Overview

- Finite element method overview
- FreeFEM++
- FreeFEM++ Helmholtz problem

- Consider the partial differential equation $\mathcal{L}u = f$ in domain Ω, $u \in \mathcal{C}^2(\Omega)$. Examples:
  - Laplace equation $\mathcal{L} = -\nabla^2$, $f = 0$
  - Poisson equation $\mathcal{L} = -\nabla^2$, $f \neq 0$
  - Helmholtz equation $\mathcal{L} = \nabla^2 + k^2$, $f = 0$
- Discretize Ω into finite elements, typically simplicia T_i (i.e., triangles in 2D, tetrahedra in 3D)

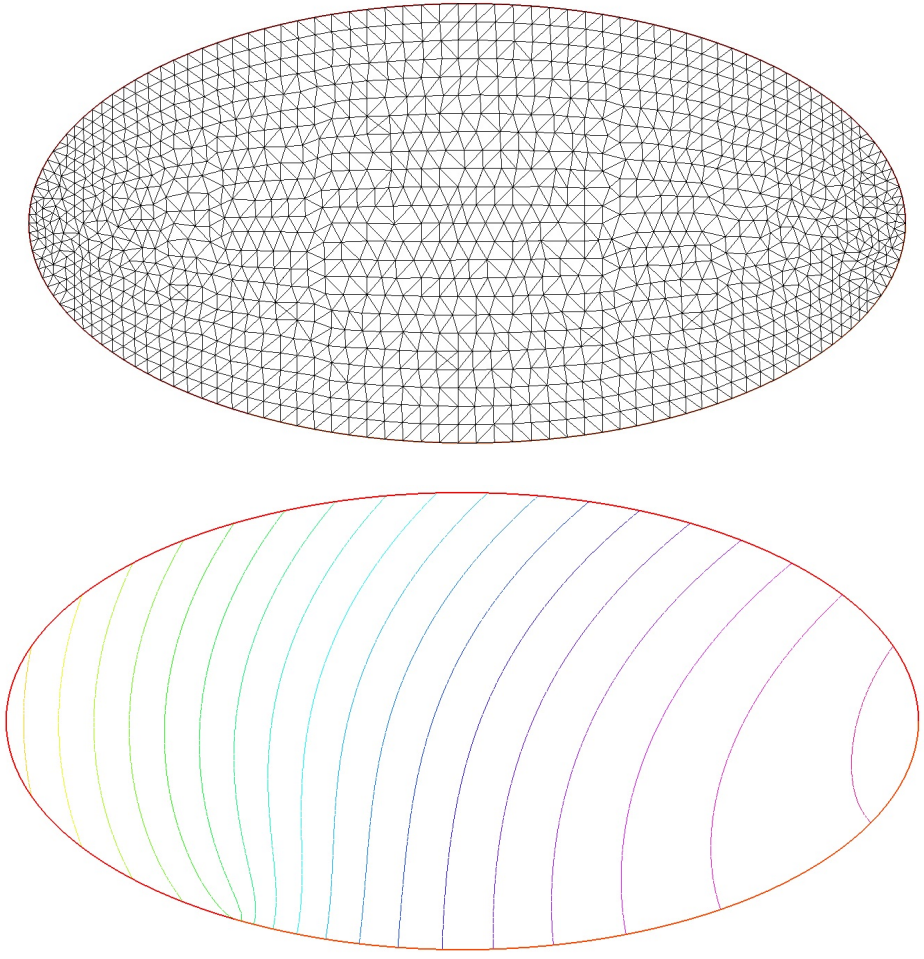
$$\Omega = \cup_{i=1}^{E} T_i$$

- Introduce a piecewise approximation $\tilde{u}_{T_i}(x) = \sum_{j=1}^{n} N_j(x) U_j^{T_i}$, where $N_j(x)$ are called form functions, and $U_j^{T_i}$ are nodal values within element T_i. The overall approximation $\tilde{u}(x)$ whose restriction on each element is $\tilde{u}(x \in T_i) = \tilde{u}_{T_i}(x)$, is an element of an appropriate Sobolev space $W^{k,p}$ (intuitively a characterization of smoothness of the function and its derivatives)
- Define a scalar product (a bilinear form), $(,): W^{k,p} \times W^{k,p} \to \mathbb{R}$, and introduce a finite set of test functions $(v_1, ..., v_M)$, typically $M = nE$, and $v_l = N_1^{T_l}, ..., v_{l+n-1} = N_n^{T_l}$ (thus defining a Galerkin method)
- Find nodal values by solving equations arising from projection $(\mathcal{L}\tilde{u}, v_k) = (f, v_k)$, $k = 1, ..., M$
- Typical problem reformulation $\mathcal{L} = -\nabla^2$, $(\mathcal{L}\tilde{u}, v_k) = b + (\nabla\tilde{u}, \nabla v_k)$, b: boundary conditions

- FreeFEM++ (see freefem.org) is a product of the Laboratoire Jacques Louis Lions (LJLL), Sorbonne University, Paris, that provides a high-level language for finite element formulations, very close to mathematical formulations obtained in functional analysis
- Example: Poisson equation in an ellipse, $\nabla^2 u = f(x) = 1$

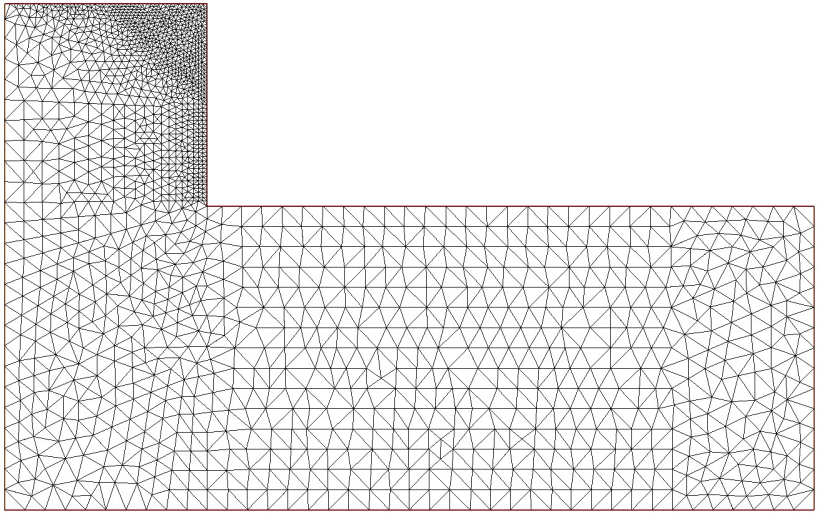
```
real theta=4.*pi/3.;
real a=2.,b=1.; // the length of the semimajor axis and semiminor axis
func z=x;
border Gamma1(t=0,theta) { x = a * cos(t); y = b*sin(t); }
border Gamma2(t=theta,2*pi) { x = a * cos(t); y = b*sin(t); }
mesh Th=buildmesh(Gamma1(100)+Gamma2(50)); // construction of mesh

plot(Th,wait=true, ps="membraneTh.eps");

fespace Vh(Th,P2); // P2 conforming triangular FEM
Vh phi,w, f=1;

solve Laplace(phi,w) =
  int2d(Th)(dx(phi)*dx(w) + dy(phi)*dy(w)) -
  int2d(Th)(f*w) +
  on(Gamma1,phi=z); // Solve Poisson equation
plot(phi,wait=true, ps="membrane.eps");
```

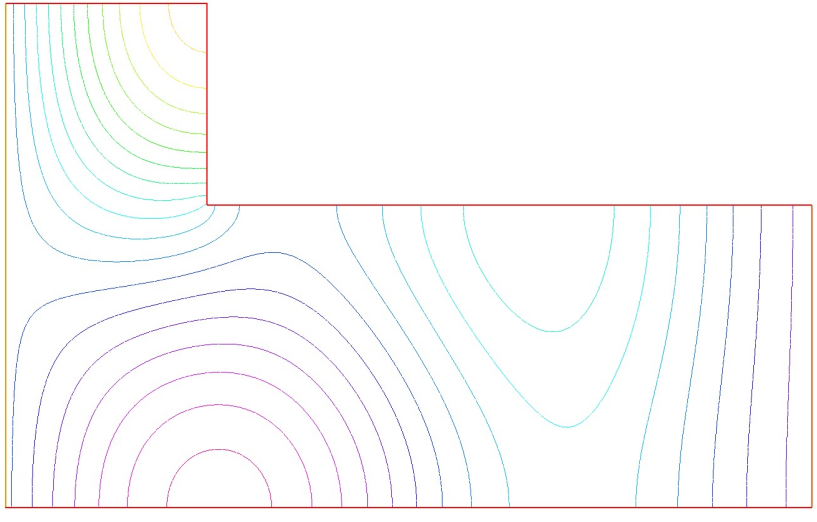
FreeFEM++ Poisson equation example



```
real aL=8., bL=3., cL=2, dL=2., aE = 0.49*aL, bE = 0.49*bL; // geometry
// Define region border
border B1(t=0,aL) { x = t; y=0; label=1; }
border B2(t=0,bL) { x = aL; y=t; label=2; }
border B3(t=0,aL-cL) { x = aL-t; y=bL; label=1; }
border B4(t=0,dL) { x = cL; y=bL+t; label=1; }
border B5(t=0,cL) { x = cL-t; y=bL+dL; label=1; }
border B6(t=0,bL+dL) { x = 0; y=bL+dL-t; label=3; }

int n=5; // Choose base boundary discretization
// Generate mesh of region interior mesh
Th=buildmesh(B1(8*n)+B2(3*n)+B3(6*n)+B4(10*n)+B5(10*n)+B6(5*n));
plot(Th,wait=true, ps="Lshape.eps");

real k2=1.; // Wavenumber
Vh(Th,P2); // Define finite element space, test functions fespace
Vh u,v;
// Weak form of Helmholtz operator solve
Helmholtz(u,v) = int2d(Th)(dx(u)*dx(v) + dy(u)*dy(v) - k2*u*v) +
                 on(2,u=1) + on(3,u=0) ;

plot(u, wait=true, ps="Lmode.eps"); // Display eigenmode
```

L-shaped membrane eigenmode





```
// Build Helmholtz problem discretization
varf vHelmholtz(u,v) = int2d(Th)(dx(u)*dx(v) + dy(u)*dy(v) - k2*u*v) +
       on(2,u=2) + on(3,u=0);

matrix A = vHelmholtz(Vh,Vh);
real[int] b = vHelmholtz(0,Vh);

// Save the discretization to a file
{ofstream fout("A.txt"); fout << A << endl;}
{ofstream fout("b.txt"); fout << b << endl;}
```