```
// example 12 : poisson-LinearCG.edp   [slide page 59]
// finite element solution of Poisson equation with mixed boundary condition
// in matrix form and solved by LinearCG with diagonal preconditioner
// for tutorial by Japan SIAM, Tokyo, 11-12 Feb.2016, Atsushi Suzuki

int n = 20;
mesh Th=square(n,n);
fespace Vh(Th,P1);

Vh u,v;
real err, hh;

func f = 5.0/4.0 * pi * pi * sin(pi * x) * sin(pi * y / 2.0);
func h = (-pi)/2.0 * sin(pi * x);
func g = sin(pi * x) * sin(pi * y / 2.0);
// for error estimation
func sol = sin(pi * x) * sin(pi * y / 2.0);
func solx = pi * cos(pi * x) * sin(pi * y / 2.0);
func soly = (pi / 2.0) * sin(pi * x) * cos(pi * y / 2.0);

varf aa(u,v) = int2d(Th)( dx(u)*dx(v)+dy(u)*dy(v) )
               + on(2,3,4,u=1.0);
varf external(u,v) = int2d(Th)( f*v ) + int1d(Th,1) (h * v);

real tgv=1.0e+30;
matrix A;
real[int] bc = aa(0, Vh, tgv=tgv);

func real[int] opA(real[int] &pp)
{
  pp = bc ? 0.0 : pp;     // SpMV operation only for node in interior of
  real[int] qq = A*pp;    // the domain without Dirichlet nodes
  pp = bc ? 0.0 : qq;
  return pp;
}

func real[int] opQ(real[int] &pp)
{
  for (int i = 0; i < pp.n; i++) {
    pp(i) = pp(i) / A(i, i);
  }
  pp = bc ? 0.0 : pp;
  return pp;
}

A = aa(Vh, Vh, tgv=tgv, solver=sparsesolver);
real[int] ff = external(0, Vh);
v = g;                // g is valid only on the boundary 1
u[] = bc ? v[] : 0.0; // u[] has Dirichlet data without tgv
v[] = A * u[];        // v[] = A_{12}*g
v[] = bc ? 0.0 : v[];
ff -= v[];            // ff_{1} -= A_{12}*u_{2}
ff = bc ? 0.0 : ff;
LinearCG(opA, u[], ff, precon=opQ, nbiter=200, eps=1.0e-10,verbosity=50);

hh = 1.0 / real(n) * sqrt(2.0);
// int2d uses qf5pT : 5th order integration quadrature
err = int2d(Th)( (dx(u) - solx) * (dx(u) - solx) +
                 (dy(u) - soly) * (dy(u) - soly) +
                 (u - sol) * (u - sol));
err = sqrt(err);

cout << "DOF=" << u[].n << "\t h=" << hh << " err-H1=" << err << endl;
```