
FEATool

Version: 1.8

Quickstart Guide

Precise Simulation Ltd.
2003, 20F, Tower 5
China Hong Kong City
33 Canton Road
Tsim Sha Tsui, Kowloon
Hong Kong

Home page and email

www.featool.com

info@featool.com

(c) Copyright 2013-2018 by Precise Simulation Limited. All rights reserved

The software described in this document is furnished under a license agreement. The software may be used, modified, and copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from Precise Simulation Ltd.

FEAToolTM and FEAToolTM Multiphysics are trademarks of Precise Simulation Limited.

MATLAB is a registered trademark of The MathWorks, Inc., and OpenFOAM is a registered trademark of ESI Group, other product or brand names are registered trademarks of their respective holders.



Contents

Contents	iii
1 Quickstart Guide	2
1.1 Prerequisites	2
1.2 Installation	3
1.2.1 FEATool with Matlab (Windows)	3
1.2.2 FEATool with Matlab (Linux/Mac)	3
1.2.3 FEATool with Octave (All systems)	4
1.2.4 Installation Script	4
1.2.5 Verification	5
1.2.6 External Solvers	7
1.2.7 External Grid Generators	7
1.3 Modeling Process	8
1.4 Working with FEATool	9
1.4.1 Graphical User Interface (GUI)	9
1.4.2 Pre-defined physics modes	11
1.4.3 Weak FEM equation formulation	13
1.4.4 Direct matrix assembly	14
1.5 Structural Mechanics Example - Thin Plate with Hole	15
1.5.1 Thin Plate with Hole using the GUI	16
1.5.2 Thin Plate with Hole using the CLI	37
1.6 Multiphysics Example - Heat Exchanger	38

1.6.1	Heat Exchanger using the GUI	39
1.6.2	Heat Exchanger using the CLI	55
1.7	Equation Editing Example - Axisymmetric Fluid Flow	56
1.7.1	Axisymmetric Fluid Flow using the GUI	58
1.7.2	Axisymmetric Fluid Flow using the CLI	67
1.8	Classic Equation Example - Poisson Equation with a Point Source	68
1.8.1	Poisson Equation with a Point Source using the GUI	68
1.8.2	Poisson Equation with a Point Source using the CLI	81
1.9	Custom Equation Example - Wave Equation on a Circle	82
1.9.1	Wave Equation using the GUI	82
1.9.2	Wave Equation using the CLI	93
1.10	More Examples and Information	94



FEATool

Welcome to FEAToolTM (short for Finite Element Analysis Toolbox), a [GNU Octave](#) and Matlab[®] toolbox for modeling and simulation of physics, partial differential equations (PDE), and mathematical problems with the finite element method (FEM).

FEATool aims to provide an easy to use and comprehensive all-in-one integrated simulation package for all types of finite element and multiphysics analyses, and combine the best of ease of use, powerful functionality, and extensibility for both beginners and advanced users. Features such as an intuitive and easy to learn graphical user interface (GUI), a complete library of grid generation, FEM, and postprocessing functions, as well as command line interface (CLI) programming, and interactive and interpreted programming and scripting capabilities, makes FEATool suitable for everyone from students learning mathematical modeling, to professionals and engineers wishing to explore new ideas in a simple, fast, and convenient way.

Please enjoy working with FEATool, and do not hesitate to get in contact if you have any questions or suggestions for improvements.

1

Quickstart Guide

The quickstart guide explains how to install and start FEATool Multiphysics as well as describes the modeling and simulation process. Furthermore, step by step instructions how to set up and solve five modeling examples illustrating different techniques and features of FEATool is also included.

1. Introductory example of modeling stresses and strains in a thin plate with a hole.
2. Multiphysics example coupling heat and fluid flow in a heat exchanger.
3. Equation editing example modeling axisymmetric fluid flow in a narrowing pipe.
4. Classic equation example for the Poisson equation on a circle with a point source.
5. Custom equation example for solving the wave equation on a circle.

1.1 Prerequisites

The FEATool Multiphysics simulation toolbox is written in the m-script language, which requires either GNU Octave or Matlab to run and interpret the source code. Octave is freely available from the [Octave homepage](#) Matlab is a commercial alternative to Octave and can be licensed

directly from [The Mathworks Inc.](#)

FEATool has been verified to run on Octave version 3.8 or later (Octave version 4.0 or later is required for GUI compatibility on Windows and Linux systems, while MAC systems do not support the FEATool Octave GUI at all). For Matlab, FEATool should run on Matlab versions 7.9 (R2009b) or later. Furthermore, a system with 4GB or more RAM memory is recommended.

1.2 Installation

In order to use the FEATool toolbox the included files and directories must be installed on the intended computer system, and also added to the Octave and/or Matlab search paths (so that they can be found by the Octave and Matlab interpreters). The recommended installation approach and installer package will differ depending on if FEATool is to be installed with Matlab under Windows, Linux or Mac, or alternatively GNU Octave. The separate installation cases are described below.

1.2.1 FEATool with Matlab (Windows)

A convenient and automated installer ***featool-multiphysics-setup.exe*** is available for installing FEATool to run under Matlab on Windows based systems.

Following the guided steps, the installer will try to install FEATool in a suitable directory (default *C:\Precise Simulation\FEATool Multiphysics*), and automatically set up the required Matlab paths (by calling the *install.m* script). In addition, start menu and desktop shortcuts for FEATool, documentation, and uninstallation will optionally be created.

1.2.2 FEATool with Matlab (Linux/Mac)

As automatic FEATool installers are not available for Linux and Mac systems with Matlab, please use the [Installation Script](#) approach described in the corresponding section below.

1.2.3 FEATool with Octave (All systems)

The recommended method for installing FEATool under GNU Octave is to use the bundled Octave package file ***featool-multiphysics.tar.gz***. To install FEATool run the command

```
>> pkg install featool-multiphysics.tar.gz
```

in Octave (verify that FEATool is installed with *pkg list*, and make sure the *featool* package is listed). After FEATool has been installed one can load and set up the paths with

```
>> pkg load featool
```

and now simply type

```
>> featool
```

to start the FEATool GUI. If FEATool is to be frequently used it can be convenient to add the command string *pkg load featool* to the *.octaverc* startup file, typically found in the home directory or any of the [alternative Octave startup locations](#).

To uninstall FEATool, run the command

```
>> pkg uninstall featool
```

1.2.4 Installation Script

For other systems configurations, recommended FEATool installation is to use the ***featool-multiphysics.zip*** archive, extract it to a suitable location, and run the install Octave/Matlab m-file script function

found in the extracted FEATool installation folder (here it is assumed to be `C:\featool`)

```
in Octave or Matlab:
```

```
>> cd C:\featool  
>> install
```

Alternatively, instead of permanently setting up the paths it is possible to instead just run `addpaths.m` "addpaths" on the command line before starting to use the FEATool functions and subroutines (starting the FEATool GUI by running the `featool` command also does this automatically).

If this for some reason should fail, then another permanent installation approach is to first manually run `addpaths` and the **savepath** function directly afterward

```
>> addpaths  
>> ier = savepath
```

Assuming `ier` returns 0 errors, this will have permanently saved the new FEATool paths that the `addpaths` function set up. The paths are saved in either your home directory `.octaverc` file (if using Octave), or the Matlab root `pathdef.m` file (typically found as `toolbox/local/pathdef.m` in the Matlab installation directory).

To permanently uninstall (that is to remove the saved paths), run the function `uninstall` in Octave/Matlab, which also can be found in the FEATool installation folder. If the uninstallation script fails to remove all FEATool paths then one can manually remove the paths from the corresponding path file with a text editor.

1.2.5 Verification

To verify that the search paths have been installed correctly type

```
>> path
```

and check that the following paths are present in the displayed list (with location and directories adjusted for where FEATool has been installed)

```
C:\featool
C:\featool\core
C:\featool\ellib
C:\featool\examples
C:\featool\fenics
C:\featool\geom
C:\featool\grid
C:\featool\gui
C:\featool\impexp
C:\featool\lib\findjobj
C:\featool\lib\distmesh
C:\featool\lib\plotly
C:\featool\lib\surfaceintersection
C:\featool\lib\uiextras
C:\featool\openfoam
C:\featool\physmodes
C:\featool\post
C:\featool\util
```

The following example and validation test models can be run on the command line to verify that FEATool works properly. On the command line run

```
>> [fea,out] = ex_poisson1; out.pass
>> [fea,out] = ex_poisson2; out.pass
>> [fea,out] = ex_poisson5; out.pass
```

and verify that the solutions look reasonable and that the *out.pass* variable returns 1=true (more example models can be found in the *examples* directory). Lastly, simply type *featool* or double click on the installed FEATool desktop icon to start the FEATool GUI.

1.2.6 External Solvers

FEATool Multiphysics includes both GUI and CLI support for the FEniCS and OpenFOAM solvers. Due to very differing system and installation processes these solvers must be installed by the user separately. Please consult the corresponding FEniCS and OpenFOAM sections for suggested installation and usage.

1.2.7 External Grid Generators

As FEATool Multiphysics comes with built-in support for the external mesh generators [Gmsh](#) and [Triangle](#), upon starting the FEATool GUI for the first time a dialog box will ask to automatically download and install them (recommended).

Advantages of using either *Gmsh* or *Triangle* compared to the built-in (*DistMesh* based) grid generation function is both robustness and mesh generation speed. Moreover, external grid generators also supports better and more control with a selection of different mesh generation algorithms, and specifying the grid size in different geometry regions, subdomains, as well as on boundaries, allowing for greater flexibility and better grids tuned for the specific problems and geometries. In addition Gmsh also supports generating unstructured 2D quadrilateral grids automatically.

1.3 Modeling Process

The typical modeling process in FEATool logically follows the six mode buttons located in the upper portion of the left side toolbar in the main graphical user interface (GUI) window. The modeling steps are thus as follows:

- **Geometry** The first step is to create a geometry to define the domain to be modeled. Geometry objects such as rectangles, circles, blocks, and cylinders can be created and combined to represent complex domains.
- **Grid** From the geometry a grid or mesh can be generated or imported from an external preprocessing tool or grid generator. The modeled phenomena is to be approximated on each grid cell by a finite element polynomial or shape function. Finer more dense grids and regularly shaped high quality grid cells generally leads to more accurate solutions.
- **Equation** Equations (PDEs) and model coefficients are then specified in subdomain/equation mode to describe the physical phenomena to be modeled. Equations are pre-defined for many types of physical phenomena like for example heat transfer, structural strains and stresses, and fluid flow. In addition FEATool also allows arbitrary and custom systems of PDE equations to be described and entered.
- **Boundary** Boundary conditions must be prescribed on the geometrical boundaries to account for interactions with the surroundings outside of the modeled domain. For time dependent problems initial conditions must also be given at the start of the simulations.
- **Solve** After the model problem is fully specified a suitable solver can be employed to compute a solution. FEATool can either use the default Matlab or Octave linear solvers or use specialized external ones, such as the OpenFOAM CFD solver or FEniCS for increased performance.
- **Post** Finally, after the problem has been solved, the solution can be visualized, post-processed, and exported to evaluate the computed results.

1.4 Working with FEATool

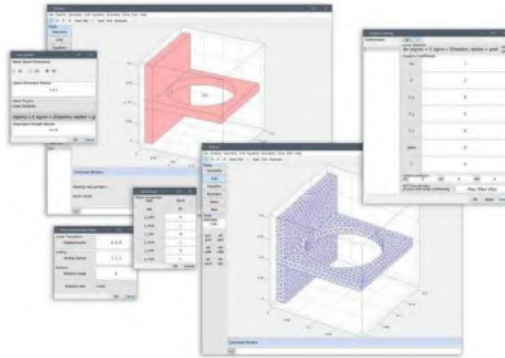
FEATool Multiphysics is unique in that it allows several different ways for users to work with FEM modeling and simulation. The whole spectrum from using the high-level graphical user interface down to low-level access of the fundamental matrices of the underlying finite element FEM discretization is possible. On top of this, since FEATool is written in m-script code, it can be extended and combined with Matlab and Octave toolboxes and custom m-file scripts and functions. The four different ways of working with FEATool are using

1. Graphical User Interface (GUI)
2. Pre-defined physics modes
3. Weak FEM equation formulation
4. Direct matrix assembly

where the last three approaches involve eschewing the GUI for the Matlab and Octave command lines and m-script files.

1.4.1 Graphical User Interface (GUI)

Designed with ease of use in mind, the graphical user interface or GUI is usually the first way one works with FEATool.

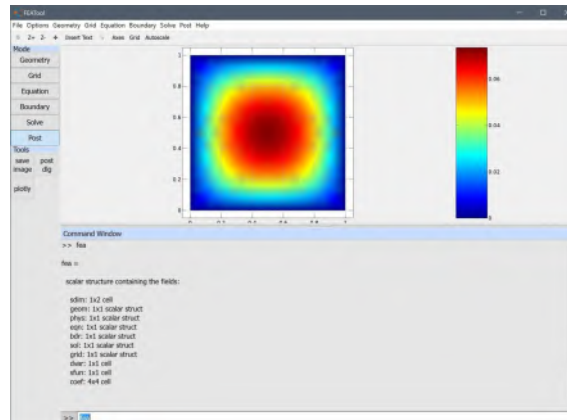


Although simple in nature, the FEATool GUI allows for some powerful features that can be utilized to extend code functionality.

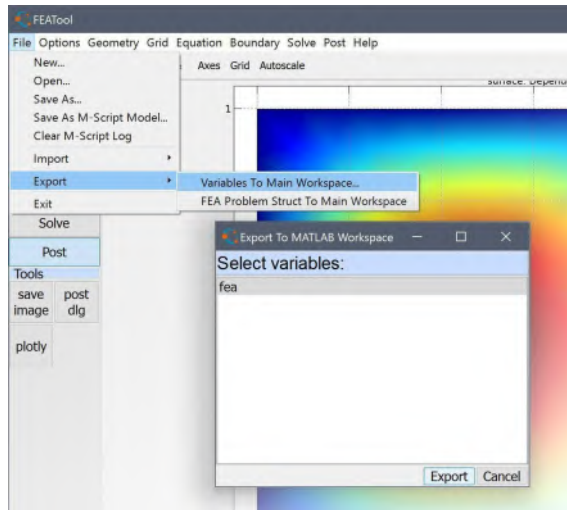
Firstly, in the lower part of the main GUI workspace, right below the *Command Window* output log, is the FEATool command prompt `>>` which can accept and interpret Matlab and Octave commands. The entered commands are evaluated in the local FEATool memory workspace and thus allows using Matlab operations without exporting models to the main Matlab workspace. The variable **fea** contains the FEATool finite element struct for the current model. For example, try entering the command

```
fea
```

to see what the **fea** struct contains. More information about the composition of the **fea** struct can be found in the problem definition section. Be aware that manipulating this variable directly in the GUI may cause the program to crash if invalid input is given.

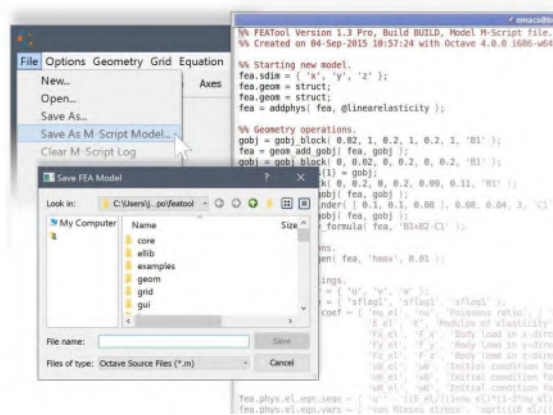


Furthermore, it is possible to import and export data and variables between the GUI and Matlab and Octave main command line workspaces by using the corresponding **Import** and **Export** menu options under the **File** menu.



1.4.2 Pre-defined physics modes

The first step in working with Matlab and Octave command line and m-script FEATool models is generally to use the pre-defined physics modes. A good way to understand and learn how FEATool script models are built up and constructed is to use the **Save As M-Script Model...** option instead of the binary .fea format file.



The output .m model source files can be opened in any text editor and the shown output corresponds directly with how the model was constructed in the GUI. Everything from the geometry definition, to solving and postprocessing has corresponding Matlab function commands.

An example of solving the Poisson equation for a unit circle defined with the Poisson physics mode can be found as the `ex_poi1_cmd1` tutorial example. As can be seen in the example code, the physics modes are stored in the `fea.phys` field of the `fea` model definition struct. The actual strong PDE formulation is defined in the `seqn` sub-field, in this case

```
fea = addphys( fea, @poisson );
fea.phys.poi.eqn.seqn = 'dts_poi*u' - d_poi*(ux_x + uy_y) = f_poi'
```

PDE equation coefficients can be specified as

```
fea.phys.poi.eqn.coef = { 'dts_poi' [] [] { 1 } ;
                          'd_poi'   [] [] { 1 } ;
                          'f_poi'   [] [] { 1 } ;
                          'u0_poi'  [] [] { 0 } };
```

where `u0_poi` is the initial condition for the dependent variable `u` as defined in `fea.phys.poi.dvar`. Similarly Dirichlet and Neumann boundary conditions can be prescribed through the `bcr_poi` and `bcr_g_poi` coefficients, respectively


```
fea.phys.poi.bdr.sel = 1;
fea.phys.poi.bdr.coef = ...
{ 'bcr_poi' [] [] [] [] [] { 0 } ;
  'bcg_poi' [] [] [] [] [] { 1 } };
```

Lastly, the command `parsephys`

```
fea = parsephys( fea );
```

parses the equations in all the `fea.phys` structs and enters the corresponding weak finite element formulations into the global `fea.eqn` and `fea.bdr` multiphysics fields.

1.4.3 Weak FEM equation formulation

If one prefers to directly work with the FEM weak formulations this is also possible as shown in the `ex_poi1_cmd2` example for the same Poisson problem as above. Note how there is no `phys` field or call to `parsephys` necessary when directly prescribing the `fea.eqn` and `fea.bdr` fields

```
fea.eqn.a.form = { [2 3; 2 3] };
fea.eqn.a.coef = { [1 1] };
fea.eqn.f.form = { 1 };
fea.eqn.f.coef = { 1 };

n_bdr          = max(fea.grid.b(3,:));
fea.bdr.d       = cell(1,n_bdr);
[fea.bdr.d{:}] = deal(0);
fea.bdr.n       = cell(1,n_bdr);
```

1.4.4 Direct matrix assembly

Finally, for advanced FEM users it is entirely possible to use the core finite element assembly routines `assemblea` and `assemblef` to assemble the system matrix and right hand side/load vector after which they can be directly manipulated. Again, the corresponding Poisson problem example is described in the CLI Poisson equation tutorials. The source code for the matrix and source term vector assembly looks like the following

```
form = [2 3;2 3];
sfun = {'sflag1','sflag1'};
coef = [1 1];
i_cub = 3;    % Numerical quadrature rule to use.

[vRowInds,vColInds,vAvals,n_rows,n_cols] = ...
    assemblea( form, sfun, coef, i_cub, ...
        fea.grid.p, fea.grid.c, fea.grid.a );

A = sparse( vRowInds, vColInds, vAvals, n_rows, n_cols );

form = [1];
sfun = {'sflag1'};
coef = [1];
i_cub = 3;

f = assemblef( form, sfun, coef, i_cub, ...
    fea.grid.p, fea.grid.c, fea.grid.a );
```

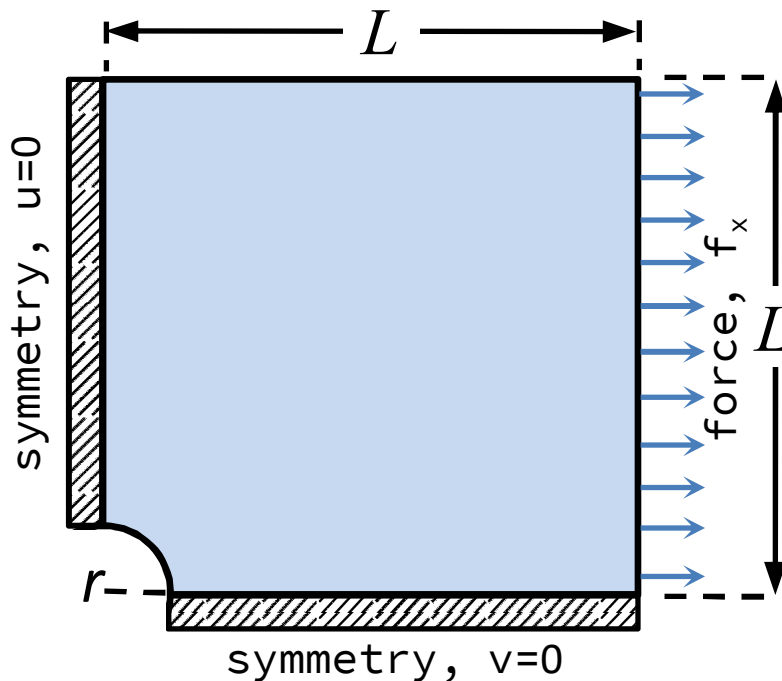
Note that here we have to directly prescribe boundary conditions to the system matrix and right hand side vector.

Altogether one can see that FEATool together with Matlab and Octave allows for many possibilities to set up, perform, and analyze multiphysics FEM simulations.

1.5 Structural Mechanics Example - Thin Plate with Hole

The first example is a well known benchmark test case from structural mechanics, a thin plate with a circular hole in the center is subjected to a load along one axis.

The plate is thin enough to satisfy the two dimensional plane stress approximation, and since both the problem and solution will be symmetric around the hole it is enough to model a quarter of the plate. The computational geometry will therefore consist of a 0.05 by 0.05 m square with a quarter of a circle with radius 0.005 m removed from one corner. Due to the symmetry the displacement of the left edge of the plate should be zero in the x -direction, and similarly the y -displacement for the lower edge should also be zero. Furthermore, a horizontal force of 1000 N is applied to the right edge. With a plate thickness of 0.001 m , the resulting load will be $1000/(2*0.05*0.001)$ N/m^2 .



Assuming that the plate is made of steel with a Poisson ratio of 0.3 and modulus of elasticity $210 \times 10^9 \text{ Pa}$ then it is expected that the maximum stress in the x-direction will be three times the stress of a plate without a hole, that is $\sigma_x = 3 \times 1000 / (2 \times 0.05 \times 0.001) \text{ Pa} = 3 \times 10^7 \text{ Pa}$.

1.5.1 Thin Plate with Hole using the GUI

This section describes how to set up and solve the thin plate with hole example with the FEATool graphical user interface (GUI).

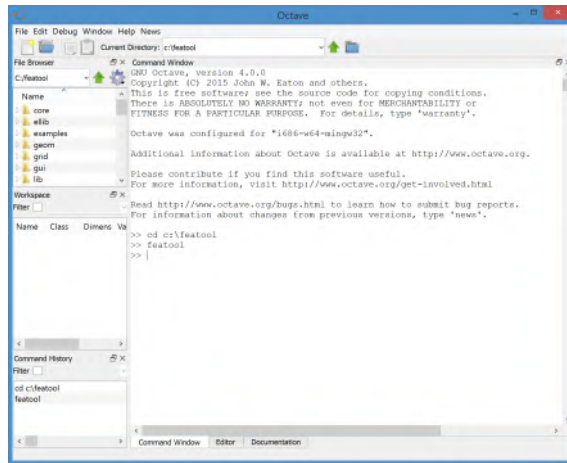
1. Start Octave or Matlab. If you have not run the installation script (which automatically adds the FEATool directory paths at startup) then change your working directory to where your FEATool installation is, for example


```
cd C:\featool
```

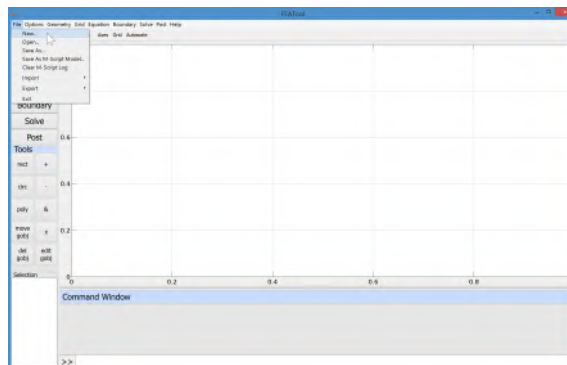
2. In the Octave/Matlab main command window type

```
featool
```

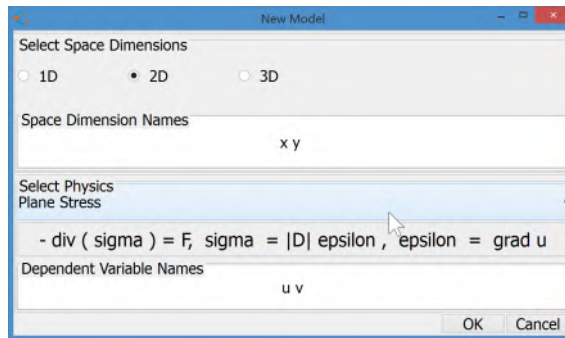
to start the FEATool graphical user interface (GUI).




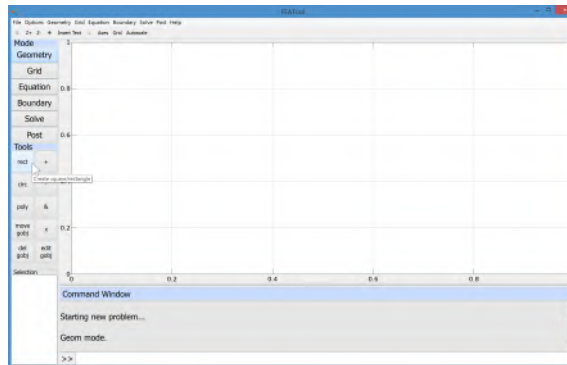
3. Either select **New...** from the **File** menu, or click on the **New Model** button  in the upper horizontal toolbar, to clear all data and start defining a new model.



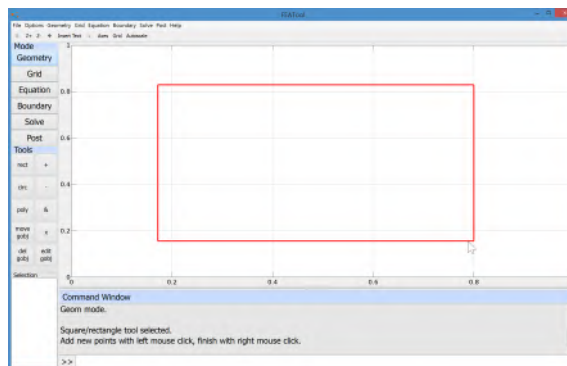
4. In the *New Model* dialog box, click on the **2D** radio button in the *Select Space Dimensions* frame, and select **Plane Stress** from the *Select Physics* drop-down list. Leave the space dimension and dependent variable names to their defaults. Finish and close the dialog box by clicking on the **OK** button.




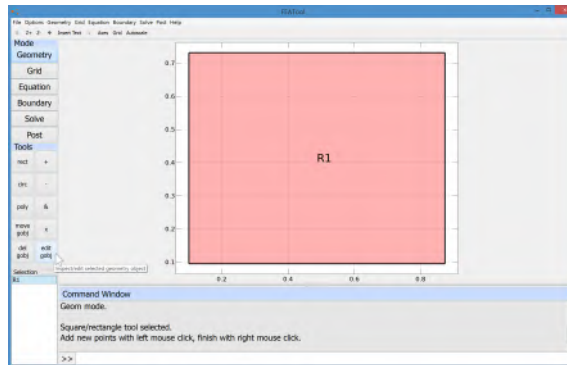
5. To create the rectangle for the plate, first click on the **Create square/rectangle** button  in the left hand side *Tools* toolbar frame.



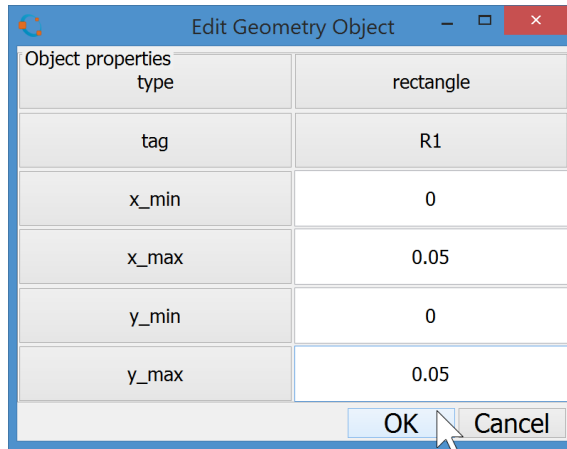
- Then left click in the main plot axes window, hold the mouse button, and move the mouse pointer to draw a rectangle with red outlines.




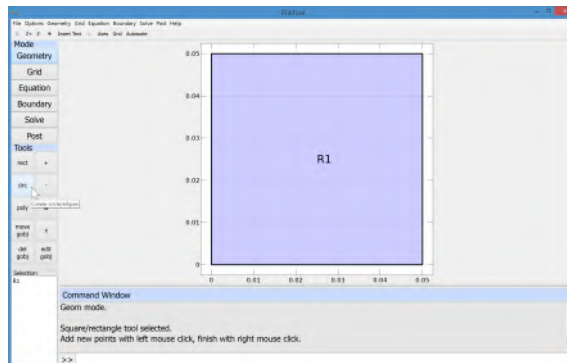
7. Release the button to finalize and create a solid geometry object. The object properties must now be edited to set the correct size and position of the rectangle. To do this, click on the rectangle *R1* to select it and highlight it in red. Then click on the **Inspect/edit selected geometry object** toolbar button 



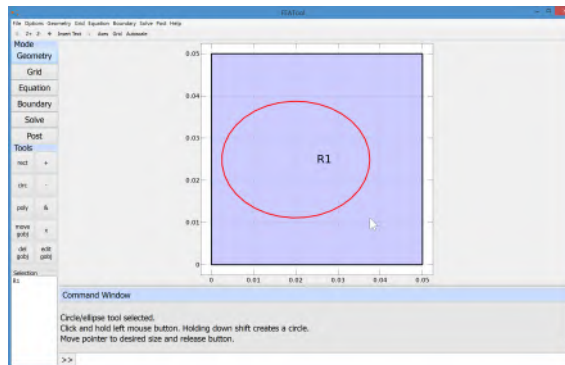
8. In the **Edit Geometry Object** dialog box, change the min and max point coordinates to define a rectangle with length and height **0.05** and lower left corner at the origin. Finish editing the geometry object and close the dialog box by clicking **OK**.




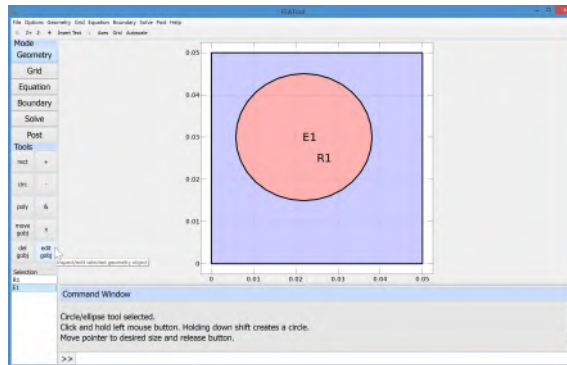
9. To create the circular hole, first left click on the **Create circle/ellipse** button  in the left hand side *Tools* toolbar frame.



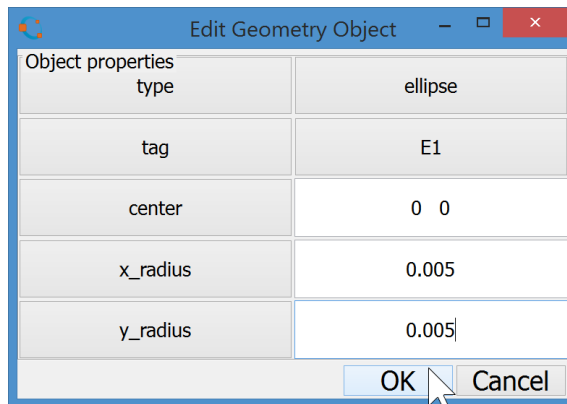
10. Then click and hold the left mouse button anywhere in the main plot axes window, move the mouse pointer to show red outlines of a circle or ellipse. Release the button to finalize and a solid ellipse will be created.




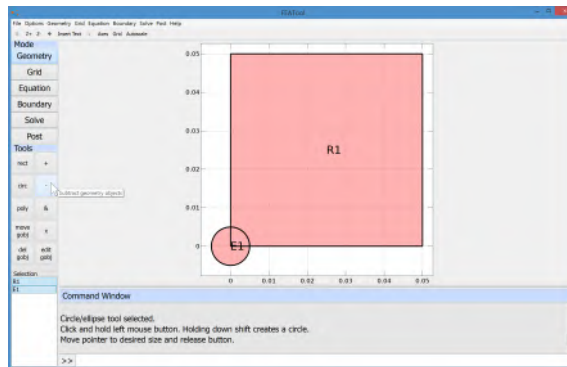
11. The object properties of the ellipse *E1* must be changed to make a circle with radius 0.005 centered at (0, 0). To do this, click on the *E1* to select it which also highlights it in red. Then click on the **Inspect/edit selected geometry object** toolbar button 



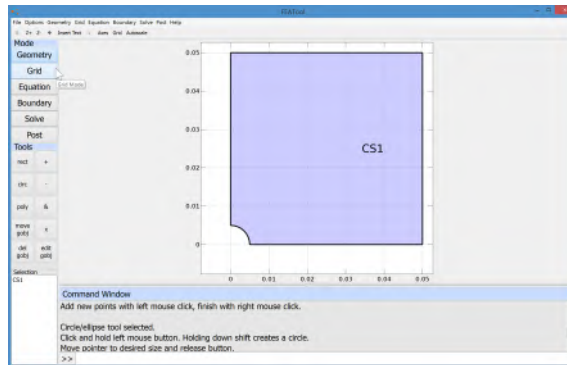
12. In the opened **Edit Geometry Object** dialog box change the center coordinates to **0 0**, and the x and y radii to **0.005** in the corresponding edit fields. Finish editing *E1* and close the dialog box by clicking **OK** (the smaller objects will automatically be selected to be subtracted from the larger).



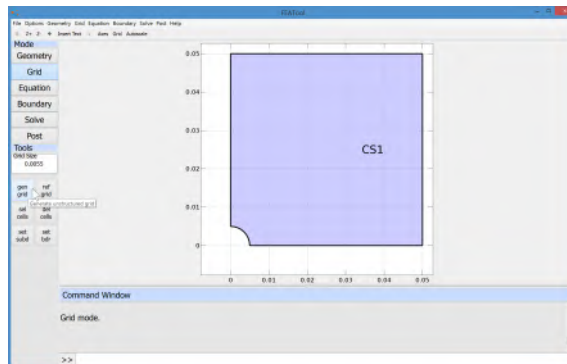
13. To subtract the circle from the rectangle first select both geometry objects by clicking on them so both are highlighted in red. Alternatively, if the circle is obscured by the rectangle they can be selected by holding the *Ctrl* key while clicking on the labels **R1** and **E1** in the *Selection* list box (or simply press *Ctrl+a* to select all objects). When the geometry objects are selected, press the **Subtract geometry objects** button  to generate the final geometry shape.




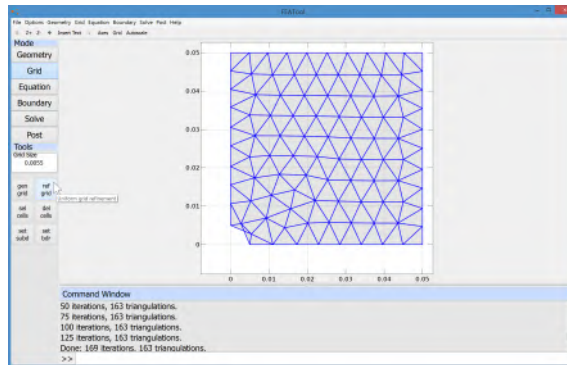
14. Switch to *Grid* mode by clicking on the corresponding  mode toolbar button.




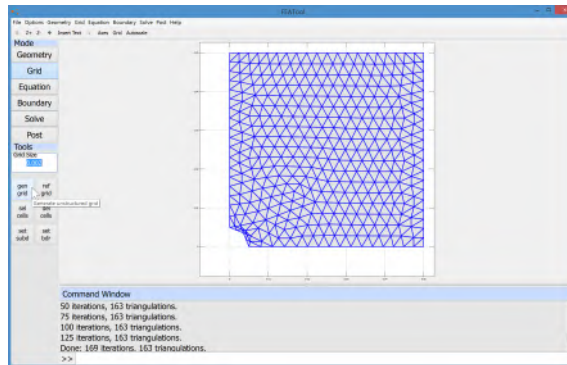
15. Click on the **Generate** button Generate to call the grid generation function which automatically generates a triangular grid for the geometry.



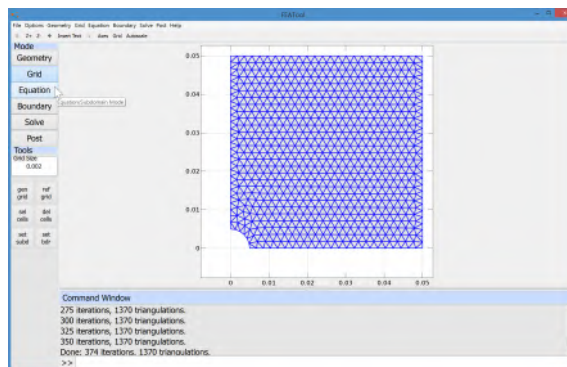
16. The default estimated grid size is sufficient to resolve the rectangle but the hole is not represented well. Try refining the grid by clicking on the **Uniform grid refinement** button  once or twice.



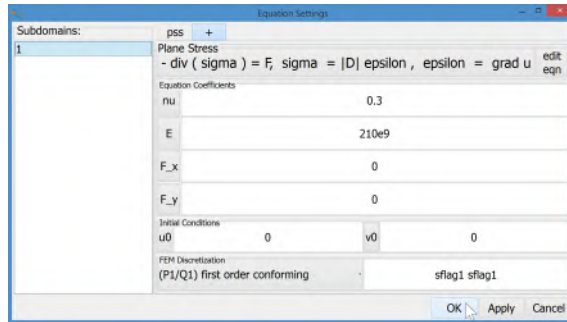
17. The uniformly refined grid does not improve resolution of curved boundaries and in this case also creates very distorted cells which leads to poor accuracy. To generate a better grid enter **0.002** in the **Grid Size** parameter edit field, or use the slider control to set the grid size, and click on the **Generate** button  again.




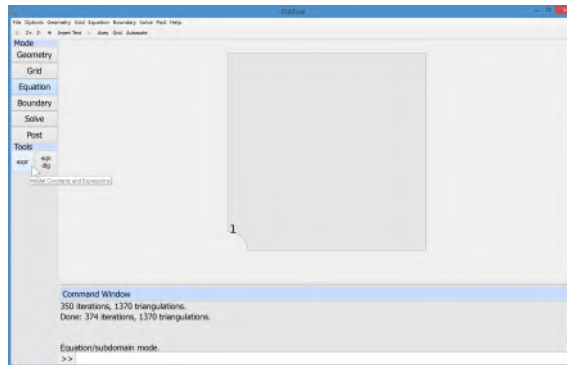
18. With an improved grid we now can proceed with specifying the equations. Press the **Equation** mode button in the *Mode* toolbar to switch from *grid* mode to *physics and equation/subdomain* specification mode.



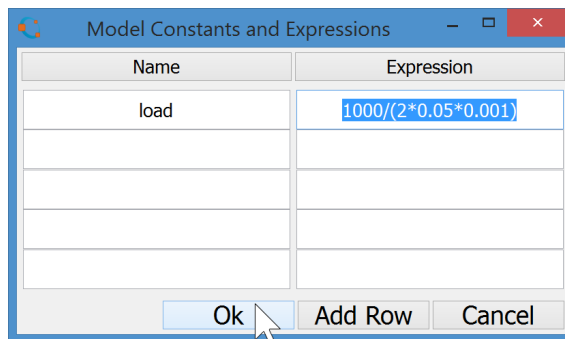
19. In the *Equation Settings* dialog box that automatically opens, enter **0.3** for the Poisson ratio ν and **210e9** for the modulus of elasticity E . The other coefficients can be left to their default values. Press **OK** to finish and close the dialog box.



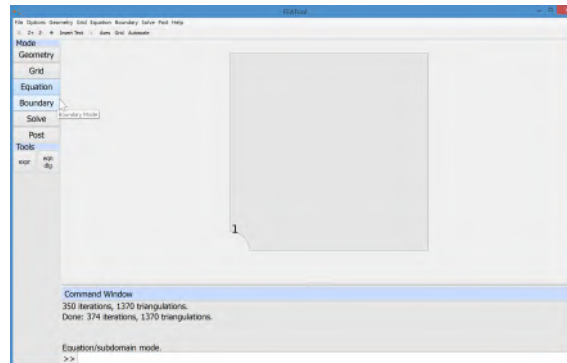
20. Now we want to add an expression for the load force. We could do it by directly entering the expression in the dialog box, however there are advantages to using the constants and expressions functionality. To use this click on the **Model Constants and Expressions** button 



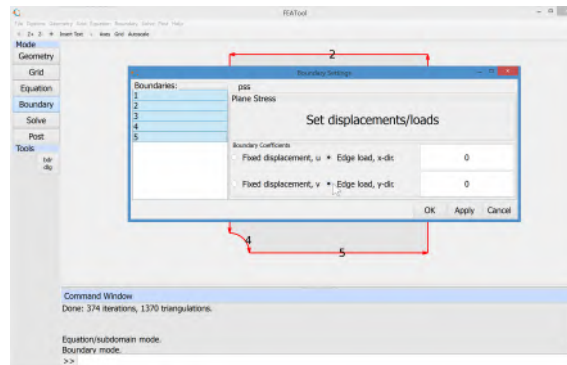
21. Enter a new variable named **load** with expression **$1000 \cdot (2 \cdot 0.05 \cdot 0.001)$** (In this case we are just entering constants one can also use complicated formulas involving dependent variables, space coordinates, time, and other expressions). Press **OK** to finish and close the dialog box.



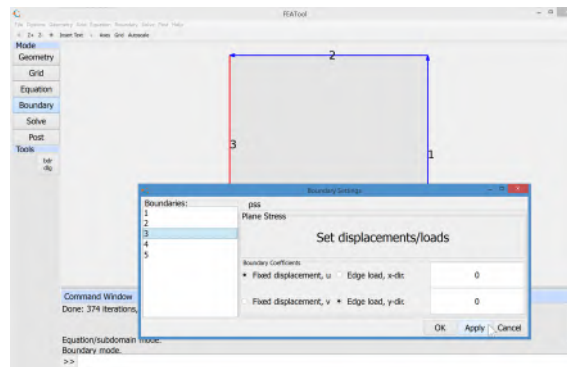
22. Change to *boundary condition specification* mode by clicking on the Boundary mode button.



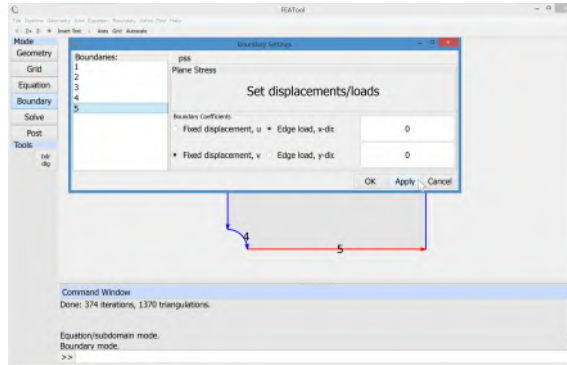
23. In the *Boundary Settings* dialog box, first select all boundaries and set all conditions to **Edge loads** with a value of zero, **0**.



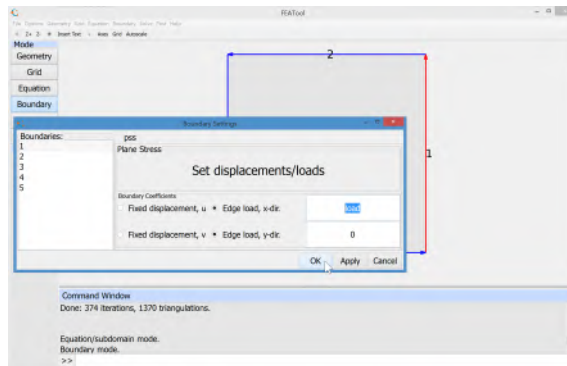
24. Then select the left boundary (number 3 in this case) in the left hand side *Boundaries* list box and select **Fixed displacement, u** and zero **Edge load, y-dir.**. This will fix the x-displacement of this boundary.

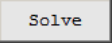



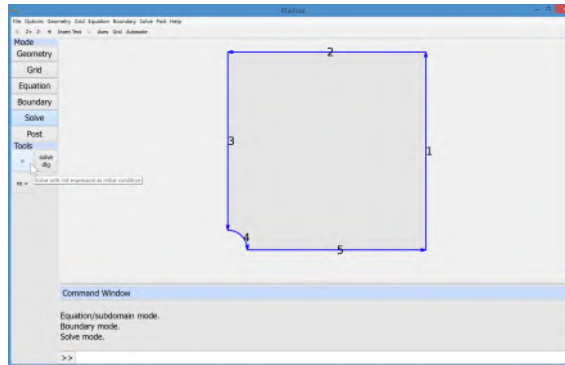
25. Continue by selecting boundary 5, the bottom boundary, and choose a zero **Edge load, x-dir.** and **Fixed displacement, v** boundary conditions. This similarly fixes the lower boundary.




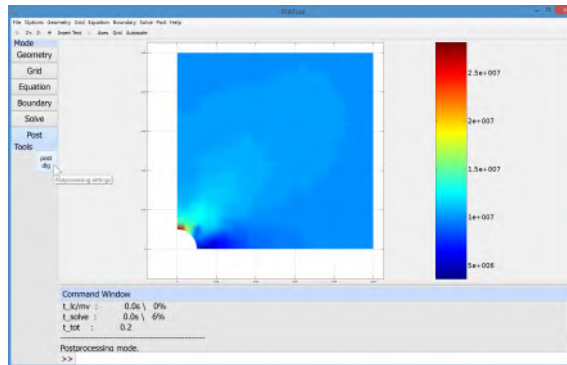
26. Lastly, select both **Edge load** boundary conditions for the right boundary (number 1). Set the edge load in the x-direction on this boundary to **load**, which will be evaluated from the expression we entered earlier. Finish by clicking the **OK** button.



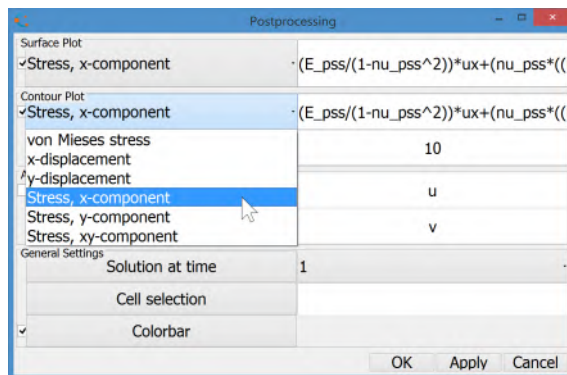
27. Now that the problem is fully specified, press the  mode button to switch to *solve* mode. Then press the button with an equals sign  in the *Tools* toolbar frame to call the solver with the default solver settings.



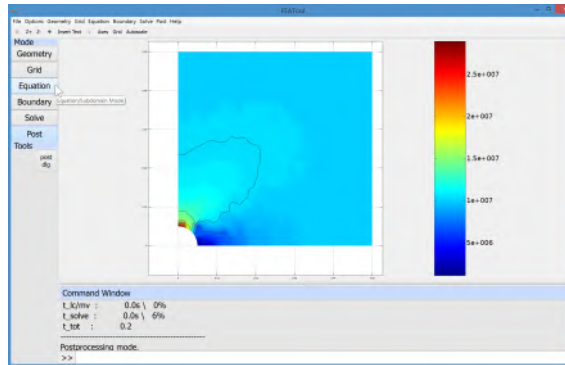
28. After the problem has been solved FEATool will automatically switch to *postprocessing* mode and display the computed von Mises stress. To change the plot, open the post-processing settings dialog box by clicking on the **Postprocessing settings** button  in the *Tools* toolbar frame.



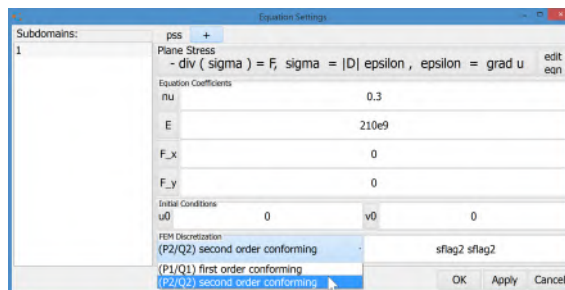
29. In the Postprocessing settings dialog box choose to plot the **Stress, x-component** for both the surface and contour plots.



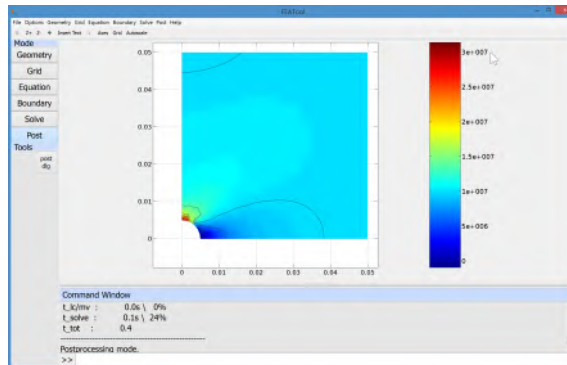
30. We can see that the solution for the stress is not smooth. This is due to the fact that stress is a function of derivatives of the displacements (the solution variables) and a linear solution approximation is used per default, this means that stresses will be represented as piecewise constant functions. We can improve on this by going back to the **Equation** specification mode.



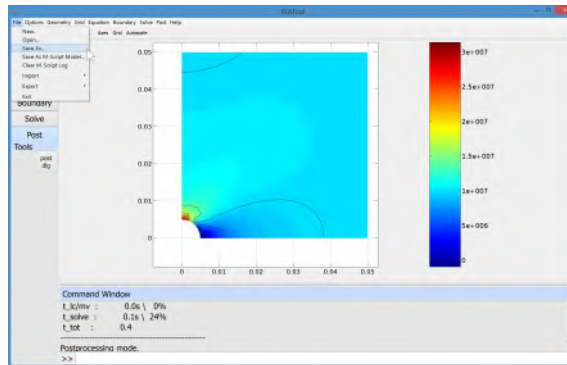
31. Select **P2/Q2 second order conforming** for the finite element shape function specification and **Solve** the problem again.



32. Now we can see that the plotted stress is smooth and the maximum value is slightly above 3×10^7 which is the expected solution.



33. Saving the model can be done from the **File** menu. **Save As...** allows you to save the model in binary (.fea) format which can be loaded into the GUI again. **Save As M-Script Model...** allows you to save all FEATool function commands used to make the model as a m-script file. This file can not be loaded into the GUI but inspected, modified, and run on the command line as a script file.

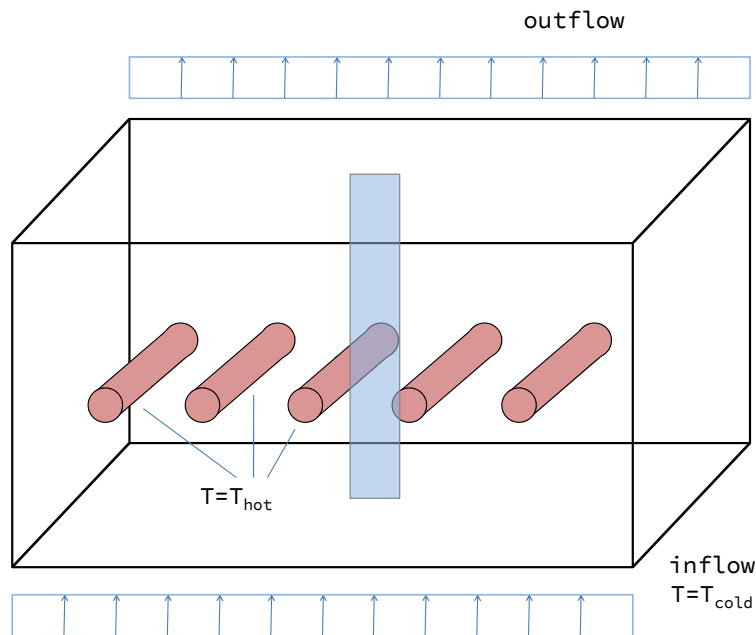


1.5.2 Thin Plate with Hole using the CLI

The process to set up and solve the thin plate with hole example problem on the command line interface is illustrated in the `ex_planestress1` script file which can be found in the examples directory. Moreover, if you export the model using the **Save As M-Script Model...** option then one can easily see exactly which FEATool functions and commands are used to build the model.

1.6 Multiphysics Example - Heat Exchanger

This example illustrates the multiphysics capabilities of FEATool with a simple heat exchanger model featuring both free and forced convection. The model consists of a series of heated pipes around which there is a lower temperature fluid flowing. Two kinds of physics are considered, fluid flow which is modeled by the Navier-Stokes equations and heat transport modeled by a convection and conduction equation for the temperature field. The Boussinesq approximation models the temperature effects on the fluid, and the flow field is coupled to and transports the temperature field. In this way the system is fully two way coupled, the fluid to the temperature and temperature to the fluid.



Due to symmetry it is enough to study a two dimensional slice between the heated pipes. The geometry will therefore consist of a 0.0075 by 0.05 m rectangle with a half circle removed (with radius 0.003 m centered at $(0, 0.02)$). The mechanism for heating the pipes are not taken in consideration and are thus assumed to be at a fixed temperature of $T_h=330\text{ K}$. A cooling fluid

flows from the bottom to the top and has an inlet temperature of $T_h=300\text{ K}$. The other model parameters can be found in the following model description.

1.6.1 Heat Exchanger using the GUI

This section describes how to set up and solve the heat exchanger example with the FEATool graphical user interface (GUI).

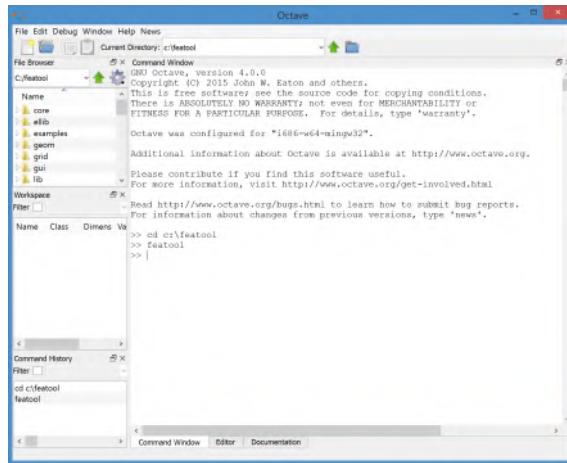
1. Start Octave/Matlab, and if you have not run the installation script (which automatically adds the FEATool directory paths at startup) then change your working directory to where your FEATool installation is, for example


```
cd C:\featool
```

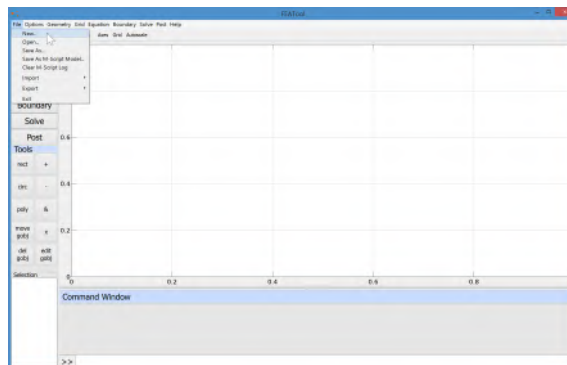
2. In the command window type

```
featool
```

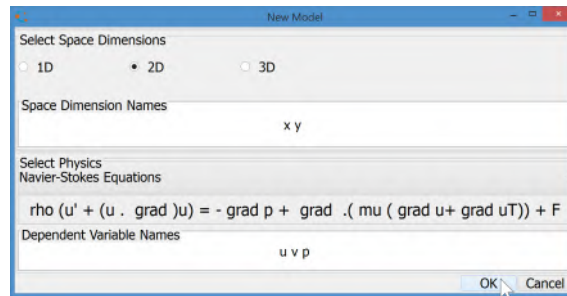
to start the graphical user interface (GUI).




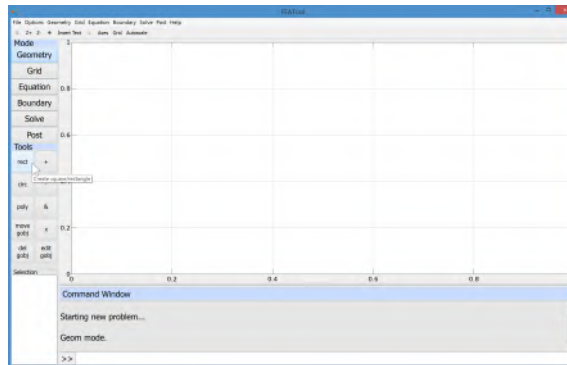
3. Either select **New...** from the **File** menu, or click on the **New Model** button  in the upper horizontal toolbar, to clear all data and start defining a new model.



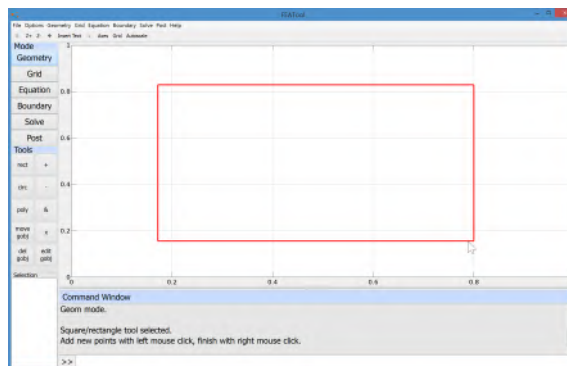
4. In the opened *New Model* dialog box, click on the **2D** radio button in the *Select Space Dimensions* frame, and select **Navier-Stokes Equations** from the *Select Physics* drop-down list. Leave the space dimension and dependent variable names to their default values. Finish and close the dialog box by clicking on the **OK** button.




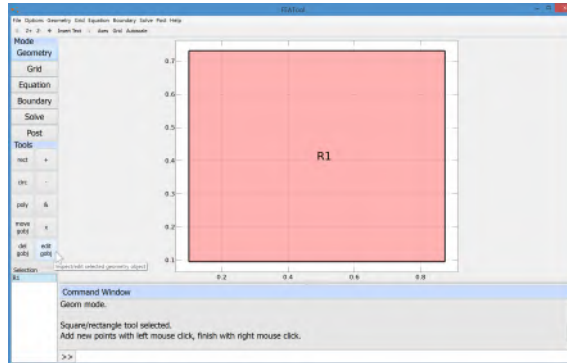
5. To define the geometry, first create a rectangle by clicking on the **Create square/rectangle** button  in the left hand side *Tools* toolbar frame.



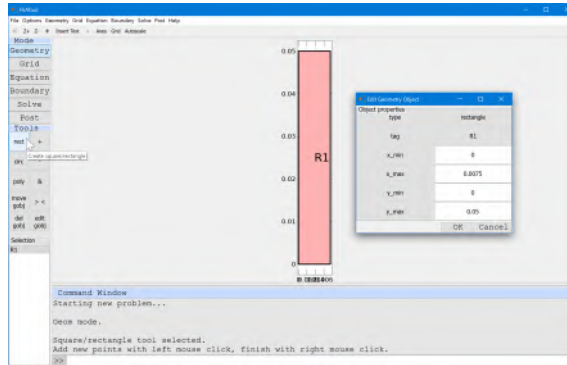
- Then left click in the main plot axes window, hold the mouse button, and move the mouse pointer to draw a rectangle.



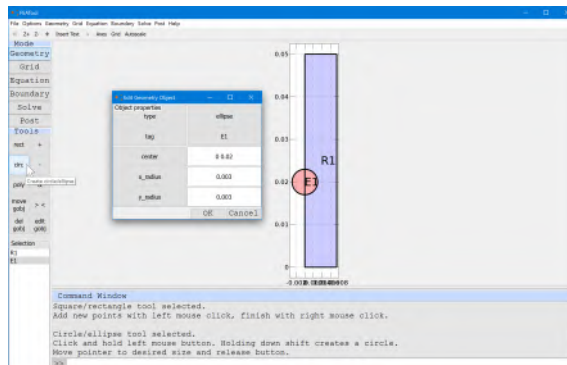
7. Release the mouse button to finalize and create a solid geometry object. The object properties must now be edited to set the correct size and position of the rectangle. To do this, click on the rectangle *R1* to select it and highlight it in red. Then click on the **Inspect/edit selected geometry object** toolbar button 




8. Change the minimum and maximum x-coordinates to **0** and **0.0075**, respectively. Also change the y-dimensions to span between **0** and **0.05**. Press **OK** to finish editing the rectangle properties.

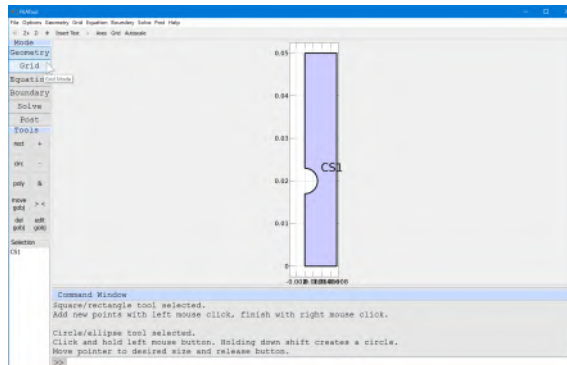


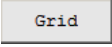
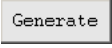
9. In a similar way, create a circle centered at **(0, 0.02)** with a radius of **0.003**.

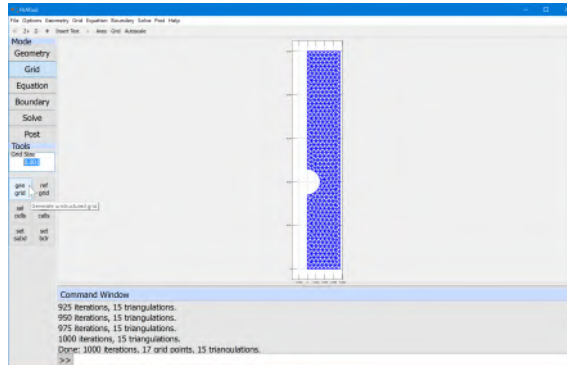


10. To create the final geometry select both the rectangle and circle so they are highlighted in red (either by directly clicking on them or selecting them in the *Selection* list box). Then

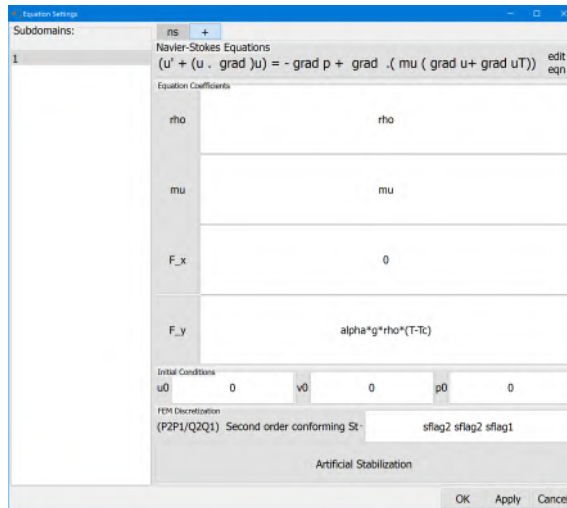
click on the **Subtract geometry objects** button  to subtract the smaller circle from the larger rectangle.



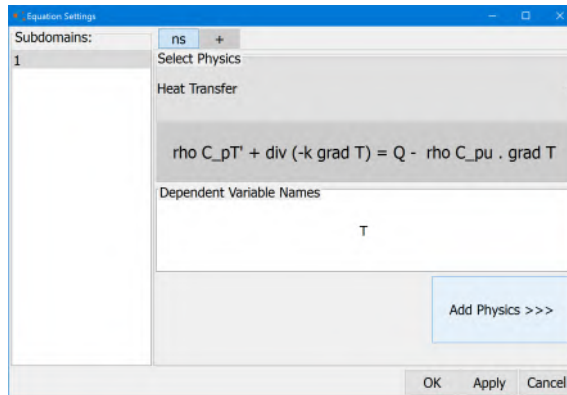
11. Switch to *Grid* mode by clicking on the corresponding  mode toolbar button. Enter **0.001** in the **Grid Size** parameter edit field and click on the **Generate** button  to automatically generate a grid.



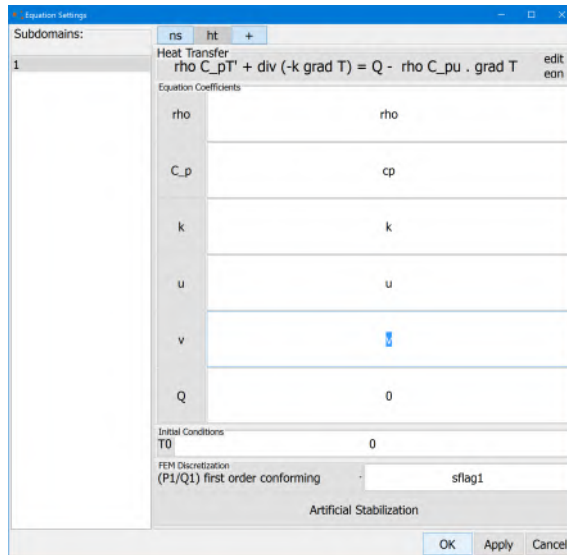
12. Press the **Equation** mode button in the *Mode* toolbar to change to *physics and equation/-subdomain* specification mode. In the *Equation Settings* dialog box that automatically opens enter the following coefficients, **rho** for the density, **mu** for the viscosity, and **alpha*g*rho*(T-Tc)** for the volume force in the y-direction.




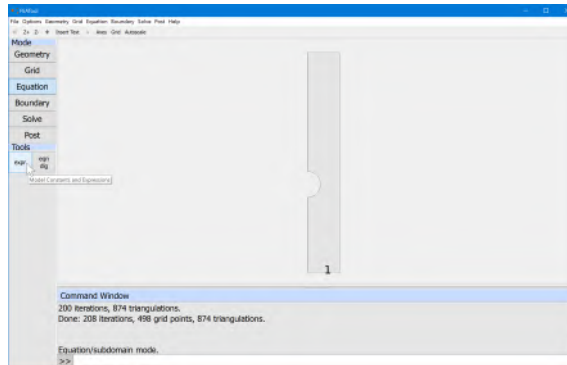
- We now have to add a *heat transfer* physics mode. To access the multiphysics selection and add another physics mode press the plus + tab and select **Heat Transfer** from the *Select Physics* drop down list. Add the selection by pressing the **Add Physics >>>** button.



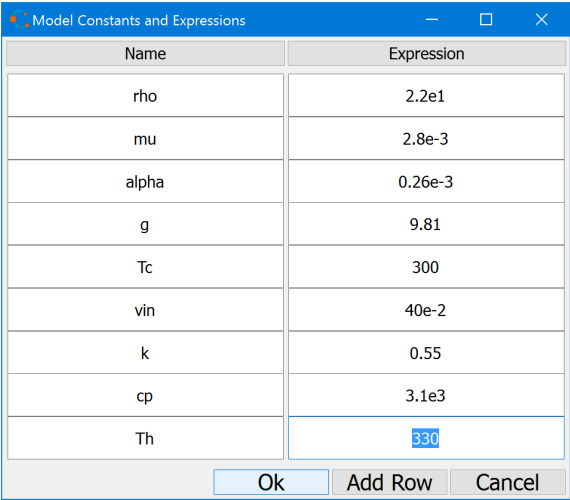
14. In the *Equation Settings* *ht* tab, set the density ρ , specific heat C_p , and heat conductivity k to **rho**, **cp** and **k**, respectively. The convective velocities should be coupled from the Navier-Stokes equations physics mode, to do this enter **u** and **v** in the corresponding edit fields (as these are the default names of the dependent variables for the velocities). Press **OK** to finish with the equation specifications.



15. The values of the specified coefficients must now be prescribed. Click on the **Model Constants and Expressions** button  to open the corresponding dialog box.

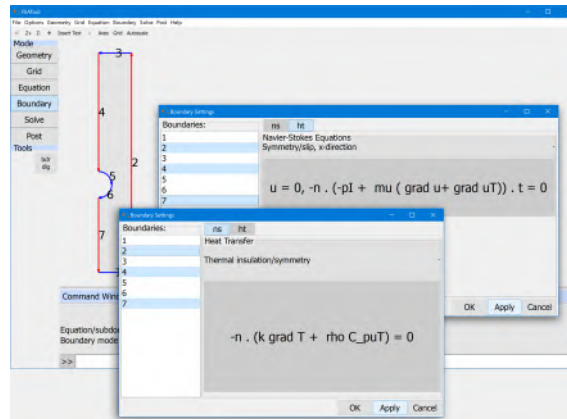


16. Enter the values shown below in the **Model Constants and Expressions** dialog box. Space for more constants can be made by clicking to the **Add Row** button. Press **OK** to finish.

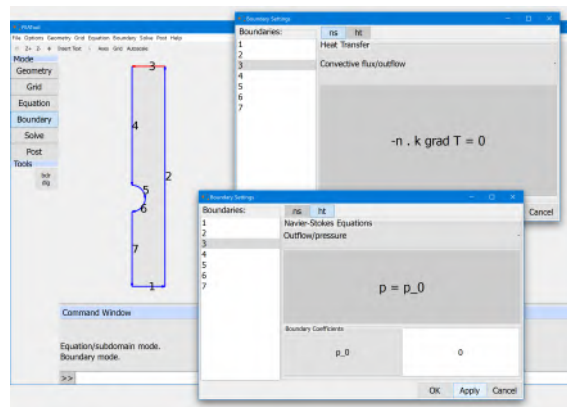


Name	Expression
rho	2.2e1
mu	2.8e-3
alpha	0.26e-3
g	9.81
Tc	300
vin	40e-2
k	0.55
cp	3.1e3
Th	330

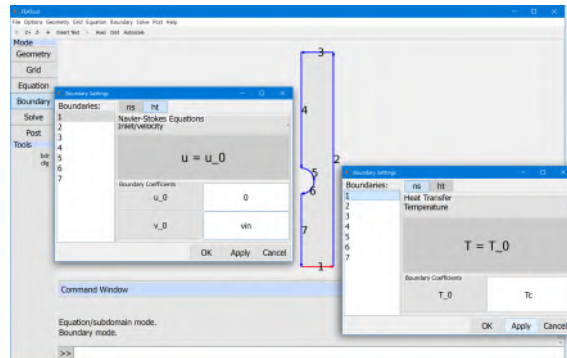
17. Switch to *boundary condition specification* mode by clicking on the **Boundary** mode button. First select the **ns** tab, which corresponds to the boundary conditions prescribed to the Navier-Stokes equations physics mode. Then select all vertical boundaries (here 2, 4, and 7). Choose **Symmetry/slip, x-direction** from the drop down box. Switch to the heat transfer physics mode by selecting clicking on the **ht** tab and select **Thermal insulation/symmetry** boundary conditions.



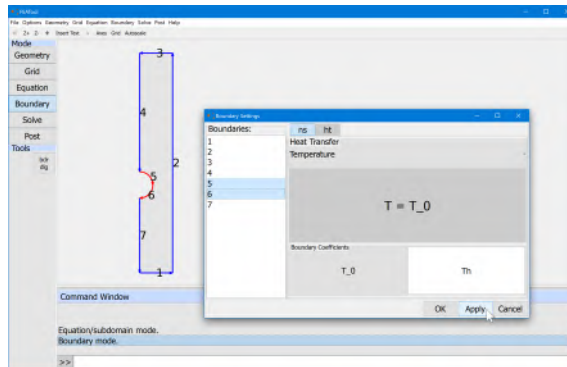
18. Now continue with the top boundary (number 3) which is the outflow. Select **Outflow/-pressure** for the Navier-Stokes physics mode and **Convective flux/outflow** for the heat transfer mode.

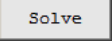




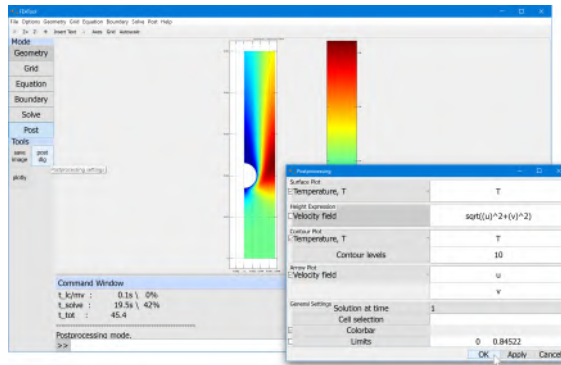
19. The bottom boundary (number 1) is the inflow and should be prescribed with the constant velocity **uin** in the y-direction by using the **Inlet/velocity** condition. The **Temperature** should here be fixed to **Tc**, the lower temperature.



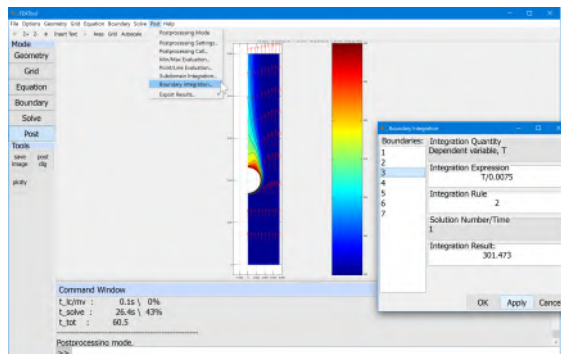
20. Lastly, the boundaries on the cylinder (5 and 6) are walls and should be prescribed with **Wall/no-slip** boundary conditions for the velocity. For the **Temperature** a constant high temperature of **Th** should be prescribed. Press **OK** to finish prescribing boundary conditions.



21. Now that the problem is fully specified, press the  mode button to switch to *solve* mode. Then press the button with an *equals* sign  to start the solution process.
22. After the problem has been solved FEATool will automatically switch to *postprocessing* mode and display the computed solution. We can see the the velocity will be accelerated when passing between the cylinders. Open the postprocessing settings dialog box by clicking on the **Postprocessing settings** button  and select to view the *Temperature*, *T*.



23. We can clearly see how the fluid is heated around the hot cylinder and follows the flow upwards. FEATool Multiphysics and Professional also allows advanced postprocessing such as boundary integration. By integrating the expression T/w (where w is the width 0.0075 of the domain) we effectively calculate the mean temperature and can see that at the outflow the temperature has risen by about 1.5 degrees.



1.6.2 Heat Exchanger using the CLI

The process to set up and solve the heat exchanger multiphysics problem on the command line interface is illustrated in the `ex_heat_exchanger1` script file which can be found in the examples directory. Moreover, you can also export the model using the **Save As M-Script Model...** option and see exactly which FEATool commands are used in building the model.

1.7 Equation Editing Example - Axisymmetric Fluid Flow

FEATool is designed to be able to perform complex Octave and Matlab multiphysics simulations in arbitrary dimensions (1D, 2D, and 3D). However, running full 3D simulations often requires a significant amount of computational resources in the form of memory and simulation time. It is therefore desirable to find simplifications to reduce simulations to two or even one dimension if possible.

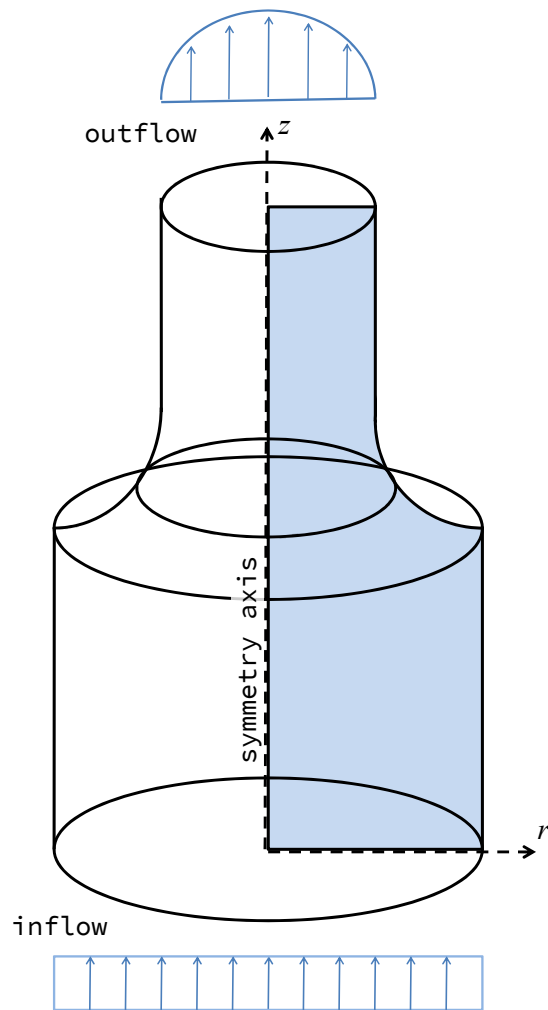
Problems which feature cylindrical and rotationally symmetric geometries and solutions can be reduced to two dimensions through a cylindrical or axisymmetric coordinate transformation (also referred to 2.5D). A symmetry axis, usually $r=0$, is taken as reference around which the coordinates and PDE operators (gradient and divergence) are transformed. In this way the governing equations will be reduced to 2D while representing a rotationally symmetric three dimensional problem.

This example models fluid flow in a narrowing pipe section. The constriction of the pipe will accelerate the flow according to the venturi effect. As the fluid is assumed to be both laminar and isothermal the problem is governed by the incompressible Navier-Stokes equations. For scalar equations like the convection and diffusion, and heat transfer equations axisymmetric transformation simply results in a multiplication of the equation with the radial coordinate. In this case the vector valued equations results in additional terms compared to the usual Cartesian case

$$\left\{ \begin{array}{l} r\rho \frac{\partial u}{\partial t} - r\mu(2\frac{\partial^2 u}{\partial r^2} + \frac{\partial^2 u}{\partial z^2} + \frac{\partial^2 v}{\partial r \partial z}) + r\rho(u\frac{\partial u}{\partial r} + v\frac{\partial u}{\partial z}) + r\frac{\partial p}{\partial r} + 2\mu\frac{u}{r} = 0 \\ r\rho \frac{\partial v}{\partial t} - r\mu(\frac{\partial^2 v}{\partial r^2} + \frac{\partial^2 u}{\partial z \partial r} + 2\frac{\partial^2 v}{\partial z^2}) + r\rho(u\frac{\partial v}{\partial r} + v\frac{\partial v}{\partial z}) + r\frac{\partial p}{\partial z} = 0 \\ u + r\frac{\partial u}{\partial r} + r\frac{\partial v}{\partial z} = 0 \end{array} \right.$$

In addition to modifying the equations an appropriate boundary condition for the symmetry boundary must be chosen. A homogeneous Neumann insulation/symmetry condition is typically employed for scalar equations, but in the case of fluid flow a slip condition preventing any radial velocity $u(r=0)=0$ while allowing axial velocity is appropriate.

The geometry of the problem considers a 2:1 constriction with an initial pipe diameter of 2. The inlet velocity is assumed to be uniform $v(z=0)=1$ and the fluid has a density of $\rho = 1$ and viscosity $\mu = 0.05$. This results in a laminar Reynolds number of $Re = \frac{\rho U d}{\mu} = 40$.



1.7.1 Axisymmetric Fluid Flow using the GUI

This section describes how to set up and solve the axisymmetric flow problem with the FEATool graphical user interface (GUI). Although FEATool version 1.8 and later feature predefined physics modes for axisymmetric coordinate systems, this example shows how one can use the *edit equations* feature to modify the built in equations and accommodate these transformations.


1. Start Octave/Matlab, and if you have not run the installation script (which automatically adds the FEATool directory paths at startup) then change your working directory to where your FEATool installation is, for example

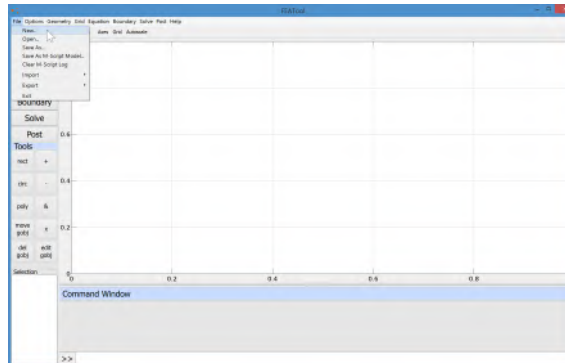
```
cd C:\featool
```

2. In the command window type

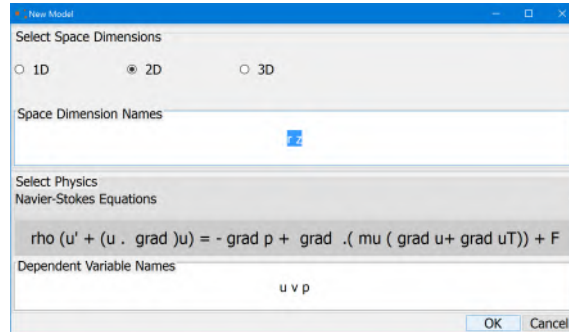
```
featool
```


to start the graphical user interface (GUI).

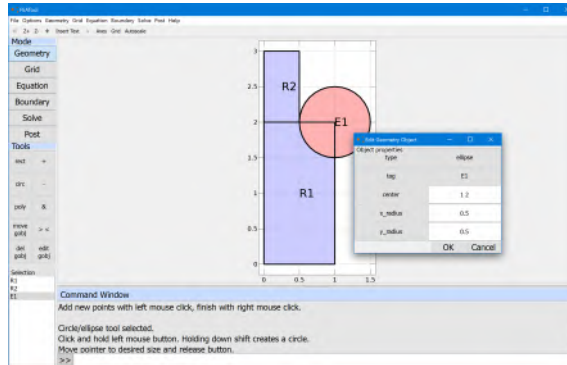
3. Either select **New...** from the **File** menu, or click on the **New Model** button  in the upper horizontal toolbar, to clear all data and start defining a new model.



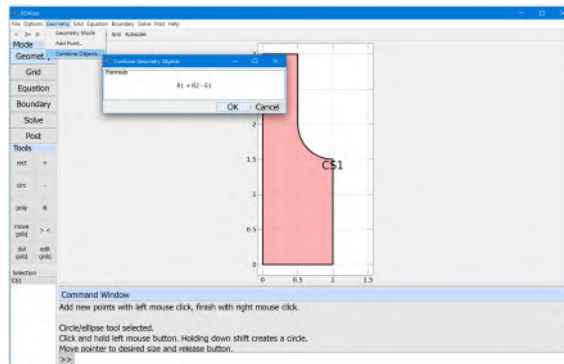
- In the opened *New Model* dialog box, click on the **2D** radio button in the *Select Space Dimensions* frame (instead of **Axi 2D** as the Cartesian 2D equations will be manually modified in this example), and select **Navier-Stokes Equations** from the *Select Physics* drop-down list. Change the space dimension names to **r** and **z** but leave the dependent variable names to their default values. Finish and close the dialog box by clicking on the **OK** button.



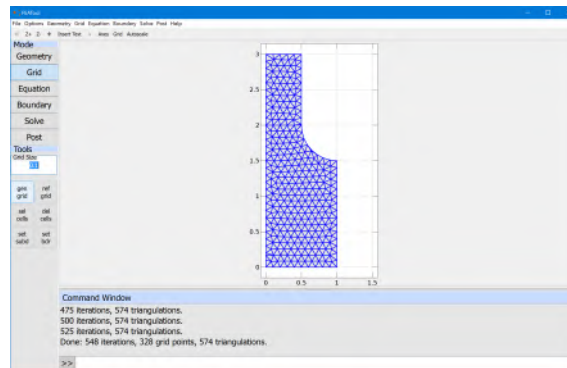
- Use the **Create square/rectangle** button  in lower left toolbar to create two rectangles. Put one on top of the other with their left edges aligned with the z-axis ($r=0$). The lower one should have dimensions 1×2 and the upper one 0.5×1 . Also create a circle with radius 0.5 centered at $(1, 2)$.



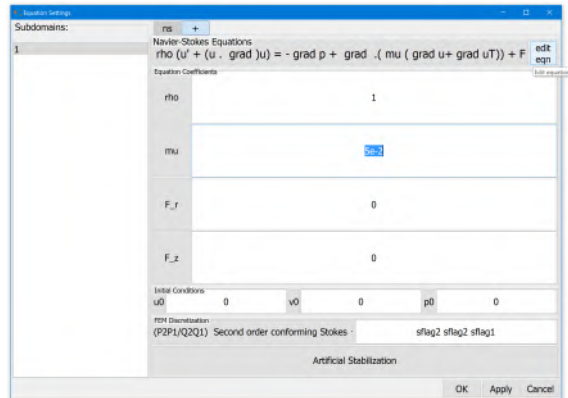
- To create the compound geometry, select **Combine Objects...** from the **Geometry** menu. Enter the formula **$R1 + R2 - E1$** in the edit field of the **Combine Geometry Objects** dialog box and press **OK**.



7. Press the **Grid** mode button in the *Mode* toolbar to switch from *geometry* mode to *grid* generation mode. To change the target grid size, enter **0.1** in the *Grid Size* edit field. Then click on the **Generate** button to call the automatic grid generation function.



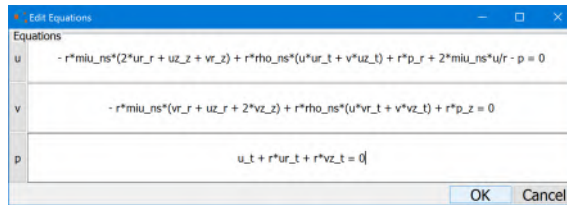
8. Change to *physics and equation/subdomain* specification mode by pressing the **Equation** button. In the *Equation Settings* dialog box that automatically opens, set the density ρ to **1**, viscosity μ to **5e-2**, and the source terms to zero.



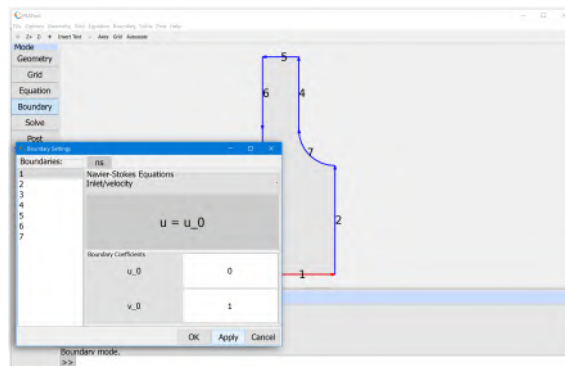
9. The equations must now be changed from a Cartesian to cylindrical coordinate system. To do this press the **edit eqn** button next to the equation description. This will bring up the **Edit Equations** dialog box and show the currently defined partial differential equations. Change the equations to the following

$$\begin{aligned}
 & - r \cdot \text{miu_ns} \cdot (2 \cdot \text{ur_r} + \text{uz_z} + \text{vr_z}) + r \cdot \text{rho_ns} \cdot (\text{u} \cdot \text{ur_t} + \text{v} \cdot \text{uz_t}) \\
 & + r \cdot \text{p_r} + 2 \cdot \text{miu_ns} \cdot \text{u/r} - \text{p} = 0 \\
 & - r \cdot \text{miu_ns} \cdot (\text{vr_r} + \text{uz_r} + 2 \cdot \text{vz_z}) + r \cdot \text{rho_ns} \cdot (\text{u} \cdot \text{vr_t} + \text{v} \cdot \text{vz_t}) \\
 & + r \cdot \text{p_z} = 0 \\
 & \text{u_t} + r \cdot \text{ur_t} + r \cdot \text{vz_t} = 0
 \end{aligned}$$

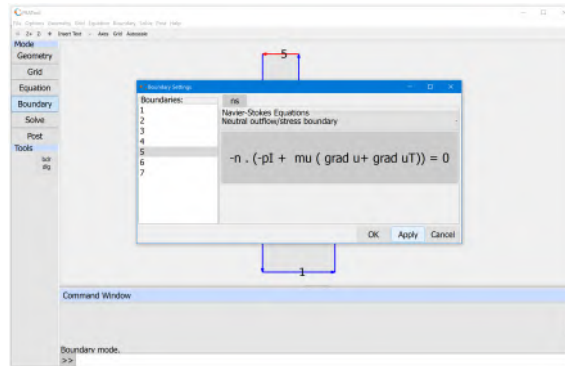
Press **OK** to finish with the equation and subdomain specifications.



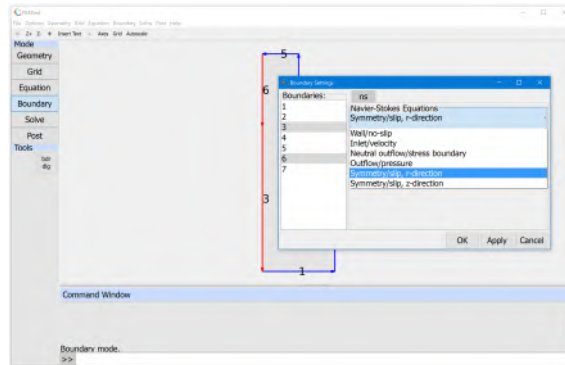
10. Switch to *boundary condition specification* mode by clicking on the Boundary mode button. In the *Boundary Settings* dialog box, first select all boundaries in the left hand side *Boundaries* list box and choose the **Wall/no-slip** boundary conditions from the drop-down list. Now select the lower inflow boundary (here number 1) in the left hand side *Boundaries* list box and choose the **Inlet/velocity** boundary condition from the drop-down list. Enter of **1** in the edit field for the z-velocity coefficient v_o .




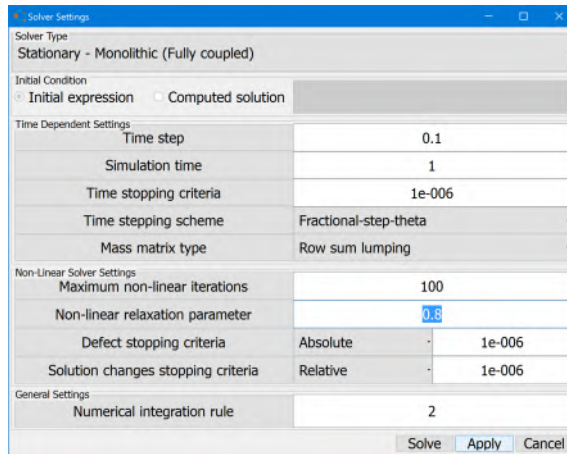
11. Select the top outflow boundary (number 5) in the left hand side *Boundaries* list box and choose the **Neutral outflow/stress boundary** boundary condition from the drop-down list (alternatively one can also prescribe a pressure p_o with the **Outflow/pressure** condition).



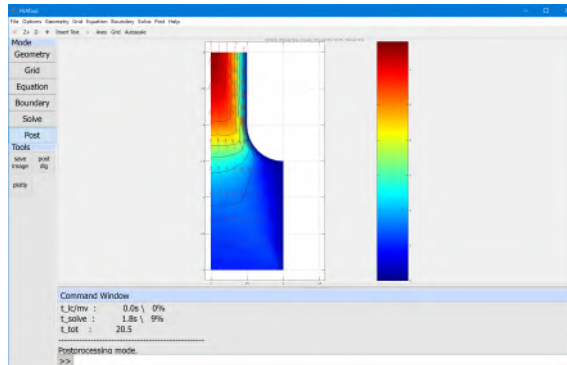
12. Lastly, select the left side boundaries (here number 3 and 6) and select the **Symmetry/slip, r-direction** boundary condition from the drop-down list which will prevent flow in the radial direction while allowing it in the axial direction. Finish by clicking the **OK** button.



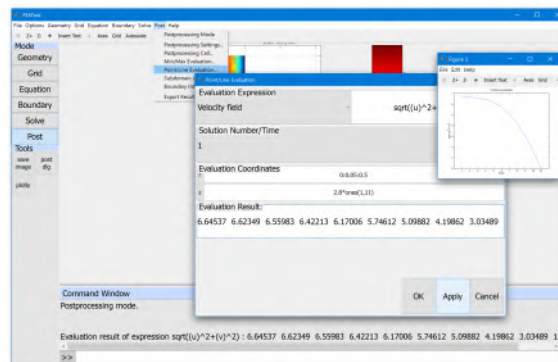
13. Now that the problem has been set up, press the **Solve** mode button to go to *solve* mode. Then press the **Solver Settings** button . Increase **Maximum non-linear iterations** to **100** and set the *Non-linear relaxation* parameter to **0.8** in the *Non-Linear Solver Settings* frame to relax the convergence of the solver. Press **Apply** and then **Solve** to start the solution process.



14. After the problem has been solved FEATool will automatically switch to *postprocessing* mode and display the computed solution. We can see that the velocity field is significantly accelerated by the pipe constriction.



15. It is also possible to study a section of the velocity profile by using the **Point/Line Evaluation...** feature from the **Post** menu (available with FEATool Multiphysics and Professional licenses). By entering a series of coordinates to examine we get both the values and a cross section plot of the evaluation expression. In this case the velocity profile close to the outlet at $z=2.8$ is starting to shift from parabolic to a more square profile indicating a higher velocity flow and we might need to study a longer outflow section to recover the expected parabolic laminar flow profile.



1.7.2 Axisymmetric Fluid Flow using the CLI

The process to set up and solve this fluid flow problem on the command line interface is illustrated in the `ex_navierstokes8` script file which can be found in the examples directory. Alternatively, one can also use the **Save As M-Script Model...** feature to get an equivalent Matlab script file for all the corresponding CLI commands that has been executed by the GUI.

1.8 Classic Equation Example - Poisson Equation with a Point Source

The classic Poisson equation is one of the most fundamental partial differential equations (PDEs). Although one of the simplest equations, it is a very good model for the process of diffusion and comes up again and again in many applications such as in fluid flow, heat transfer, and chemical transport.

This example shows how to up and solve the Poisson equation

$$d_{ts} \frac{\partial u}{\partial t} + \nabla \cdot (-D \nabla u) = f \quad (1)$$

for a scalar field $u = u(\mathbf{x})$ on a circle Ω with radius $r = 1$ in two dimensions. The diffusion coefficient $D = 1$ and right hand side source term $f = \delta(0,0)$ which prescribes a point source at the center. The Poisson problem is also considered stationary meaning the time dependent term can be neglected. With these assumptions equation (1) simplifies to

$$-\Delta u = \delta(0,0).$$

Moreover, homogeneous Dirichlet boundary conditions are prescribed on all boundaries of the domain, that is $u = 0$ on $\partial\Omega$. The exact solution for this problem is $u(x,y) = -\frac{1}{2\pi} \log(r)$ which can be used to measure the accuracy of the computed solution.

1.8.1 Poisson Equation with a Point Source using the GUI

This section describes how to set up and solve the Poisson equation (1) with the FEATool graphical user interface (GUI) which is available when using FEATool together with Octave version 4.0 or later and Matlab.


1. Start Octave/Matlab, and if you have not run the installation script (which automatically adds the FEATool directory paths at startup) then change your working directory to where your FEATool installation is, for example

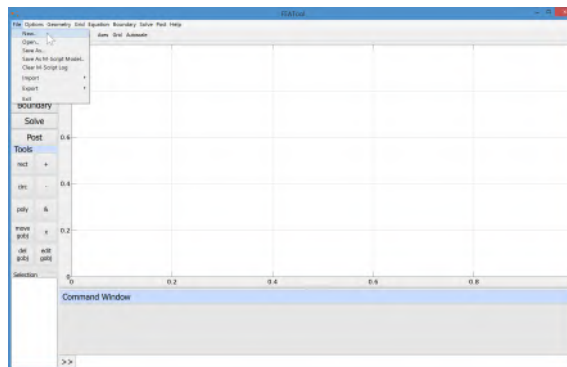

```
cd C:\featool
```

2. In the command window type

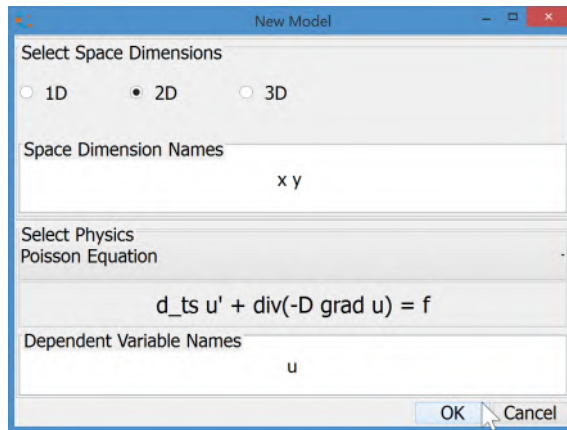
```
featool
```

to start the graphical user interface (GUI).

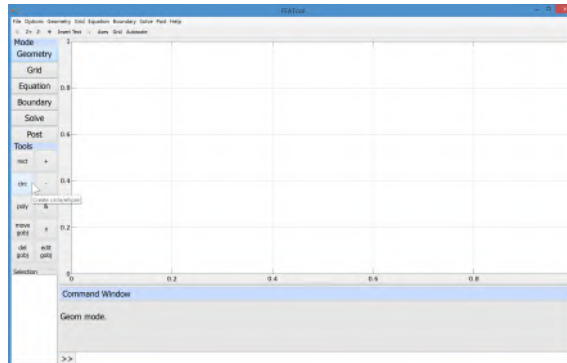
3. Either select **New...** from the **File** menu, or click on the **New Model** button  in the upper horizontal toolbar, to clear all data and start defining a new model.



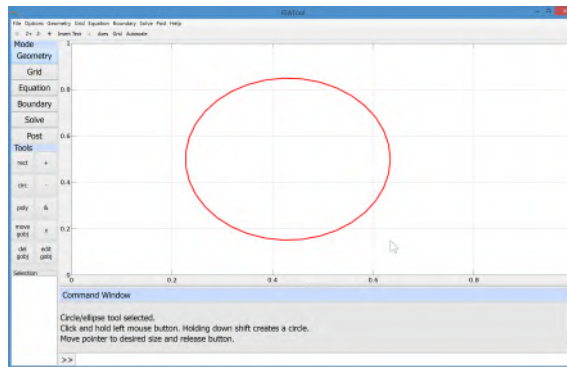
4. In the opened *New Model* dialog box, click on the **2D** radio button in the *Select Space Dimensions* frame, and select **Poisson Equation** from the *Select Physics* drop-down list. Leave the space dimension and dependent variable names to their default values. Finish and close the dialog box by clicking on the **OK** button.



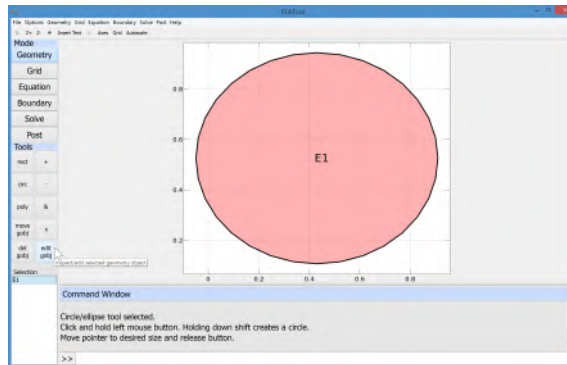
- To create a circle, first left click on the **Create circle/ellipse** button  in the left hand side *Tools* toolbar frame.



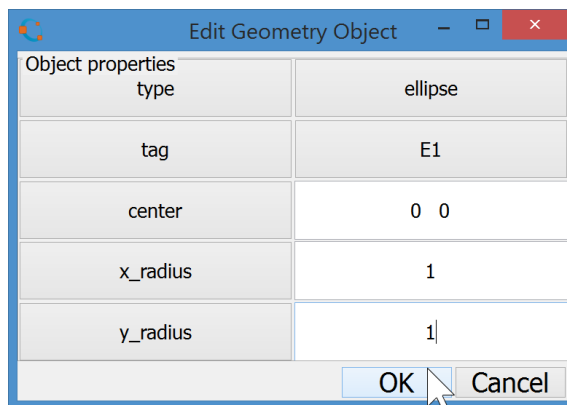
6. Then click and hold the left mouse button anywhere in the main plot axes window, and move the mouse pointer to show red outlines of a circle or ellipse. Release the button to finalize and create a solid geometry object.



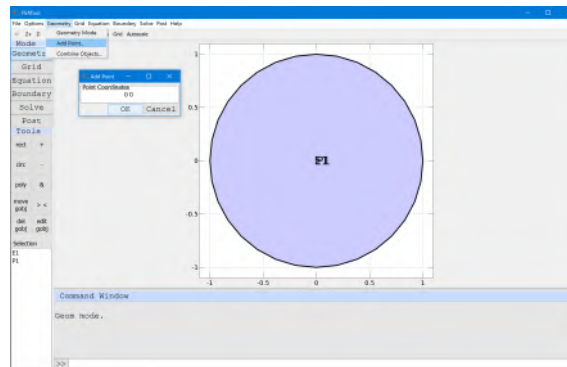
7. The object properties must be be changed to make a circle with radius 1 centered at the origin. To do this, click on the ellipse *E1* to select it which also highlights it in red (alternatively you select it by clicking on *R1* in the selection list box under the left side toolbar buttons). Then click on the **Inspect/edit selected geometry object** toolbar button



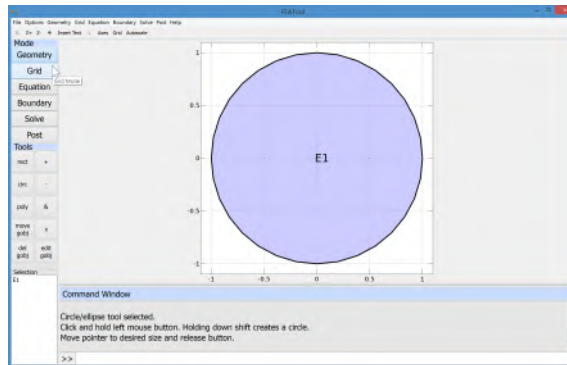
8. In the opened **Geometry Object** dialog box change the *center* coordinates edit field **0 0**, and the *x radius* and *y radius* to **1** in the corresponding fields. Finish editing the geometry object and close the dialog box by clicking **OK**.



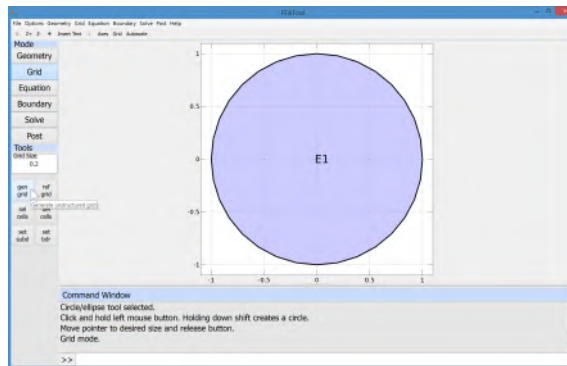
9. To define a point select **Add Point...** from the **Geometry** menu. This opens a dialog box where you can define a new point. Enter **0 0** in the *Point Coordinates* edit field and press **OK** to add the point *P1* to the center. This point will ensure that we will have a grid point in the center where constraints can be prescribed.



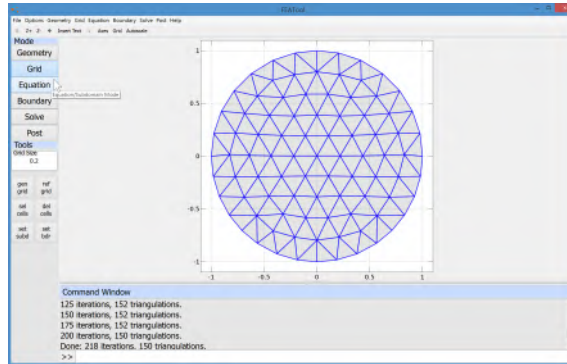
10. Press the **Grid** mode button in the *Mode* toolbar to switch from *geometry* mode to *grid* generation mode.



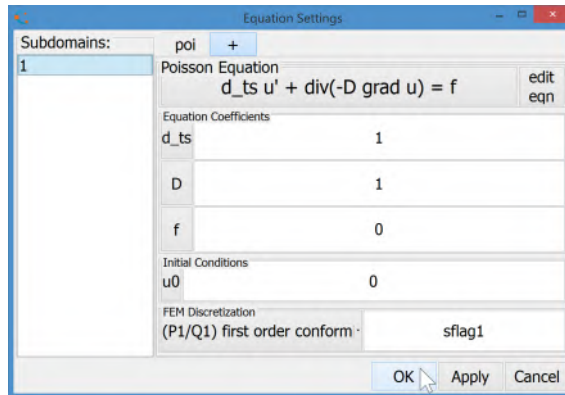
11. Click on the **Generate** button Generate to call the grid generation function which automatically generates a grid of triangles for the circle.



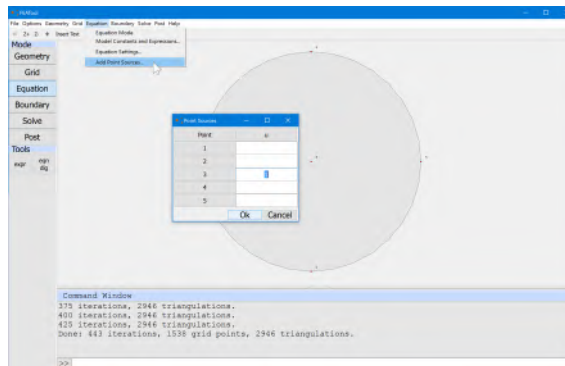
12. Press the **Equation** mode button in the *Mode* toolbar to switch from *grid* mode to *physics* and *equation/subdomain* specification mode.



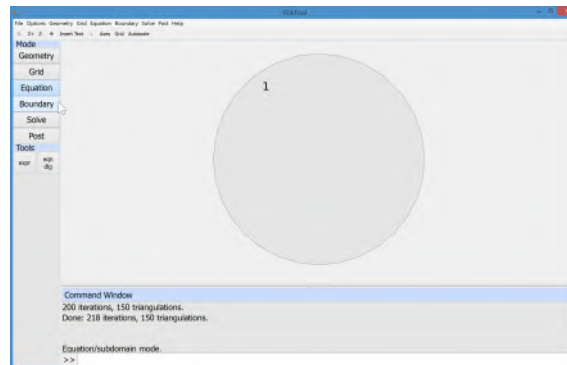
13. In the *Equation Settings* dialog box that automatically opens, set the diffusion coefficient D to **1** and source term coefficient f to **0** in the corresponding edit fields. All other coefficients can be left to their default values. Press **OK** to finish and close the dialog box.



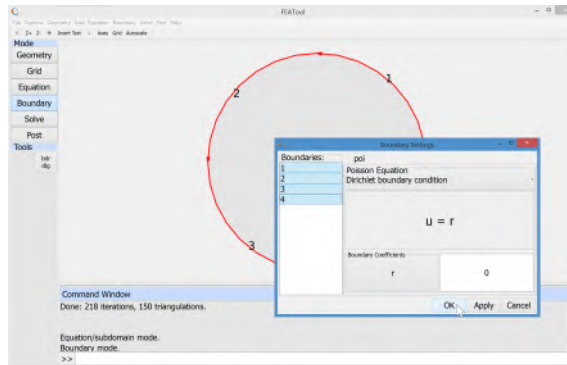
14. To add the point source select **Add Point Sources...** the from the **Equation** menu, and enter **1** in the corresponding edit field for the point in the center (in this case point number 3). Press **OK** to finish.



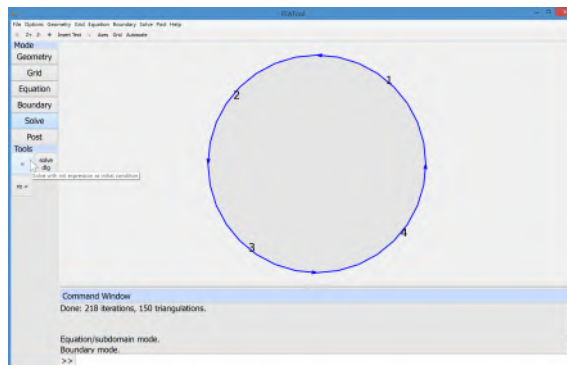
15. Switch to *boundary condition specification* mode by clicking on the Boundary mode button.




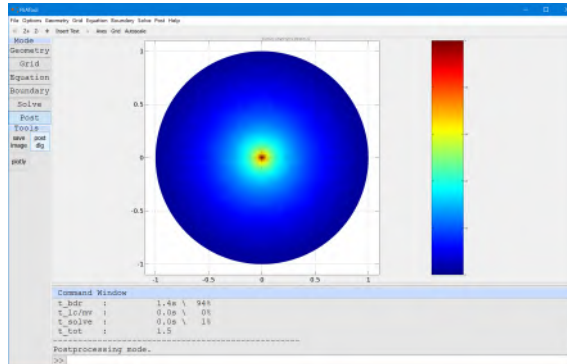
16. In the *Boundary Settings* dialog box, select all boundaries in the left hand side *Boundaries* list box and choose **Dirichlet boundary condition** in the drop-down list. Set the Dirichlet boundary coefficient r equal to **0** in the *Boundary Coefficients* frame and finish by clicking on **OK**.



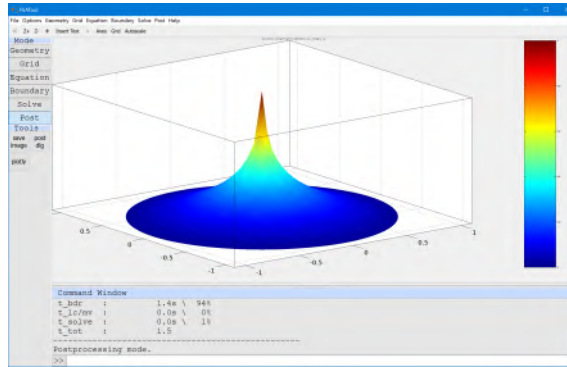
17. Now that the problem is fully specified, press the Solve mode button to switch to *solve* mode. Then press the button with an equals sign = in the *Tools* toolbar frame to call the solver with the default solver settings.



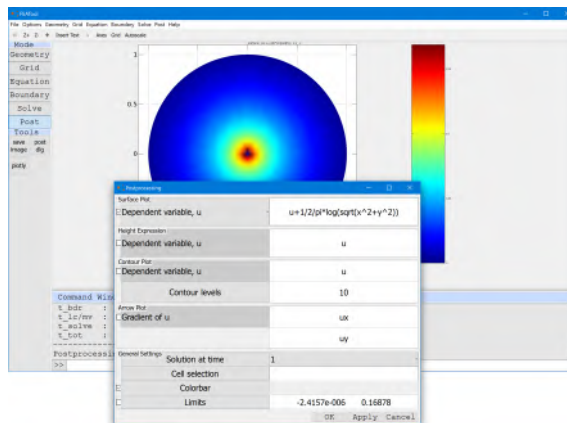
18. After the problem has been solved FEATool will automatically switch to *postprocessing* mode and display the computed solution. To change the plot, open the postprocessing settings dialog box by clicking on the **Postprocessing settings** button  in the *Tools* toolbar frame.



19. Activate *Height* plot by marking the corresponding check boxes and press **OK** or **Apply** to show how the solution looks like in three dimensions. We can clearly see how the central point source results in a spike in the solution.



20. By returning to the *Postprocessing* settings dialog box and entering the expression **$u+1/2\pi\sqrt{x^2+y^2}$** in the *Surface Plot* expression edit field it is possible to plot and visualize the difference between the computed and exact reference solution. We can see that the largest errors are found in the center due to the coarse grid. To improve the accuracy one should ideally create a grid that is locally refined around the central point.



1.8.2 Poisson Equation with a Point Source using the CLI

The process to set up and solve this Poisson problem on the command line interface is illustrated in the `ex_poisson7` script file which can be found in the examples directory.

1.9 Custom Equation Example - Wave Equation on a Circle

This section explains how to set up and solve a generalized wave equation model. The wave equation is a hyperbolic partial differential equation (PDE) of the form

$$\frac{\partial^2 u}{\partial t^2} = c\Delta u + f$$

where c is a constant defining the propagation speed of waves, and f is a source term. This equation cannot be solved as it reads due to the second order time derivative. However, the problem can be transformed by reformulating the wave equation as two coupled parabolic PDEs, that is

$$\begin{cases} \frac{\partial u}{\partial t} = v \\ \frac{\partial v}{\partial t} = c\Delta u + f \end{cases}$$

This dual coupled problem can easily be implemented in FEATool with the custom equation feature. An example of the wave equation on a unit circle, with zero boundary conditions, constant $c = 1$, source term $f = 0$, and initial condition $u(t = 0) = 1 - (x^2 + y^2)$ is described in the following

1.9.1 Wave Equation using the GUI

This section describes how to set up and solve the wave equation with the FEATool graphical user interface (GUI) which is available when using FEATool together with Octave version 4.0 or later and Matlab.


1. Start Octave/Matlab, and if you have not run the installation script (which automatically adds the FEATool directory paths at startup) then change your working directory to where your FEATool installation is, for example

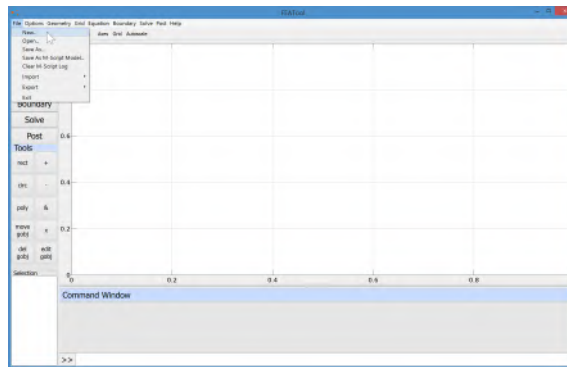
```
cd C:\featool
```

2. In the command window type

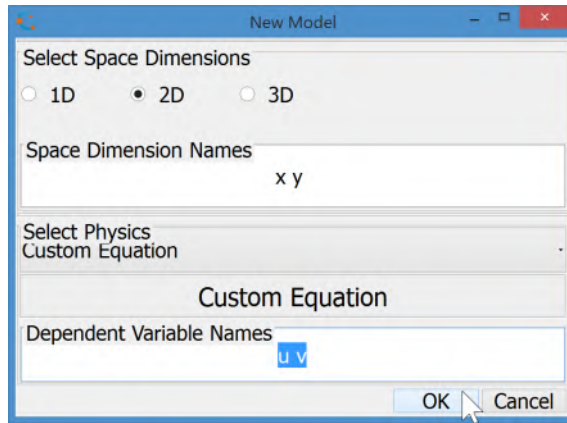
```
featoool
```

to start the graphical user interface (GUI).

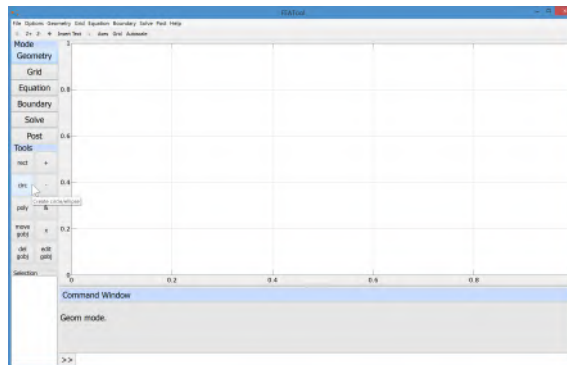
3. Click on the **New Problem** button  in the upper horizontal toolbar to clear all data and start defining a new problem.



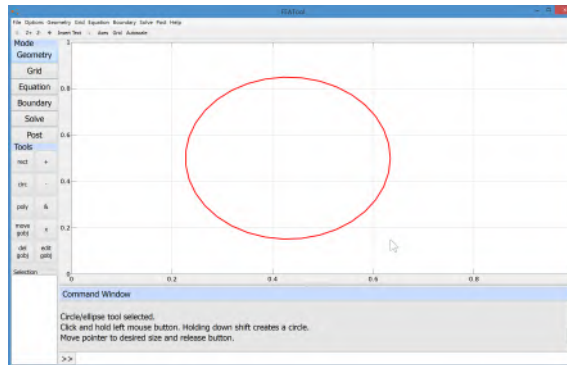
4. In the opened *New Problem* dialog box, click on the **2D** radio button in the *Select Space Dimensions* frame, and select **Custom Equation** from the *Select Physics* drop-down list. Leave the space dimension as it is but change the dependent variable names to **u** **v** (The custom equation physics mode allows for entering an arbitrary number of dependent variables through the use of a space separated list). This will add two equations for u and v , respectively. Finish and close the dialog box by clicking on the **OK** button.




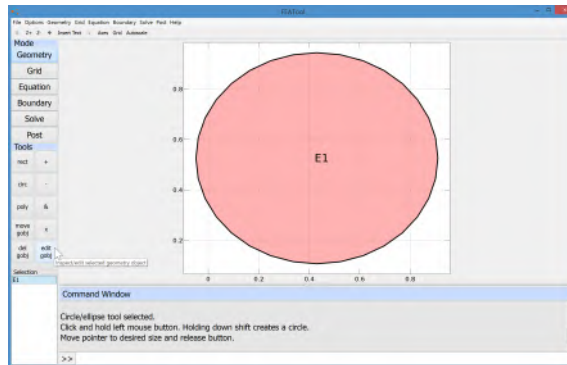
5. To create a circle, first left click on the **Create circle/ellipse** button  in the left hand side *Tools* toolbar frame.



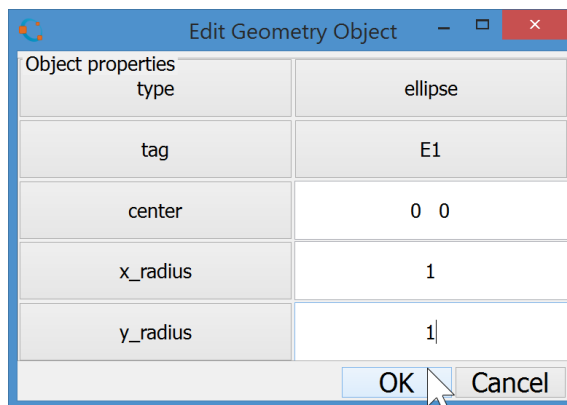
6. Then click and hold the left mouse button anywhere in the main plot axes window, and move the mouse pointer to show red outlines of a circle or ellipse. Release the button to finalize and create a solid geometry object.



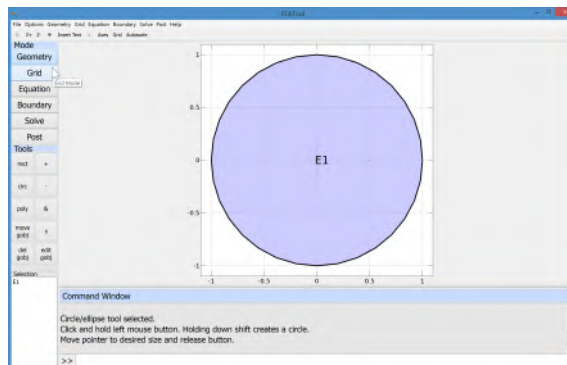
7. The object properties must be changed to make a circle with radius 1 centered at the origin. To do this, click on the ellipse *E1* to select it which also highlights it in red. (Alternatively you select it by clicking on *R1* in the selection list box under the left side toolbar buttons.) Then click on the **Inspect/edit selected geometry object** toolbar button 



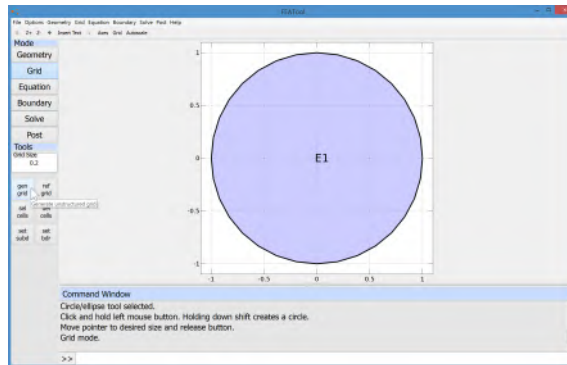
8. In the opened **Geometry Object** dialog box change the *center* coordinates edit field **0 0**, and the *x radius* and *y radius* to **1** in the corresponding fields. Finish editing the geometry object and close the dialog box by clicking **OK**.



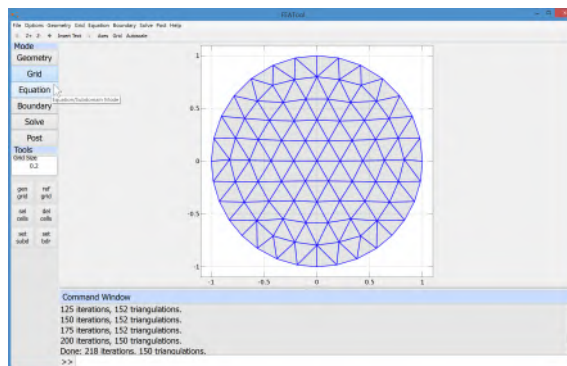
9. Press the **Grid** mode button in the *Mode* toolbar to switch from *geometry* mode to *grid* generation mode.



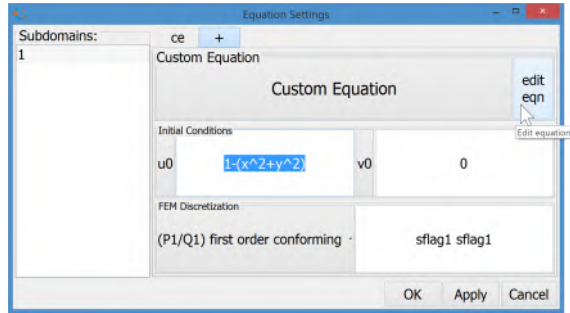
10. Click on the **Generate** button **Generate** to call the grid generation function which automatically generates a grid of triangles for the circle.



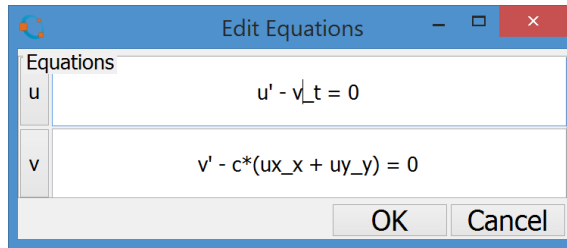
- Press the **Equation** mode button in the *Mode* toolbar to switch from *grid* mode to *physics* and *equation/subdomain* specification mode.




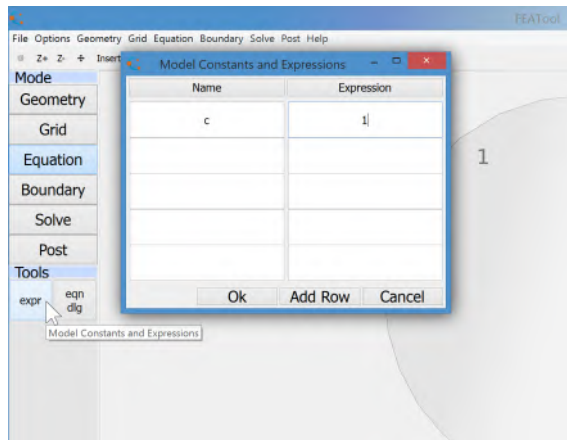
12. An *Equation Settings* dialog box will now automatically open. Set the initial condition for u , u_0 to $1-(x^2+y^2)$. Then click on the **edit eqn** button.



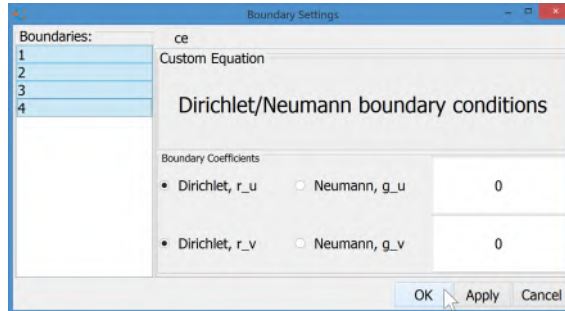
13. In the *Edit Equations* dialog box enter the equations $\mathbf{u}' - \mathbf{v}_t = \mathbf{0}$ and $\mathbf{v}' + \mathbf{c}*(\mathbf{u}_x\mathbf{x} + \mathbf{u}_y\mathbf{y}) = \mathbf{0}$ in the corresponding edit fields for u and v . Here \mathbf{u} and \mathbf{v} are the dependent variables, \mathbf{u}'/\mathbf{v}' denote the corresponding time derivative, and an underscore will treat it implicitly in the weak finite element formulation (for example \mathbf{v}_t corresponds to v multiplied with the test function for u , and $\mathbf{u}_x\mathbf{x}$ is analogous to $du/dx * dv_t/dx$). Note, that the first equation could also be implemented as $\mathbf{u}' = \mathbf{v}$ but then v would be evaluated explicitly in the right hand side, and by transferring it to the implicit left hand side matrix we will get a linear problem which is more efficient to solve. Press **OK** and close the equation settings dialog boxes.



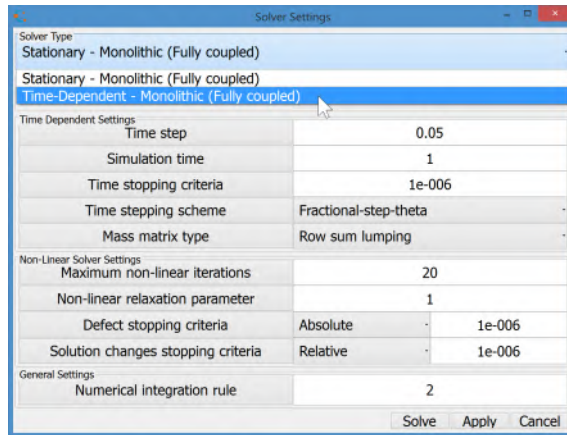
- Click on the **Model Constants and Expressions** button  and enter a new constant named **c** with value **1** (This is the constant used in diffusion term of the v equation). Press **OK** to finish.



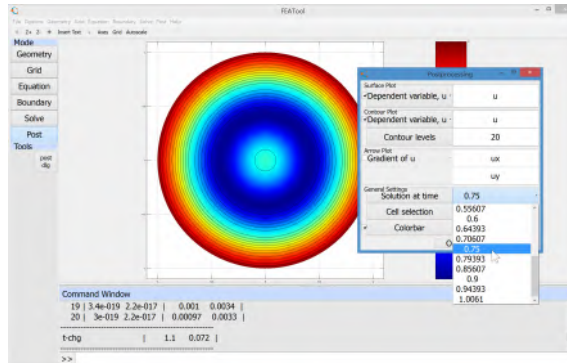
15. Switch to **Boundary** condition specification mode and set Dirichlet conditions with prescribed value **0** on all boundaries.



16. Change to **Solve** mode and open the **Solver Settings** dialog box. Choose the **Time-Dependent** solver and also set the time step to **0.05**. Press **Apply** and **Solve** to start the solution process.



17. Once the solver has finished the solution at the final time step will be displayed. In the **Postprocessing Settings** dialog box it is also possible to select and visualize the solution at different times.



1.9.2 Wave Equation using the CLI

The process to set up and solve the wave equation problem on the command line interface is illustrated in the `ex_waveequation1` script file which can be found in the examples directory.

1.10 More Examples and Information

Additional m-script command line examples and can also be found in the [featool/examples](#) directory. Also bookmark the [FEATool news](#) and [blog](#) page where tips and tutorials are posted.

