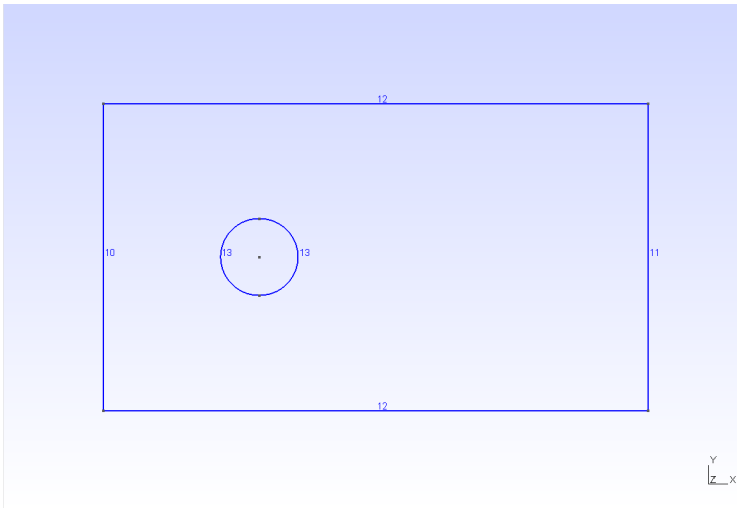
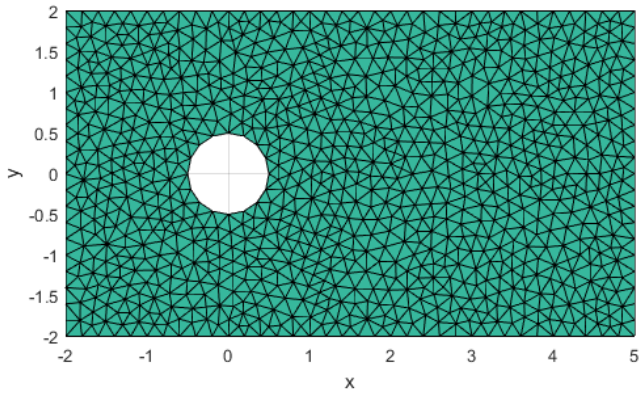
QuickerSim CFD Toolbox Tutorial 1: Meshing, Basic N-S Solver and Postprocessing

Geometry & Mesh Generation

Lite version of our toolbox importss meshes prepared with the opne source Gmsh mesh generator. Go to Gmsh download and geometry creation and meshing tutorial.
After geometry and mesh have been created, you should end up with the following geometry (note the ids of physical lines) and the mesh shown below. You can also refer to files cylinderGeometry.geo and cylinder.msh which are attached to this tutorial.





Solving a Navire-Stokes Problem

In this section we show code of a complete Navier-Stokes solver, which we advice always to use as a starting point when doing any flow simulation. You can later modify this code according to your needs.
This QuickerSim CFD Toolbox for MATLAB® solver is incompressible, laminar, fluid flow governed by the Navier-Stokes equations, i.e.:

$$(\vec{u} \cdot \nabla)\vec{u} = -\nabla p + \nu \Delta \vec{u} + \vec{f}$$

Alternatively you can solve a Stokes problem define by:

$$-\nabla p + \nu \Delta \vec{u} + \vec{f} = 0$$

subject to

$$\nabla \cdot \vec{u} = 0$$

supposed that you will use `assembleStokeMatrix2D` function in place of `assembleNavierStokesMatrix2D` function.
You have to apply proper boundary conditions to above systems that will guarantee a unique solution to the stationary Stokes or Navier-Stokes problem. You can use an 'inlet' boundary condition for which you need to specify two velocity components (not necessarilly perpendicular to the inlet), 'wall' boundary condition, which sets both velocity components to zero or 'slipAlongY' boundary conditions which imply zero shear stress at the given boundary. If you do not apply explicitly any boundary condition, the toolbox will assue this boundary to be open boundary (i.e. outflow) where the following boundary condition will be applied:

$$-\nu \nabla \vec{u} \cdot \vec{u} + p = 0$$

Code

To start with lets clear respectively MATLAB Comment Window and Workspace.

```
clc;
clear;
```

Read and display mesh

Import mesh generated by Gmsh; `p,e,t` arrays store the computational mesh. Type: `help imporMeshGmshi`n Command Window to get more details.
You can now display mesh to check, if everything is alright. `displayMesh2D(p,t);`

Convert mesh to second order

For a fluid flow simulation we always need to convert mesh to second order. nVnodes stands for number of velocity nodes in the mesh, nPnodes number of pressure nodes and indices is a Matlab structure which will help us to refer to solution fields (x-velocity, y-velocity and pressure), since all of them will be stored in one solution vector u. Indices will enable us easy access.

```
[p,e,t,nVnodes,nPnodes,indices] =
    convertMeshToSecondOrder(p,e,t);
```

Define fluid

We define kinematic viscosity of the fluid. We never specify dynamic viscosity. The Toolbox solves only incompressible flows, assuming constant density, which is nowhere introduced to equations. In other words density always equals 1. If you simulate e.g. water flow with real density of $1000\frac{kg}{m^3}$, your computed pressure field will be actually real pressure field divided by fluid density. So, if you want to obtain real static pressure in Pascals, you would need to multiply computed values of pressure by 1000.

```
nu = 0.01;
```

For our geometry and velocity this implies Re = 100

Initiate solution and convergence criteria

Now, define some initial approximation to the solution and define convergence criteria, i.e. maximum number of iterations to solve for the nonlinearity in convection term and maximum residual value. These are absolute residuals. We set initial velocity field to [1 0] everywhere and initial pressure field to 0 everywhere. Convergence is an additional array which wonderfully helps plotting convergence plots and checking for breaking the iteration loop.

```
[u, convergence] = initSolution(p,t,[1 0],0);
```

We require absolute (not a scaled) residuals to drop to 1e-3, but in any case we don't allow for more than 25 iterations.

```
maxres = 1e-3;
maxiter = 25;
```

Iterate nonlinear terms

```
for iter = 1:maxiter
```

Assemble Navier-Stokes matrix and right-hand side vector u(indices.indu) accesses x-velocities in the solution vector u(indices.indv) accesses y-velocities in the solution vector 'nosupg' flag tells that we do not use any kind of SUPG stabilization.

```
[NS,F]=assembleNavierStokesMatrix2D(p,e,t,nu,u(indices
    .indu),u(indices.indv),'nosupg');
```

Apply boundary conditions

Inlet at wall 10 with velocity set to [1 0]

```
[NS,F]=imposeCfdBoundaryCondition2D(p,e,t,NS,F,10,'
    inlet',[1 0]);
```

Free slip along x axis on channel walls (wall id = 12)

```
[NS,F]=imposeCfdBoundaryCondition2D(p,e,t,NS,F,12,'
    slipAlongX');
```

No-slip wall boundary condition at cylinder walls (id = 13)

```
[NS,F]=imposeCfdBoundaryCondition2D(p,e,t,NS,F,13,'
    wall');
```

Do nothig for outflow at boundary with id = 11 Compute and plot actual residuals

```
[stop,convergence]=computeResiduals(NS,F,u,size(p),
    convergence,maxres);
```

Plot residuals to figure(2)

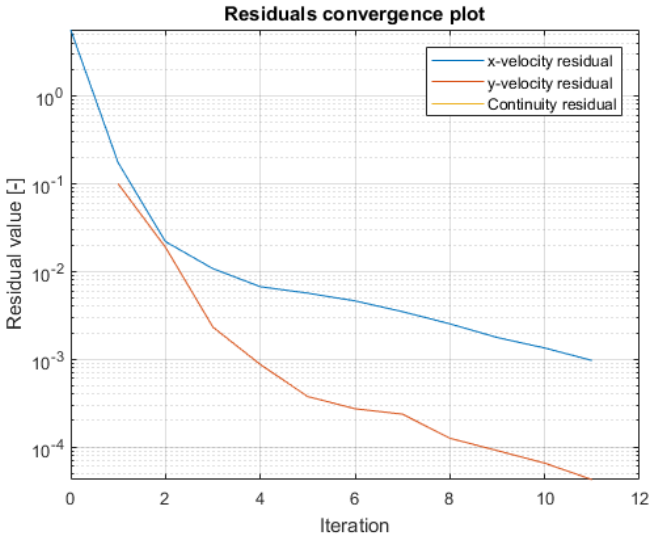
```
plotResiduals(convergence,2);
```

Break if solution converged

```
if(stop)
    break;
end
```

Solve equations

```
u = NS\F;
end
```



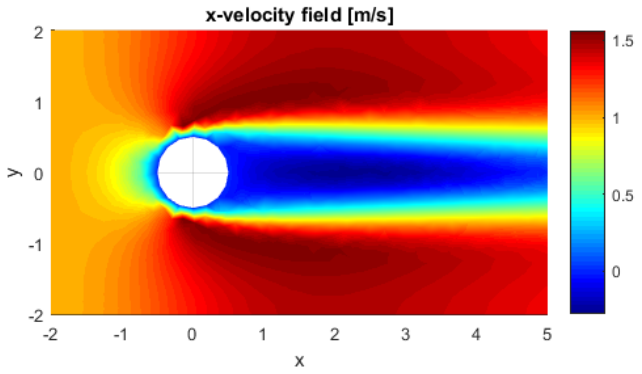
Displaying and Exporting Solution Fields

Generate pressure data in all mesh nodes (of the second order mesh - not only in nodes of the first order mesh, where it is actually solved).

```
pressure = generatePressureData(u,p,t);
```

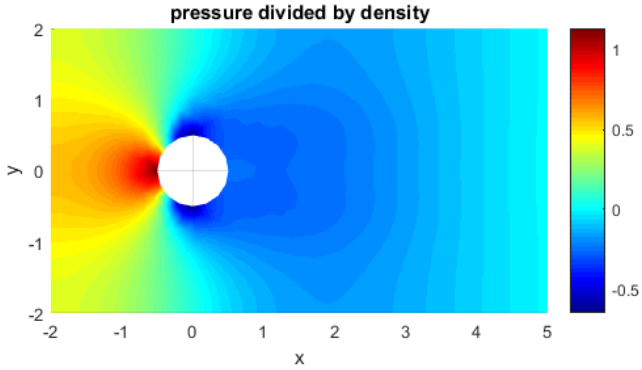
Display x-velocity to figure(3)

```
figure(3)
displaySolution2D(p,t,u(indices.indu),'x-velocity
    field [m/s]')
```



Display pressure field to figure(4)

```
figure(4)
displaySolution2D(p,t,u(indices.indp),'pressure
    divided by density')
```



Export velocity vector field to Gmsh

```
exportToGmsh2D('velocityVector.msh',[u(indices.indu);
    u(indices.indv)],p,t 'velocity_vector');
```

```
exportToGmsh2D('velocity_y_component.msh',u(indices.
    indv),p,t,'y-velocity');
exportToGmsh2D('pressure.msh',pressure,p,t,'pressure
    field');
```