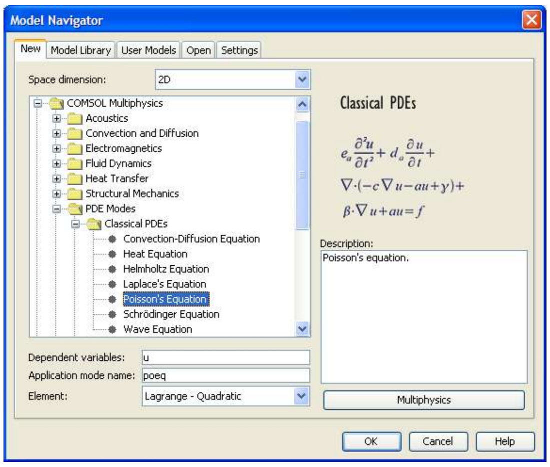
Numerical Solution of PDE: Comsol Multiphysics

This is an introduction to get you acquainted with Comsol Multiphysics for numerical solutuon of PDE by finite elements. The program has many facilities and we will use only a small fraction. The documentation is available (PDF and HTML) under the help button, **Help**, rightmost on the menu bar (top). The simplest is *Quick Start*, but even that is quite comprehensive. The GUI-functions are hopefully intuitive.

Example 1, The Poisson equation on an ellipse
Start comsol. In the first screen **`ModelNavigator`** we choose 2D och



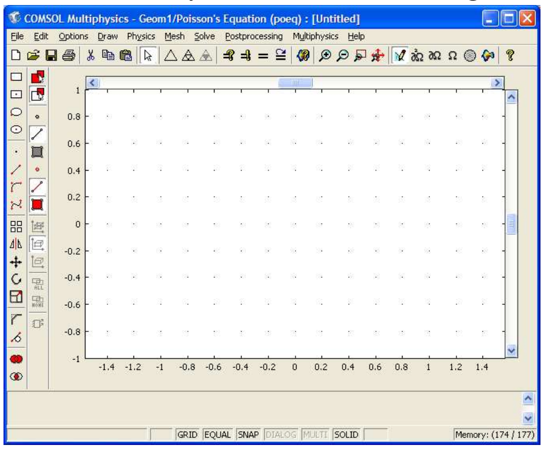
Space dimension: 2D

Dependent variables: u
Application mode name: poeq
Element: Lagrange - Quadratic

Description:
Poisson's equation.

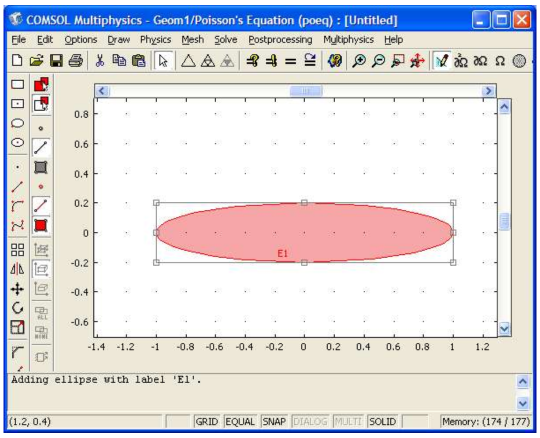
`ApplicationModes>ComsolMultiphysics>PDEModes>ClassicalPDEs>Poisson's Equation.`
We see that the element type is **`Lagrange P2`**-triangles. The solution is approximated with second degree polynomials, one over each triangle, a globally C^0 function. The elements can be changed later, if you wish. But the name of the solution (here **`u`**) must be changed here.

The next screen shows the work plane (2D …) and one usually works from left to right in the menu bar:


COMSOL Multiphysics - Geom1/Poisson's Equation (poeq) : [Untitled]
File Edit Options Draw Physics Mesh Solve Postprocessing Multiphysics Help
GRID EQUAL SNAP DIALOG MULTI SOLID
Memory: (174 / 177)

1. Create geometry (**`Draw`**)
2. Define the PDE (**`Physics`**)
3. Set boundary conditions (**`Physics`**)
4. Make the mesh (**`Mesh`**)
5. Solve (**`Solve`**)
6. Plot (**`Postprocessing`**)

Under **`File`** we find operations like **`open, save, save as`**, etc. **`Options`** gives facilities like defining constants and expressions. It is practical to have names for important model parameters like heat transfer coefficients, specific heat, etc.


File Edit Options Draw Physics Mesh Solve Postprocessing Multiphysics Help
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-1.4 -1.2 -1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1 1.2
E1
GRID EQUAL SNAP DIALOG MULTI SOLID

COMSOL Multiphysics - Geom1/Poisson's Equation (poeq) : [Untitled]

```
Adding ellipse with label 'E1'.
```

(1.2, 0.4) Memory: (174 / 177)

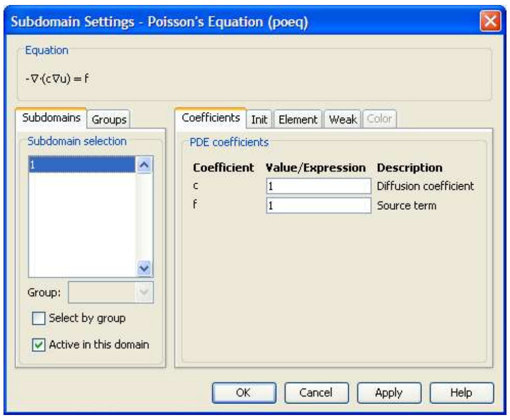
Geometry

The geometry is built in a drawing program, from elementary bodies (2D Rectangles and ellipses), or surfaces bounded by Bezier-curves. The bodies are put together by set operations (union, intersection, difference). We make an ellipse, centered at the origin and half axes 1 and 0.2. Note that the objects are snapped to the grid, change the grid under `Options>GridSettings` if needed. Our domain is only the single ellipse so we move on to

Define PDE

`Physics>SubdomainSettings` (there is `boundary` too, in a minute…)
The PDE is in the Equation pane,

and we need to define the coefficient *c* and the source term *f*. One can give functions of *x,y, u, ux, uy* and, if time-dependent, *t*. There is only one subdomain, the ellipse, so click it and accept a *c* = *f* = 1 by OK.



Subdomain Settings - Poisson's Equation (poeq)

Equation

$-\nabla\cdot(c\nabla u) = f$

Subdomain selection: 1

Group:

☐ Select by group

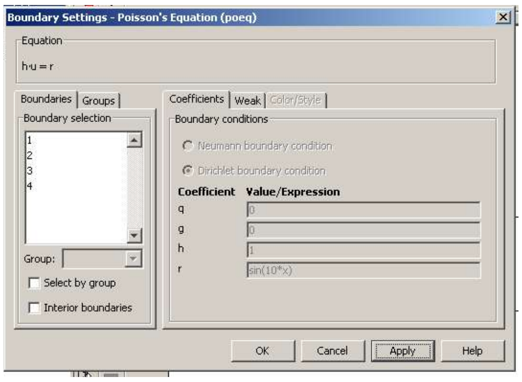
☑ Active in this domain

PDE coefficients

Coefficient	Value/Expression	Description
c	1	Diffusion coefficient
f	1	Source term

`Physics>BoundarySettings`

Each ellipse gives four boundaries. Choose them all by **`ctrl-a`**; They can be selected also by clicking (shift-clicking, ctrl-clicking) on the work pane. The default boundary condition is apparently Dirichlet: *h u* = *r* with *h* = 1 and *r* = 0 and we change to *r* = sin(10*x*) for some variation, OK.



Boundary Settings - Poisson's Equation (poeq)

Equation

$h\cdot u = r$

Boundary selection: 1 2 3 4

Group:

☐ Select by group

☐ Interior boundaries

Boundary conditions

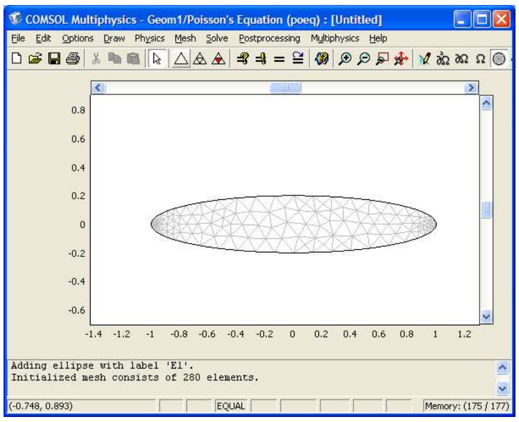
○ Neumann boundary condition

● Dirichlet boundary condition

Coefficient	Value/Expression
q	0
g	0
h	1
r	sin(10*x)

Element mesh

The triangle icon gives a mesh. It looks good, OK. The mesh can be controlled in some detail by parameters under **`Mesh>MeshParameters`** .



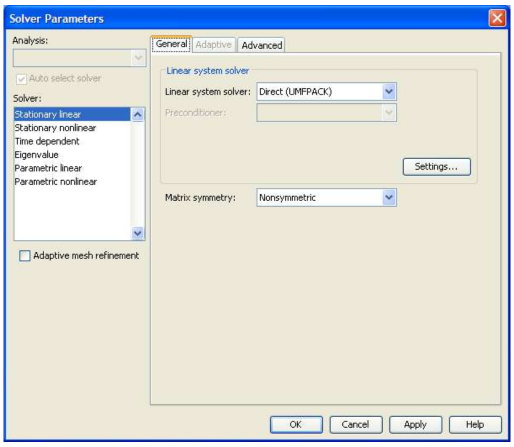
COMSOL Multiphysics - Geom1/Poisson's Equation (poeq) : [Untitled]

```
Adding ellipse with label 'E1'.
Initialized mesh consists of 280 elements.
```

(-0.748, 0.893) Memory: (175 / 177)

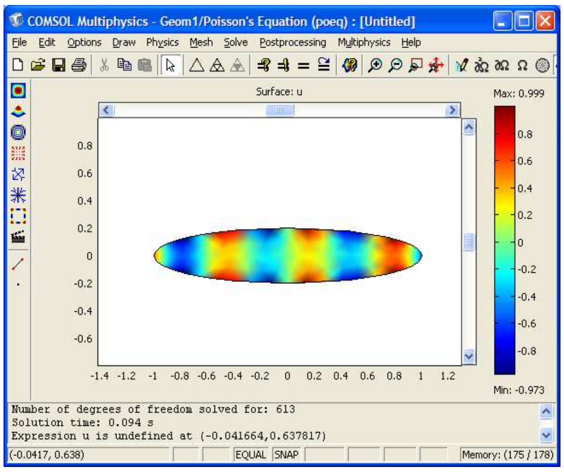
Solution

The generated equations are solved when you click the = icon. Scroll down to **`Solve>SolverParameters`** and check the possibilities offered, Our choice of *c*, *f* and boundary conditions makes the problem linear so the default selection **`stationary linear`** is correct, OK. Details that can be very important for large (and non-linear) problems, like choice of solver for linear systems, are controllable here. It says **`Unsymmetric`** about the matrix – whereas it is in fact symmetric – but never mind, OK, and solve by =.


Solver: Stationary linear
Linear system solver: Direct (UMFPACK)
Matrix symmetry: Nonsymmetric

After a fraction of a second, the system with 613 unknowns has been constructed and solved and painted as colored iso-surfaces on the work pane, and the log-line (bottom of screen) says

```
Number of degrees of freedom
solved for: 613
Solution time: 0.093 s
```


```
Number of degrees of freedom solved for: 613
Solution time: 0.094 s
Expression u is undefined at (-0.041664,0.637817)
```
COMSOL Multiphysics - Geom1/Poisson's Equation (poeq) : [Untitled]
File Edit Options Draw Physics Mesh Solve Postprocessing Multiphysics Help
Surface: u
Max: 0.999
Min: -0.973
(-0.0417, 0.638)
EQUAL
SNAP
Memory: (175 / 178)

Convergence study

To see how the solution converges with refinement of the mesh, we compute *u* in the center. Choose
`Postprocessing>DataDisplay>Subdomain`
where we give the coordinates for the center – (0,0) and see

```
Value: 0.020201, Expression: u, Position: (0,0)
```
in the log-line.
Make a regular refinement (each triangle split into four) by the four triangle icon, solve again, and print u(center) again:

```
       Number of degrees of freedom solved for: 2345
       Solution time: 0.188 s
       Value: 0.019069, Expression: u, Position: (0,0)
```
and again, and again …, ... try up to, say, a million degrees of freedom when the solver runs out of memory.
Convergence:

u(0,0)	**diff.**
1201	
69	-1132
231	162
234	3
231	-3

The order of convergence should be 2 for regular solutions in energy norm and 3 in max-norm. The exponent is hard to see from the data, but the convergence is rapid. The

convergence is *not* as regular as you have seen for e,g, the trapezoidal rule for quadrature, because the mesh is so irregular.
Now try piecewise linear basis functions instead. **`Lagrange P1`** elements are chosen under **`Physics>SubdomainSettings>Element`**

Repeat the experiment: Slower convergence.

```
#dof      time     u(0.0)    diff.
   613   0.094   0.019103
  2345   0.156   0.019115  12
  9169   0.547   0.019205  90
 36257   2.172   0.019224  19
144193  11.204   0.019230   6
```