(SIDE_DOF);
   ms.removeImposedDOF();

   SpMatrix<double> A(ms);
   Vect<double> b(ms.getNbEq()), lambda(ms.getNbSides());
   Vect<double> f(ms.getNbElements()), g(ms.getNbSides());
// Initialize vectors f and g
   ...

   Laplace2DMHRT0 eq(ms,A,b);
   eq.build(f,g);
   eq.solve(lambda);
```

Mixed Hybrid Finite Elements (3/6)

Implementation: The main program

```
Mesh ms("test.m");
ms.setDOFSupport(SIDE_DOF);
ms.removeImposedDOF();

SpMatrix<double> A(ms);
Vect<double> b(ms.getNbEq()), lambda(ms.getNbSides());
Vect<double> f(ms.getNbElements()), g(ms.getNbSides());
// Initialize vectors f and g
...

Laplace2DMHRT0 eq(ms,A,b);
eq.build(f,g);
eq.solve(lambda);
```

Mixed Hybrid Finite Elements (3/6)

Implementation: The main program

```
   Mesh ms("test.m");
   ms.setDOFSupport(SIDE_DOF);
   ms.removeImposedDOF();

   SpMatrix<double> A(ms);
   Vect<double> b(ms.getNbEq()), lambda(ms.getNbSides());
   Vect<double> f(ms.getNbElements()), g(ms.getNbSides());
// Initialize vectors f and g
   ...

   Laplace2DMHRT0 eq(ms,A,b);
   eq.build(f,g);
   eq.solve(lambda);
```

Mixed Hybrid Finite Elements (3/6)

Implementation: The main program

```
   Mesh ms("test.m");
   ms.setDOFSupport(SIDE_DOF);
   ms.removeImposedDOF();

   SpMatrix<double> A(ms);
   Vect<double> b(ms.getNbEq()), lambda(ms.getNbSides());
   Vect<double> f(ms.getNbElements()), g(ms.getNbSides());
// Initialize vectors f and g
   ...

   Laplace2DMHRT0 eq(ms,A,b);
   eq.build(f,g);
   eq.solve(lambda);
```

Mixed Hybrid Finite Elements (4/6)

Implementation: The class Laplace2DMHRT0

```
class Laplace2DMHRT0 :  virtual public FE_Laplace<double,3,3,2,2> {

public :
   Laplace2DMHRT0();
   Laplace2DMHRT0(Mesh &ms, SpMatrix<double> &A, Vect<double> &b);
   ~Laplace2DMHRT0();
   void build(const Vect<double> &f, const Vect<double> &g);
   void Post(const Vect<double> &lambda, const Vect<double> &f
             Vect<double> &v, Vect<Point<double> > &p, Vect<double> &u);
   int solve(Vect<double> &u);

private :
   SpMatrix<double> *_A;
   Vect<double> *_b;
   const Vect<double> *_f, *_g;
   Triang3 *_tr;
   Side *_sd1, *_sd2, *_sd3;
   LocalVect<Point<double>,3> _n, _ce;
   void ElementSet(const Element *el);
   void LM_LHS();
   void LM_RHS();
};
```

Mixed Hybrid Finite Elements (4/6)

Implementation: The class `Laplace2DMHRT0`

```
class Laplace2DMHRT0 :  virtual public FE_Laplace<double,3,3,2,2> {
public :
   Laplace2DMHRT0();
   Laplace2DMHRT0(Mesh &ms, SpMatrix<double> &A, Vect<double> &b);
   ~Laplace2DMHRT0();
   void build(const Vect<double> &f, const Vect<double> &g);
   void Post(const Vect<double> &lambda, const Vect<double> &f
             Vect<double> &v, Vect<Point<double> > &p, Vect<double> &u);
   int solve(Vect<double> &u);

private :
   SpMatrix<double> *_A;
   Vect<double> *_b;
   const Vect<double> *_f, *_g;
   Triang3 *_tr;
   Side *_sd1, *_sd2, *_sd3;
   LocalVect<Point<double>,3> _n, _ce;
   void ElementSet(const Element *el);
   void LM_LHS();
   void LM_RHS();
};
```

Mixed Hybrid Finite Elements (4/6)

Implementation: The class Laplace2DMHRT0

```
class Laplace2DMHRT0 :  virtual public FE_Laplace<double,3,3,2,2> {

public :
   Laplace2DMHRT0();
   Laplace2DMHRT0(Mesh &ms, SpMatrix<double> &A, Vect<double> &b);
   ~Laplace2DMHRT0();
   void build(const Vect<double> &f, const Vect<double> &g);
   void Post(const Vect<double> &lambda, const Vect<double> &f
             Vect<double> &v, Vect<Point<double> > &p, Vect<double> &u);
   int solve(Vect<double> &u);

private :
   SpMatrix<double> *_A;
   Vect<double> *_b;
   const Vect<double> *_f, *_g;
   Triang3 *_tr;
   Side *_sd1, *_sd2, *_sd3;
   LocalVect<Point<double>,3> _n, _ce;
   void ElementSet(const Element *el);
   void LM_LHS();
   void LM_RHS();
};
```

Mixed Hybrid Finite Elements (4/6)

Implementation: The class Laplace2DMHRT0

```
class Laplace2DMHRT0 :  virtual public FE_Laplace<double,3,3,2,2> {
public :
   Laplace2DMHRT0();
   Laplace2DMHRT0(Mesh &ms, SpMatrix<double> &A, Vect<double> &b);
   ~Laplace2DMHRT0();
   void build(const Vect<double> &f, const Vect<double> &g);
   void Post(const Vect<double> &lambda, const Vect<double> &f
             Vect<double> &v, Vect<Point<double> > &p, Vect<double> &u);
   int solve(Vect<double> &u);

private :
   SpMatrix<double> *_A;
   Vect<double> *_b;
   const Vect<double> *_f, *_g;
   Triang3 *_tr;
   Side *_sd1, *_sd2, *_sd3;
   LocalVect<Point<double>,3> _n, _ce;
   void ElementSet(const Element *el);
   void LM_LHS();
   void LM_RHS();
};
```

Mixed Hybrid Finite Elements (4/6)

Implementation: The class `Laplace2DMHRT0`

```
class Laplace2DMHRT0 :  virtual public FE_Laplace<double,3,3,2,2> {
public :
   Laplace2DMHRT0();
   Laplace2DMHRT0(Mesh &ms, SpMatrix<double> &A, Vect<double> &b);
   ~Laplace2DMHRT0();
   void build(const Vect<double> &f, const Vect<double> &g);
   void Post(const Vect<double> &lambda, const Vect<double> &f
             Vect<double> &v, Vect<Point<double> > &p, Vect<double> &u);
   int solve(Vect<double> &u);

private :
   SpMatrix<double> *_A;
   Vect<double> *_b;
   const Vect<double> *_f, *_g;
   Triang3 *_tr;
   Side *_sd1, *_sd2, *_sd3;
   LocalVect<Point<double>,3> _n, _ce;
   void ElementSet(const Element *el);
   void LM_LHS();
   void LM_RHS();
};
```

Mixed Hybrid Finite Elements (4/6)

Implementation: The class Laplace2DMHRT0

```
class Laplace2DMHRT0 :  virtual public FE_Laplace<double,3,3,2,2> {
public :
   Laplace2DMHRT0();
   Laplace2DMHRT0(Mesh &ms, SpMatrix<double> &A, Vect<double> &b);
   ~Laplace2DMHRT0();
   void build(const Vect<double> &f, const Vect<double> &g);
   void Post(const Vect<double> &lambda, const Vect<double> &f
             Vect<double> &v, Vect<Point<double> > &p, Vect<double> &u);
   int solve(Vect<double> &u);

private :
   SpMatrix<double> *_A;
   Vect<double> *_b;
   const Vect<double> *_f, *_g;
   Triang3 *_tr;
   Side *_sd1, *_sd2, *_sd3;
   LocalVect<Point<double>,3> _n, _ce;
   void ElementSet(const Element *el);
   void LM_LHS();
   void LM_RHS();
};
```

Mixed Hybrid Finite Elements (4/6)

Implementation: The class Laplace2DMHRT0

```
class Laplace2DMHRT0 :  virtual public FE_Laplace<double,3,3,2,2> {
public :
   Laplace2DMHRT0();
   Laplace2DMHRT0(Mesh &ms, SpMatrix<double> &A, Vect<double> &b);
   ~Laplace2DMHRT0();
   void build(const Vect<double> &f, const Vect<double> &g);
   void Post(const Vect<double> &lambda, const Vect<double> &f
             Vect<double> &v, Vect<Point<double> > &p, Vect<double> &u);
   int solve(Vect<double> &u);
private :
   SpMatrix<double> *_A;
   Vect<double> *_b;
   const Vect<double> *_f, *_g;
   Triang3 *_tr;
   Side *_sd1, *_sd2, *_sd3;
   LocalVect<Point<double>,3> _n, _ce;
   void ElementSet(const Element *el);
   void LM_LHS();
   void LM_RHS();
};
```

Mixed Hybrid Finite Elements (4/6)

Implementation: The class `Laplace2DMHRT0`

```
class Laplace2DMHRT0 :  virtual public FE_Laplace<double,3,3,2,2> {
public :
   Laplace2DMHRT0();
   Laplace2DMHRT0(Mesh &ms, SpMatrix<double> &A, Vect<double> &b);
   ~Laplace2DMHRT0();
   void build(const Vect<double> &f, const Vect<double> &g);
   void Post(const Vect<double> &lambda, const Vect<double> &f
             Vect<double> &v, Vect<Point<double> > &p, Vect<double> &u);
   int solve(Vect<double> &u);
private :
   SpMatrix<double> *_A;
   Vect<double> *_b;
   const Vect<double> *_f, *_g;
   Triang3 *_tr;
   Side *_sd1, *_sd2, *_sd3;
   LocalVect<Point<double>,3> _n, _ce;
   void ElementSet(const Element *el);
   void LM_LHS();
   void LM_RHS();
};
```

Mixed Hybrid Finite Elements (5/6)

```
Laplace2DMHRT0::Laplace2DMHRT0(Mesh &ms, SpMatrix<double> &A, Vect<double> &b)
{
   _theMesh = &ms;
   _A = &A;
   _b = &b;
}

void Laplace2DMHRT0::ElementSet(const Element *el)
{
// Some geometric stuff
}

void Laplace2DMHRT0::LM_LHS()
{
   for (size_t i=1; i<=3; i++)
      for (size_t j=1; j<=3; j++)
         eMat(i,j) = _n(i)*_n(j)/_area;
}
```

Mixed Hybrid Finite Elements (5/6)

```
Laplace2DMHRT0::Laplace2DMHRT0(Mesh &ms, SpMatrix<double> &A, Vect<double> &b)
{
   _theMesh = &ms;
   _A = &A;
   _b = &b;
}

void Laplace2DMHRT0::ElementSet(const Element *el)
{
// Some geometric stuff
}

void Laplace2DMHRT0::LM_LHS()
{
   for (size_t i=1; i<=3; i++)
      for (size_t j=1; j<=3; j++)
         eMat(i,j) = _n(i)*_n(j)/_area;
}
```

Mixed Hybrid Finite Elements (5/6)

```
Laplace2DMHRT0::Laplace2DMHRT0(Mesh &ms, SpMatrix<double> &A, Vect<double> &b)
{
   _theMesh = &ms;
   _A = &A;
   _b = &b;
}

void Laplace2DMHRT0::ElementSet(const Element *el)
{
// Some geometric stuff
}

void Laplace2DMHRT0::LM_LHS()
{
   for (size_t i=1; i<=3; i++)
      for (size_t j=1; j<=3; j++)
         eMat(i,j) = _n(i)*_n(j)/_area;
}
```

Mixed Hybrid Finite Elements (6/6)

```
void Laplace2DMHRT0::build(const Vect<double> &f, const Vect<double> &g)
{
   Element *el;
   MeshElementLoop(*_theMesh,el) {
      ElementSet(el);
      _g = &g;
      _f = &f;
      LM_LHS();
      LM_RHS();
      SideAssembly(*el,EA(),*_A);
      SideAssembly(*el,Eb(),*_b);
   }
}

int Laplace2DMHRT0::solve(Vect<double> &u)
{
   double toler = 1.e-8;
   Vect<double> x;
   int nb_it = CG(*_A,Prec<double>(*_A,ILU_PREC),*_b,x,1000,toler,1);
   return nb_it;
}
```

Mixed Hybrid Finite Elements (6/6)

```
void Laplace2DMHRT0::build(const Vect<double> &f, const Vect<double> &g)
{
   Element *el;
   MeshElementLoop(*_theMesh,el) {
      ElementSet(el);
      _g = &g;
      _f = &f;
      LM_LHS();
      LM_RHS();
      SideAssembly(*el,EA(),*_A);
      SideAssembly(*el,Eb(),*_b);
   }
}

int Laplace2DMHRT0::solve(Vect<double> &u)
{
   double toler = 1.e-8;
   Vect<double> x;
   int nb_it = CG(*_A,Prec<double>(*_A,ILU_PREC),*_b,x,1000,toler,1);
   return nb_it;
}
```

Mixed Hybrid Finite Elements (6/6)

```
void Laplace2DMHRT0::build(const Vect<double> &f, const Vect<double> &g)
{
   Element *el;
   MeshElementLoop(*_theMesh,el) {
      ElementSet(el);
      _g = &g;
      _f = &f;
      LM_LHS();
      LM_RHS();
      SideAssembly(*el,EA(),*_A);
      SideAssembly(*el,Eb(),*_b);
   }
}

int Laplace2DMHRT0::solve(Vect<double> &u)
{
   double toler = 1.e-8;
   Vect<double> x;
   int nb_it = CG(*_A,Prec<double>(*_A,ILU_PREC),*_b,x,1000,toler,1);
   return nb_it;
}
```