Introduzione a FreeFem++

Manolo Venturin

EnginSoft, Padova

14 maggio 2012

Indice della presentazione

FreeFem++

- Risolve problemi 2D e 3D;
- Formulazione variazionale dei problemi;
- Free software (Linux, Windows e Mac)[1]
- Facile da installare;
- Sviluppato da: Olivier Pironneau, Frédéric Hecht, Antoine Le Hyaric, Jacques Morice;
- Scritto in C++ e con una grammatica simile;

[1] **FreeFem++ web site**

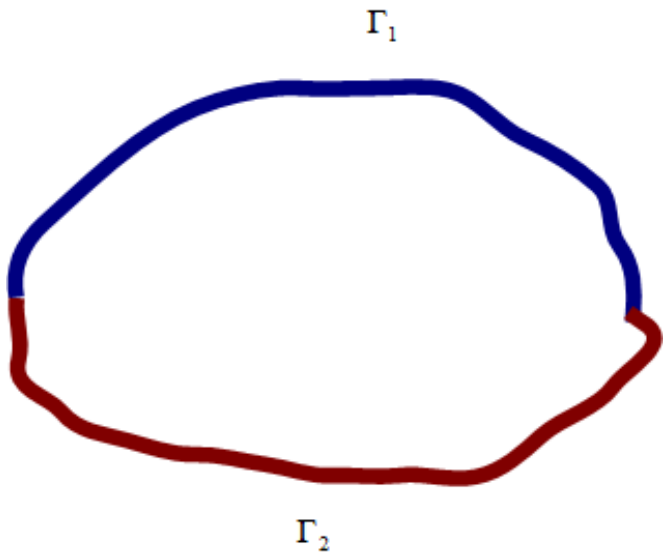
FreeFem++

- Include:
  - Generazione/caricamento di mesh;
  - Molti tipi di elementi finiti disponibili;
  - Molti solutori lineari inclusi e librerie incluse: CG, GMRES, UMFPACK, SUPERLU, MUMPS;
  - Matrici in formato sparso;
  - Strumento per visualizzare i risultati.
- Elimina complicati overhead di programmazione (geometria, mesh, assemblaggio, interpolazione, formule di quadratura) concentrandosi maggiormente sul problema invece che sulla sua implementazione;
- I problemi possono essere scritti direttamente nella loro formulazione variazionale;
- Ben documentato e con molti esempi;
- Facile per provare nuove idee senza scrivere centinaia di righe di codice.

Il problema

Dato $f \in L^2(\Omega)$, trovare $\varphi \in H_0^1(\Omega)$ tale che

$$\begin{cases} -\Delta\varphi = f & \text{in} \quad \Omega \\ \varphi(x,y) = 0 & \text{su} \quad \Gamma_1 \\ \partial_n\varphi = \nabla\varphi \cdot n = 0 & \text{su} \quad \Gamma_2 \end{cases}$$



Formulazione variazionale

La fomulazione debole o variazionale è ottenuta moltiplicando l'equazione per la generica funzione test

$$w \in V \equiv H^1_{\Gamma_1} = \{w \in H^1(\Omega) : w|_{\Gamma_1} = 0\}$$

e integrando per parti il termine Laplaciano si ottiene

$$\int_\Omega \nabla\varphi \cdot \nabla w \; d\omega = \int_\Omega fw \; d\omega$$

Formulazione variazionale

Equivalentemente il problema può essere rifomulato come: trovare $\varphi \in V$ tala che

$$a(\varphi, w) = l(w) \quad \forall v \in V$$

dove la forma bilineare è

$$a(\varphi, w) = \int_\Omega \nabla\varphi \cdot \nabla w \, d\omega$$

e quella lineare

$$l(w) = \int_\Omega fw \, d\omega$$

È facile provare che la forma bilineare è continua e coerciva (oltre che simmetrica) in V e di conseguenza la soluzione esiste ed è unica.

Approssimazione numerica

Indicato con V_h il sottospazio degli elementi finiti lineari di V, la fomulazione di Galerkin è: trovare $\varphi_h \in V_h$ tale che

$$a(\varphi_h, w_h) = l(w_h) \quad \forall v_h \in V_h$$

$V_h \subset V$ con $dim V_h = n_h$ è definito come

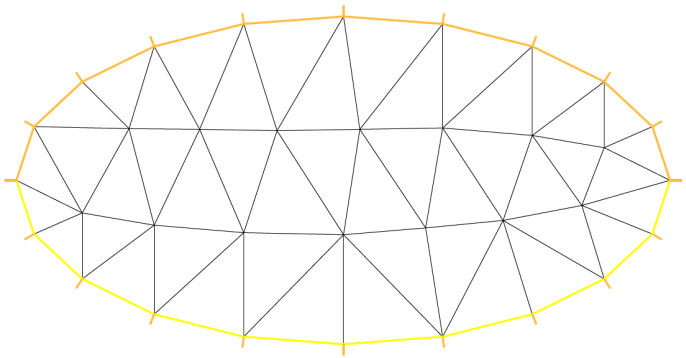
$$V_h = \left\{ \varphi(x,y) : \varphi(x,y) = \sum_{k=1}^{n_h} \varphi_k \phi_k(x,y), \varphi_k \in \mathbb{R} \right\}$$

dove $\phi_k \in P_s$ è un polinomio di grado s.

Approssimazione numerica

Lo spazio $V_h = V_h(\Omega_h, P)$ dipende dalla mesh:

$$\Omega_h = \bigcup_{e=1}^{n_e} T_e$$



e dall'approssimazione P

- P_0 approssimazione constante a tratti;
- P_1 approssimazione lineare a tratti;
- ...

Approssimazione numerica

Quindi il problema diventa: trovare $\varphi_h \in V_h$ tale che

$$a(\varphi_h, w_h) = l(w_h) \quad \forall v_h \in V_h$$

che equivale a risolvere il seguente sistema lineare

$$a(\phi_j, \phi_i) = l(\phi_i) \quad \forall i = \{1, \ldots, n_h\}.$$

Approssimazione con FreeFem++

Per risolvere numericamente l'equazione di Laplace, bisogna eseguire i seguenti passi:

1. Dichiarazione dei alcuni parametri globali;
2. Descrizione della geometria;
3. Calcolo della mesh del problema;
4. Scrittura della formulazione variazione;
5. Visualizzazione dei risultati.

Parametri globali

```
//  ************************************************************************
//  FILE        : Laplace.edp
//  DESCRIPTION : Laplace equation over an ellipse
//  AUTHOR      : M. Venturin
//  ************************************************************************

//  Problem prefix
//  ______________
string savedir = "results_laplace/";// Save directory
string problemPrefix = "laplace";   // Problem prefix used in output files
```

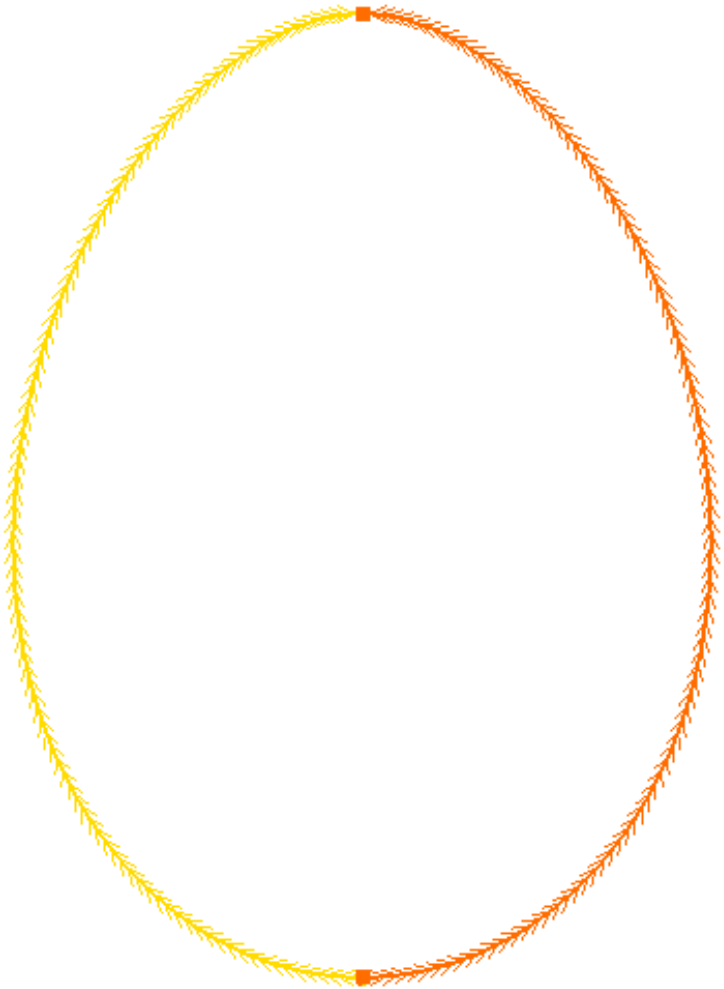
Sono state create due variabili di tipo string utilizzate successivamente per il salvataggio delle immagini.

Geometria

```
// Defining the boundary as an egg shape
real eggheight = 1;
real a = eggheight/2;
border Gamma1(t=0,pi){x=0.78*a*cos(t/4)*sin(t); y=-a*(cos(t)-1);}
border Gamma2(t=0,pi){x=-0.78*a*cos(t/4)*sin(t); y=-a*(cos(t)-1);}

// Geometry
int nbdiv = 101;     // Number of subdivisions
func geom = Gamma1(nbdiv) + Gamma2(-nbdiv) ;
plot(geom,wait=true,bw=1,ps=savedir + problemPrefix + "_geom.eps");
```

Il bordo viene definito in modo parametrico attraverso il comando **border**. È possibile definire particolare domini con buchi oppure no a seconda della orientazione del bordo (si veda la documentazione ufficiale).

Geometria

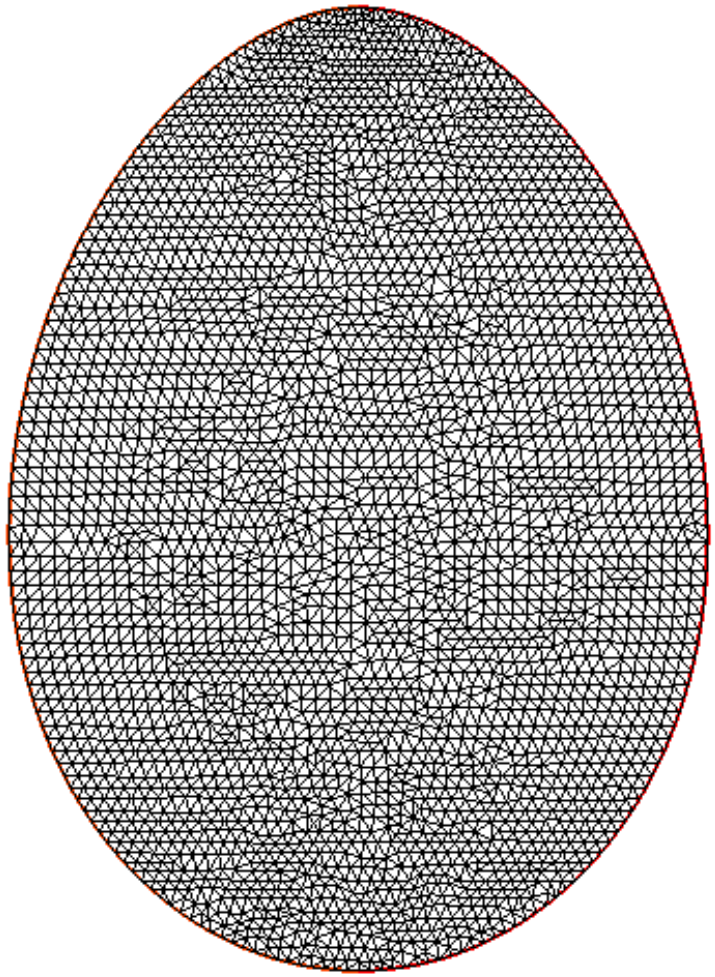


Mesh

```
// Mesh
mesh Th = buildmesh(geom);
plot(Th, wait=true, bw=1, ps=savedir + problemPrefix + "_mesh.eps");
```

La mesh viene calcolata utilizzando il comando **buildmesh**.

Mesh



Formulazione variazione

```
// The finite element space using P1 elements
fespace Vh(Th,P1);

// Force term
func f = 1.0;

// Defines u and v as piecewise-P1 continuous function
Vh phi, w;

//  Laplace VF
problem Laplace(phi,w,solver=UMFPACK) =
    int2d(Th)(dx(phi)*dx(w)+dy(phi)*dy(w))
    - int2d(Th)(f*w)
    + on(Gamma1,phi=0); // Dirichlet bc
```

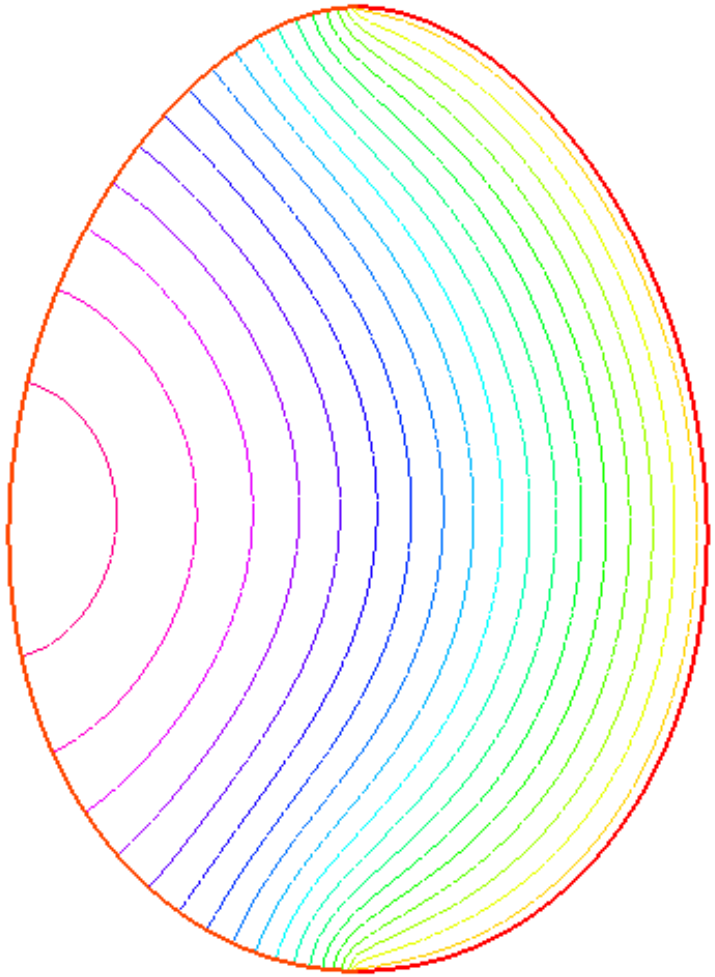
È possibile definire funzioni, spazi funzionale e problemi nella formulazione variazionale.

Risoluzione e visualizzazione dei risultati

```
// Solve the PDE
Laplace;

// Plot solution
plot(phi,wait=1,ps=savedir + problemPrefix + "_sol.eps");
```

Richiamo il problema da risolvere e poi visualizzo i risultati.

Risoluzione e visualizzazione dei risultati



Il problema

Risolvere il seguente problema di Poisson

$$\begin{cases} -\Delta u = f & \text{in} \quad \Omega \\ u(x,y) = e^{x+y} & \text{su} \quad \Gamma_D \\ \partial_n u(x,y) = u_n & \text{su} \quad \Gamma_N \end{cases}$$

dove

- la soluzione esatta del problema sia $u(x,y) = e^{x+y}$;
- Ω è il quadrato unitario $\Omega = (0,1) \times (0,1)$;
- Γ_N è il bordo formato dal lato sinistro e destro del quadrato;

Formulazione variazionale

La forma debole associata al seguente problema generale

$$
\begin{aligned}
-\frac{\partial}{\partial x_i}\left(k\frac{\partial \phi}{\partial x_i}\right) &= f && \text{in } \Omega \\
\phi &= \phi_d && \text{su } \Gamma_D \\
k\frac{\partial \phi}{\partial x_i} n_i &= \phi_n && \text{su } \Gamma_N
\end{aligned}
$$

è

$$
\int_\Omega \frac{\partial v}{\partial x_i} k \frac{\partial \phi}{\partial x_i}\, d\omega - \int_{\Gamma_N} v\phi_N\, d\gamma - \int_\Omega vf\, d\omega = 0\,.
$$

Approssimazione con FreeFem++

```
//  Poisson equation over the unit square with Neumann b.c., M. Venturin

//  Include macro and define problem prefix
include "error_L2.edp"                    //  Macro for L2 error
include "error_H1.edp"                    //  Macro for H1 error
string savedir = "results_poisson/";      // Save directory
string problemPrefix = "poisson";         // Problem prefix used in output files

//  Exact solution and its derivatives
func ue     = exp(x+y);                   // Exact solution
func dxue   = exp(x+y);                   // Derivative w.r.t. x
func dyue   = exp(x+y);                   // Derivative w.r.t. y

//  PDE Parameters for the Poisson equation
func k      = 1.0;                        // Diffusion term
func f      = -2.0*exp(x+y);              // Force term
func gd     = exp(x+y);                   // Dirichlet boundary condition
func un2    = exp(x+y);                   // Derivative right domain
func un4    = -exp(x+y);                  // Dirichlet left domain

//  Problem parameters and error variables
int ndiv    = 16;                         // Number of initial subdivisions
int i       = 0;                          // Loop variable
int maxiter = 5;                          // Number of maximum subdivisions
real[int] errNT(maxiter);                 // Number of elements
real[int] errL2(maxiter);                 // Error in L2 norm
real[int] errH1(maxiter);                 // Error in H1 norm
real[int] maxh(maxiter);                  // Max element size
real[int] cputime(maxiter);               // CPU time information for each iteration
```

Approssimazione con FreeFem++

```
// Main problem over mesh refinement
for(i=0;i<maxiter;i++){
    cout << "Performin iteration #" << i+1 << endl;

    mesh Th = square(ndiv,ndiv); // Uniform mesh over the unit square [0,1] x [0,1]
    plot(Th,wait=1,ps=savedir + problemPrefix + "_mesh_" + i + ".eps");

    fespace Vh(Th,P1);  // Piecewise-P1 continuous function for u and v
    Vh u, v;

    // Poisson equation: - d/dxi(k*dphi/dxi) = f
    problem Poisson(u,v,solver=UMFPACK) =
        int2d(Th)(dx(v)*k*dx(u)+dy(v)*k*dy(u)) - int2d(Th)(v*f) // Diffusion and source
        - int1d(Th,4)(v*un4) - int1d(Th,2)(v*un2)               // Diffusion line integrals
        + on(1,3,u=gd);                                         // Dirichlet bc

    // Solve the PDE saving its computational time
    cputime[i] = clock(); Poisson; cputime[i] = clock()-cputime[i];
    plot(u,wait=1,ps=savedir + problemPrefix + "_sol_" + i + ".eps");

    // Saving solution data for output
    errNT[i] = Th.nt;                                           // Number of triangles
    errorL2(Th,u,ue,errL2[i]);                                  // Compute error L2
    errorH1(Th,u,ue,dxue,dyue,errH1[i]);                        // Compute error H1
    fespace Ph(Th,P0); Ph h = hTriangle; maxh[i] = h[].max; // Element size

    ndiv = ndiv*2;  // Set the number of subdivisions
    cout << endl << endl << endl;       // End lines
}
```

Approssimazione con FreeFem++

```
//  Save data to an external file
{
    ofstream outfile(savedir + problemPrefix + "_result.dat");
    for(i=0;i<maxiter;i++){
        outfile << i+1 << " " << errNT[i] << " " << maxh[i];
        outfile << " " << errL2[i] << " " << errH1[i];
        outfile << " " << cputime[i] << endl;
    }
}
```

Approssimazione con FreeFem++

```
//  ********************************************************************************
//  FILE        : errorL2.edp
//  DESCRIPTION : Error in the L2 norm
//  AUTHOR      : M. Venturin
//  ********************************************************************************

//  - Th    (input)     : Mesh
//  - u     (input)     : Approximated solution
//  - ue    (input)     : Exact solution
//  - err   (output)    : Error

macro errorL2(Th,u,ue,err)(
    err = sqrt(int2d(Th)((u-ue)^2))
)   // end macro
```

L'errore in norma L^2 è definito come

$$||e||_{L^2}^2 = \int_\Omega ||u_h - u||^2 \, d\Omega$$

dove u e u_h sono rispettivamente la soluzione di riferimento e quella approssimata.

Approssimazione con FreeFem++

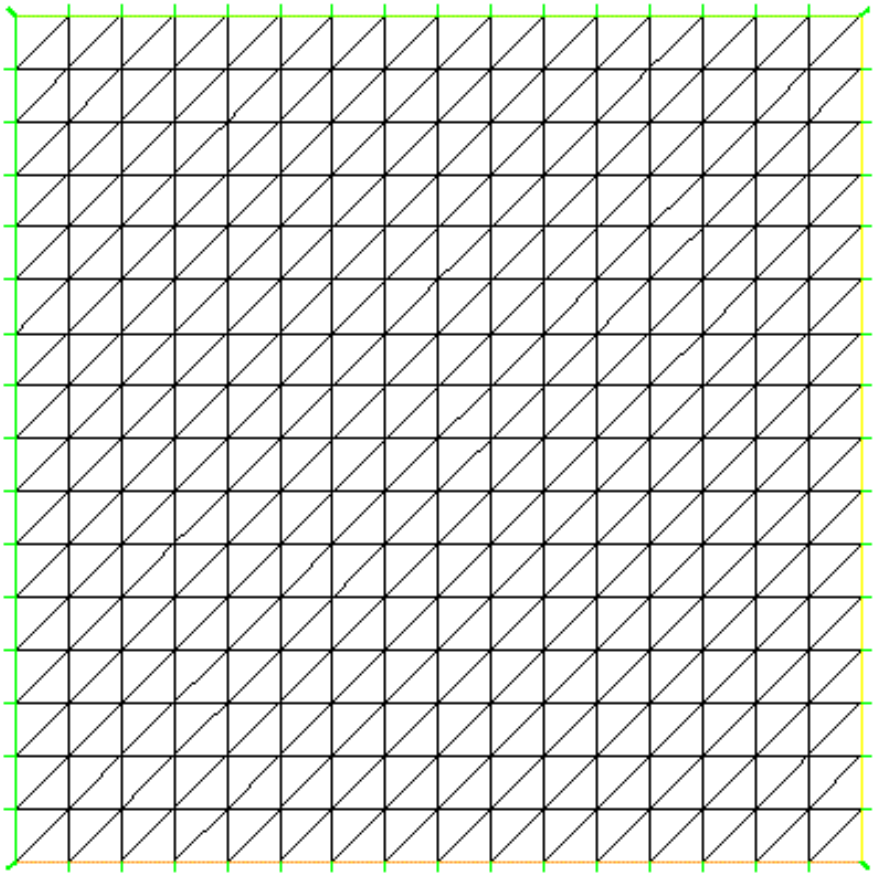
```
//  ********************************************************************************
//  FILE        : errorH1.edp
//  DESCRIPTION : Error in the H1 norm
//  AUTHOR      : M. Venturin
//  ********************************************************************************

//  - Th    (input)     : Mesh
//  - u     (input)     : Approximated solution
//  - ue    (input)     : Exact solution
//  - dxue  (input)     : Derivative of the exact solution along x
//  - dyue  (input)     : Derivative of the exact solution along y
//  - err   (output)    : Error

macro errorH1(Th,u,ue,dxue,dyue,err)(
    err = sqrt(int2d(Th)((u-ue)^2) +
    int2d(Th)((dx(u)-dxue)^2)+ int2d(Th)((dy(u)-dyue)^2))
)   // end macro
```

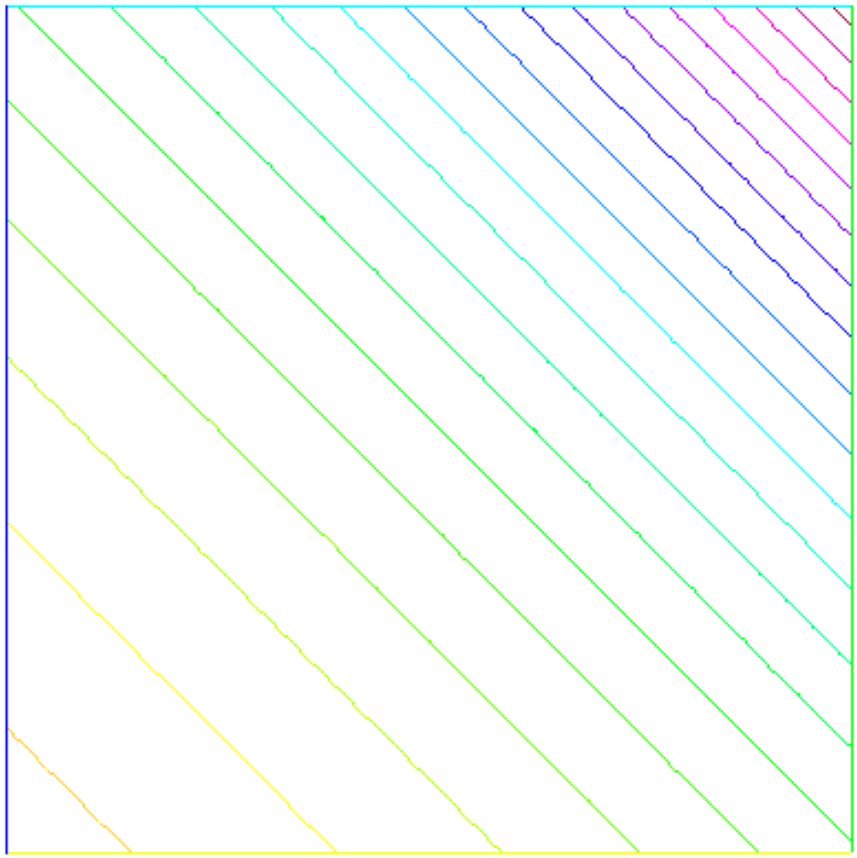
L'errore in norma H^1 è definito come

$$||e||_{H^1}^2 = \int_\Omega ||u_h - u||^2 \, d\Omega + \int_\Omega \left|\left| \frac{\partial u_h}{\partial x_i} - \frac{\partial u}{\partial x_i} \right|\right|^2 d\Omega$$

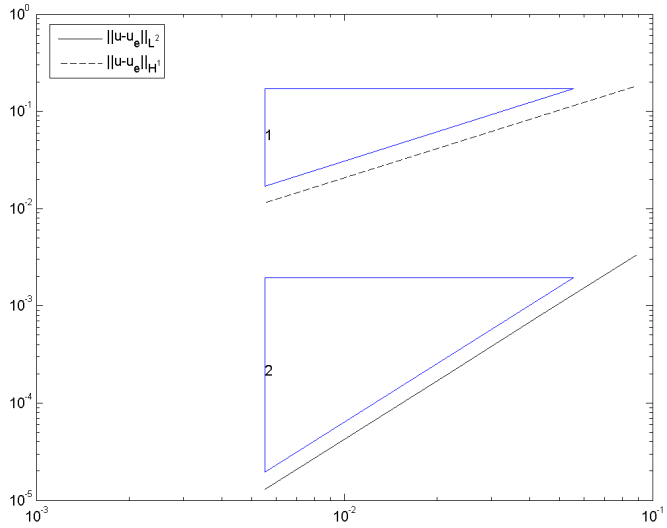
dove u e u_h sono rispettivamente la soluzione di riferimento e quella approssimata.

Analisi dei risultati





Analisi dei risultati



Il risultato rispecchia quanto attesso dalla teoria nel caso di elementi finiti lineari. Dato u la soluzione di riferimento e u_h la sua approssimazione di ordine k, si ha

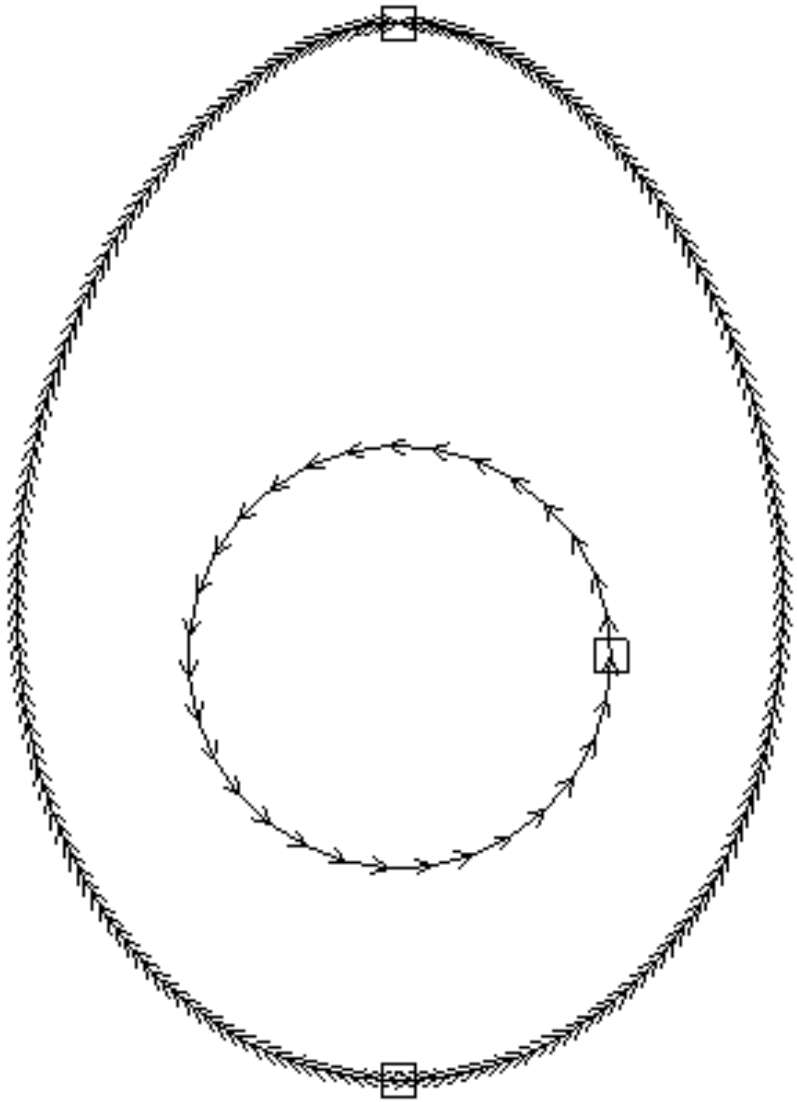
- $||u - u_h||_{H^1(\Omega)} \leq Ch^s |u|_{H^{s+1}(\Omega)}$
- $||u - u_h||_{L^2(\Omega)} \leq Ch^{s+1} |u|_{H^{s+1}(\Omega)}$

con $s = \min\{k, p\}$ dove $u \in H^{p+1}$.

Parametri globali e geometria

```
//  ********************************************************************************
//  FILE        : ex_geo.edp
//  DESCRIPTION : Example of geometry
//  AUTHOR      : M. Venturin
//  ********************************************************************************

//  Problem prefix
//  ______________
string savedir = "results_ex_geo/"; // Save directory
string problemPrefix = "ex_geo";    // Problem prefix used in output files

// Defining the boundary as an egg shape
real eggheight = 1;
real a = eggheight/2;
border Gamma1(t=0,pi){x=0.78*a*cos(t/4)*sin(t); y=-a*(cos(t)-1); label=1;}
border Gamma2(t=0,pi){x=-0.78*a*cos(t/4)*sin(t); y=-a*(cos(t)-1); label=2;}
real c = eggheight/2.5;
real r = eggheight/5;
border Inside(t=0,2*pi){ x=r*cos(t); y=c+r*sin(t);label=3;}

// Geometry
int nbdiv1 = 101;    // Number of subdivisions (external)
int nbdiv2 =  31;    // Number of subdivisions (internal)
func geom = Gamma1(nbdiv1) + Gamma2(-nbdiv1) + Inside(nbdiv2);  // with inside
// func geom = Gamma1(nbdiv1) + Gamma2(-nbdiv1) + Inside(-nbdiv2); // without inside
plot(geom,wait=true,bw=1,ps=savedir + problemPrefix + "_geom.eps");
```

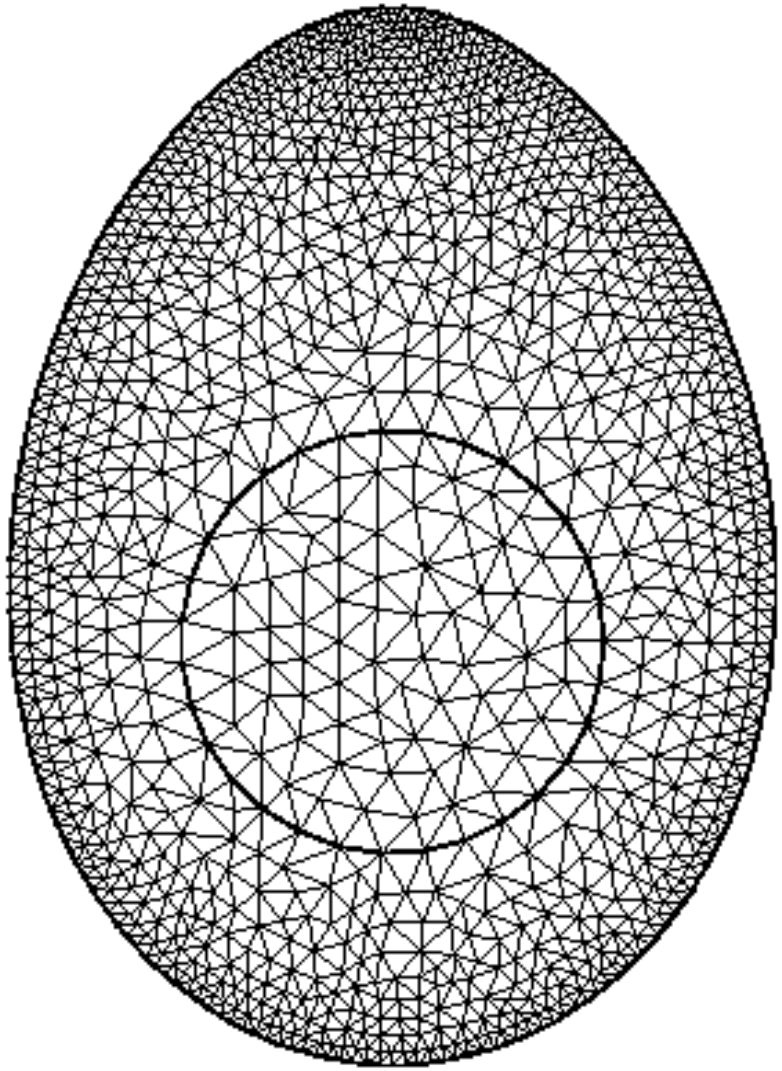
Geometria



Mesh e dati del problema

```
// Mesh
mesh Th = buildmesh(geom);
plot(Th,wait=true,bw=1,ps=savedir + problemPrefix + "_mesh.eps");

// Saving mesh (Borders name are lost only refs are kept)
// savemesh(Th,savedir + problemPrefix + "_meshdata.msh");
int regext = Th(0,0.001).region;
int regint = Th(0,c).region;

// The finite element space using P1 elements
fespace Vh(Th,P1);

// Force term
func f = 0.1 ;
func k = 0. + .5*(region==regext) + 0.1*(region==regint);

// Defines u and v as piecewise-P1 continuous function
Vh phi, w;
```

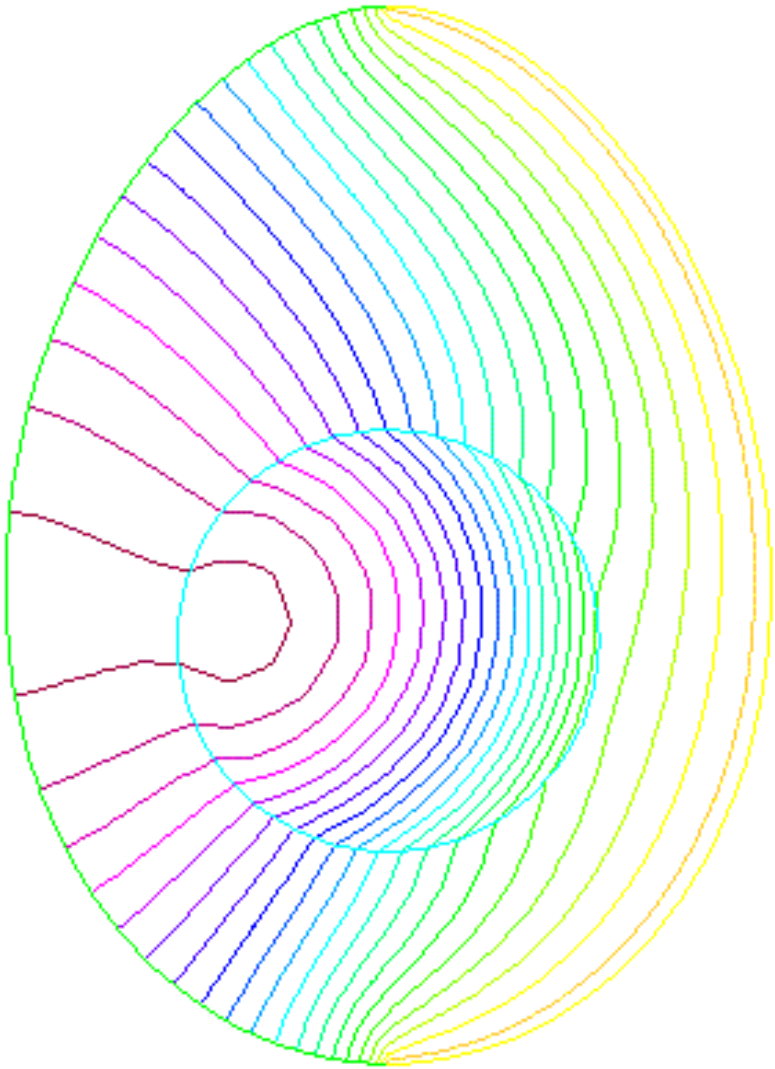
Mesh



Soluzione

```
//  Laplace VF
problem Laplace(phi,w,solver=UMFPACK) =
    int2d(Th)(dx(phi)*k*dx(w)+dy(phi)*k*dy(w))
    - int2d(Th)(f*w)
    + on(Gamma1, phi=30);    // Dirichlet bc

// Solve the PDE
Laplace;

// Plot solution
plot(phi,wait=1,ps=savedir + problemPrefix + "_sol.eps");
```

Visualizzazione dei risultati



Conclusione

Grazie per l'attenzione