

```

// example 13b : poisson-Lagrange-mult.edp
// finite element solution of Poisson equation with full-Neumann condition
// and solved by Lagrange multiplier using 2x2 block matrix
// for tutorial by Japan SIAM, Tokyo, 11-12 Feb.2016, Atsushi Suzuki

int n;
n = 40;
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 h1 = (-pi)/2.0 * sin(pi * x) * cos(pi * y / 2.0);
func h2 = pi * cos(pi * x) * sin(pi * y / 2.0);
func h3 = pi/2.0 * sin(pi * x) * cos(pi * y / 2.0);
func h4 = (-pi) * cos(pi * x) * sin(pi * y / 2.0);
// for error estimation
func sol = sin(pi * x) * sin(pi * y / 2.0) - 4.0 / (pi * pi); // int sol = 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) );
varf external(u,v) = int2d(Th)( f*v )
+ intl1d(Th,1)(h1*v) + intl1d(Th,2)(h2*v)
+ intl1d(Th,3)(h3*v) + intl1d(Th,4)(h4*v);

matrix A = aa(Vh, Vh, solver=sparsesolver);
real[int] c(u[.].n);
c = 1.0; // kernel of A
matrix AA=[[A, c], [c', 0]]; // matrix with constraint (c, u)=0
set(AA, solver=UMFPACK);

real[int] ff = external(0, Vh);
ff -= ff.sum / ff.n; // forcing RHS in image of A

real[int] b(u[.].n + 1), x(u[.].n + 1);
b = [ff, 0.0];
x = AA^-1 * b;
real xx;
[u[.], xx] = x;

real meanu = int2d(Th)(u) / Th.area; // mean value of u on FEM mesh
u[.] -= meanu;

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;

```