Spatial Mesh Convergence Analysis

August Frechette and Zbigniew J. Kabala
6 Ocotber 2022

Overview

- This analysis is completed for the simulation of flow through the idealized dead-end pore (DEP) geometry and for the purpose of analyzing the concept of mobile and immobile porosity.
- In this mesh convergence analysis, we use the comparison of sequential solutions (i.e., the difference between each numerical solution is determined) to identify suitable mesh parameters. It it important to note that this analysis is only semi-quantitative, and that the outcome of this analysis is based on the discretion of the investigator.

System Initialization

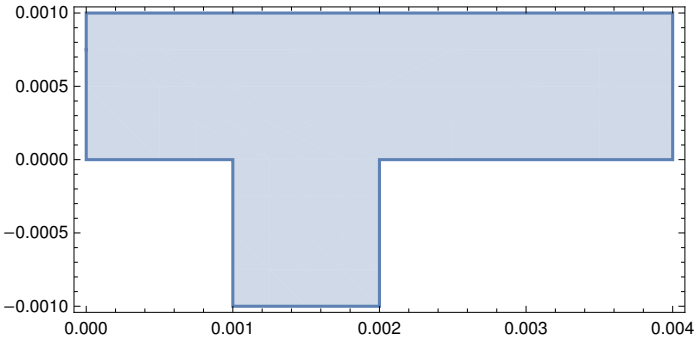
In this chapter, we define the non-dimensional flow solver used to generate the results in the *Mesh Refinement* chapter.

Solver Domain

Pictured below is the geometry that corresponds to the idealized DEP.

In[•]:=
```
height = 1. * 10.^-3; (*m*)
length = 4. * height;(*m*)
DEP = RegionUnion[Rectangle[{0., 0.}, {length, height}],
    Rectangle[{height, 0}, {2. * height, -height}]];
RegionPlot[Region[DEP], AspectRatio → Automatic]
```

Out[•]=

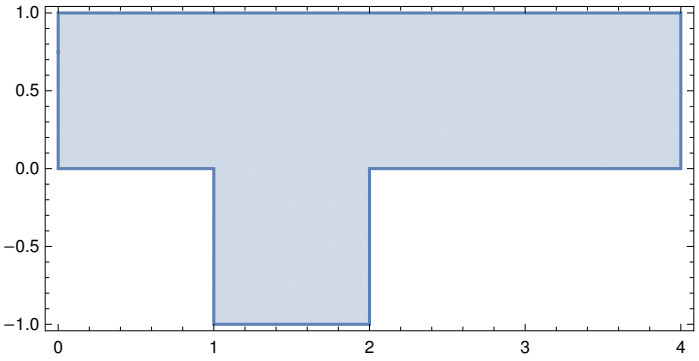


Note: *these are the dimensions utilized by (Kahler and Kabala, 2016).*

Non-dimensional formulation:

In[•]:=
```
DEPnd = RegionUnion[Rectangle[{0., 0.}, {length, height} / height],
    Rectangle[{height, 0.} / height, {2. * height, -height} / height]];
RegionPlot[Region[DEPnd], AspectRatio → Automatic]
```

Out[•]=



Solver Framework

Flow parameters

In[•]:=
```
U = 10. * 10^-3; (*m/s*)
μ = 1.23 * 10^-3;(*kg m^-1 s^-1*)
ρ = 9.99 * 10^2;(*kg m^-3*)
RE = ρ U height / μ;
```

Note: *these are the parameters utilized by (Kahler and Kabala, 2016).*

Governing equations (i.e., NS equations for incompressible, 2D, steady-state flow)

```
In[•]:= OP = {{u[x, y], v[x, y]}}.Inactive[Grad][u[x, y], {x, y}] +
           D[p[x, y], x] - RE^-1 * Laplacian[u[x, y], {x, y}],

         {{u[x, y], v[x, y]}}.Inactive[Grad][v[x, y], {x, y}] +
           D[p[x, y], y] - RE^-1 * Laplacian[v[x, y], {x, y}],

         D[u[x, y], x] + D[v[x, y], y]};

        EQs = OP == {0., 0., 0.};
```

Boundary conditions

```
In[•]:= PDiff = -12 * μ * U / height^2; (*N/m^3*)
        PDiffND = height / (ρ U^2) * PDiff; (*1*)

        bcWalls = DirichletCondition[{u[x, y] == 0., v[x, y] == 0.}, x > 0. && x < length / height];

        bcVelocity = DirichletCondition[{u[x, y] == RE/2 PDiffND (y^2 - y), v[x, y] == 0.}, x == 0.];

        bcOutlet = DirichletCondition[p[x, y] == 0., x == length / height];
        BCs = {bcVelocity, bcWalls, bcOutlet};
```

Mesh Refinement

This is a two-part procedure. We need to validate the coarse mesh size within the center of the domain (i.e., internal mesh refinement), and then refine the mesh size at the boundary of the domain (i.e., boundary mesh refinement).

```
In[•]:= Needs["NDSolve`FEM`"]
```

Note: *this package is required for mesh generation.*

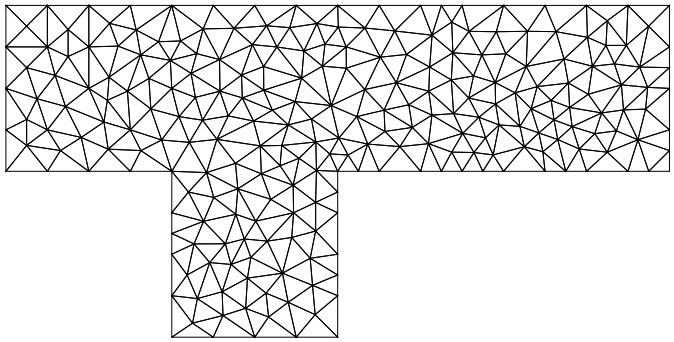
Internal Mesh Refinement

The default solver mesh for the geometry outlined in the previous chapter is generated with a coarse grid.

In[•]:=
```
Dmesh = ToElementMesh[DEPnd, MeshQualityGoal → "Maximal"]
Dmesh["Wireframe"]
```

Out[•]=
```
ElementMesh[{{0., 4.}, {-1., 1.}}, {TriangleElement[<427>]}]
```

Out[•]=



Note: the *MeshQualityGoal* → *"Maximal"* option enforces that the mesh is automatically generated to have the highest possible quality.

This mesh is described by 427 elements within the following size range:

In[•]:=
```
{Max[Dmesh["MeshElementMeasure"]], Min[Dmesh["MeshElementMeasure"]]}
```

Out[•]=
```
{0.0198244, 0.00546562}
```

The average element size can be approximated by the domain area and the number of mesh elements:

In[•]:=
```
N[5 / 427]
```

Out[•]=
```
0.0117096
```

And the mesh order is:

In[•]:=
```
Dmesh["MeshOrder"]
```

Out[•]=
```
2
```

This mesh order provides more accurate numerical solutions over a mesh order of 1 and utilizes nodes along the midpoints of mesh elements (rather than just at the vertices).

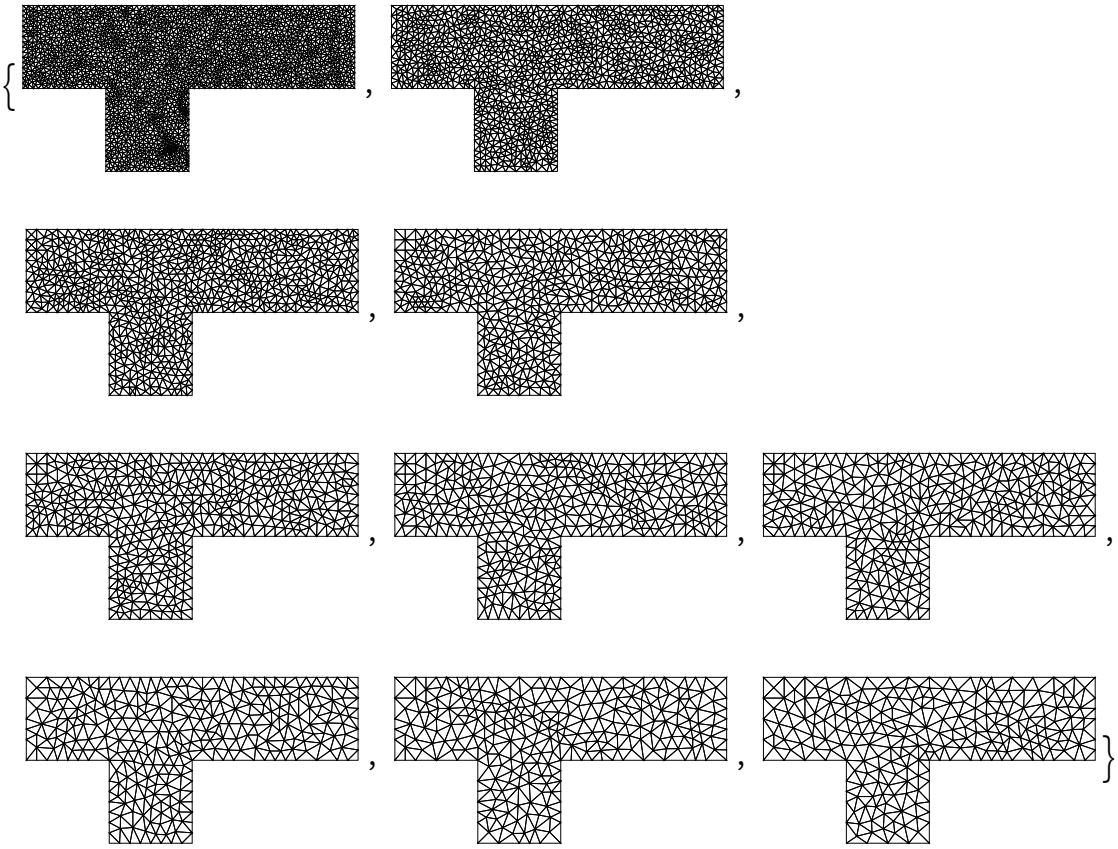
Mesh Generation

In order to ensure proper resolution of the flow domain, we should refine the default mesh generated by Mathematica.

The default *"MaxCellMeasure"* is 0.02, as exemplified by the maximum element size, provided above. To refine the mesh, we will reduce this value by 0.002 until we have reduced the *"MaxCellMeasure"* by one full order of magnitude.

In[•]:=
```
Dmesh = Table[ToElementMesh[DEPnd, MaxCellMeasure → n,
    MeshQualityGoal → "Maximal"], {n, 0.002, 0.02, 0.002}];
Table[Dmesh[[n]]["Wireframe"], {n, 1, 10, 1}]
```

Out[•]=



Solution Generation

We now have 10 meshes over which we can solve the NS equations.

In[•]:=
```
Length[Dmesh]
```

Out[•]=
```
10
```

In[•]:=
```
Clear[u, v, p, x, y]
```

```
results = Table[NDSolveValue[{EQs, BCs}, {u, v, p}, {x, y} ∈ Dmesh[[11 - n]],
    Method → {"FiniteElement", "InterpolationOrder" → {u → 2, v → 2, p → 1},
      "MeshOptions" → {AccuracyGoal → 16, PrecisionGoal → 16}}], {n, 1, 10, 1}];

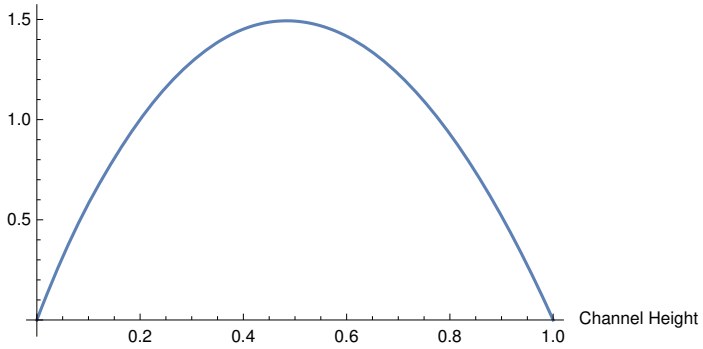
xVel = results[[All, 1]];
yVel = results[[All, 2]];
pressure = results[[All, 3]];
```

Convergence Analysis

Let's start by visualizing the velocity profile at $y = 0.75$, right before the location of the poorly-connected pore.

In[•]:=
```
Plot[xVel[[1]][0.75, y], {y, 0, 1}, AxesLabel → {"Channel Height"}]
```

Out[•]=

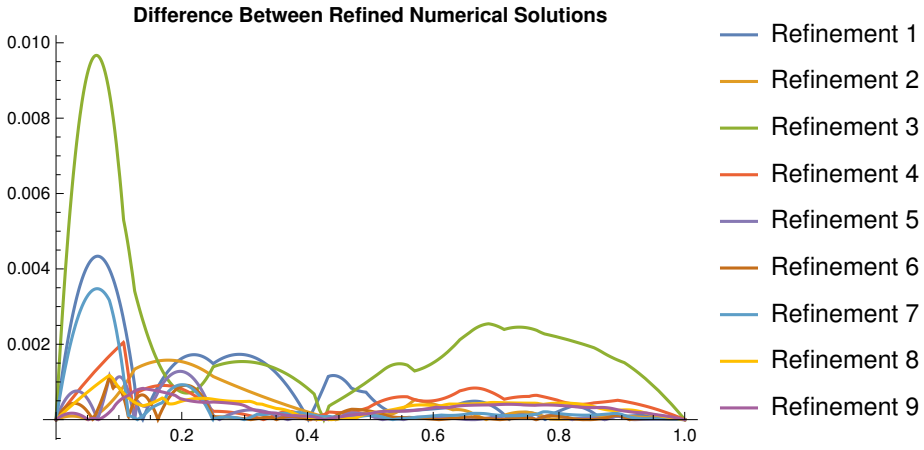


As expected, even with the default mesh, we obtain the Poiseuille velocity profile.

Let's see how our velocity profile changes with mesh refinements at this location.

In[•]:=
```
Plot[Evaluate[Table[Abs[xVel[[i]][0.75, y] - xVel[[i + 1]][0.75, y]], {i, 1, 9, 1}]], {y, 0, 1},
 PlotRange → All, PlotLabel → "Difference Between Refined Numerical Solutions",
 PlotLegends → Table[StringForm["Refinement `1`", i], {i, 1, 9, 1}]]
```

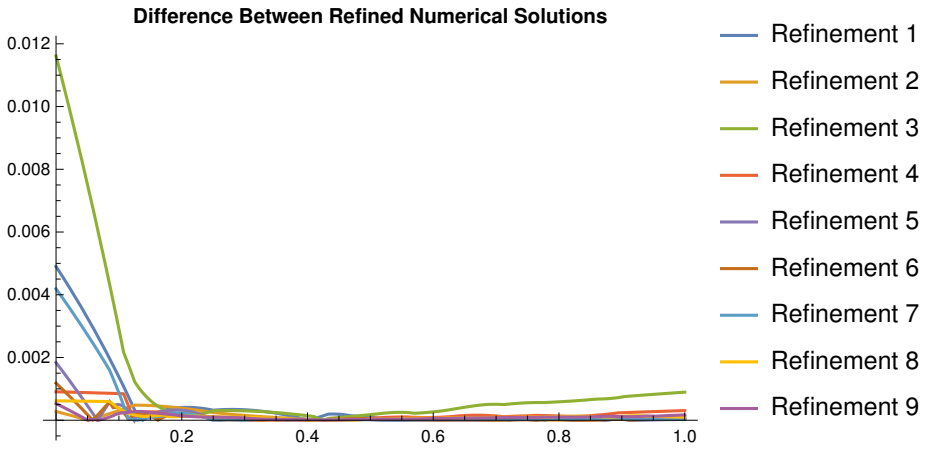
Out[•]=



We can also look at the % difference between solutions.

In[•]:=
```
Plot[Evaluate[
  Table[Abs[xVel[[i]][0.75, y] - xVel[[i + 1]][0.75, y]]/(xVel[[i]][0.75, y] + xVel[[i + 1]][0.75, y])/2,
   {i, 1, 9, 1}]], {y, 0, 1}, PlotRange → All,
 PlotLabel → "Difference Between Refined Numerical Solutions",
 PlotLegends → Table[StringForm["Refinement `1`", i], {i, 1, 9, 1}]]
```

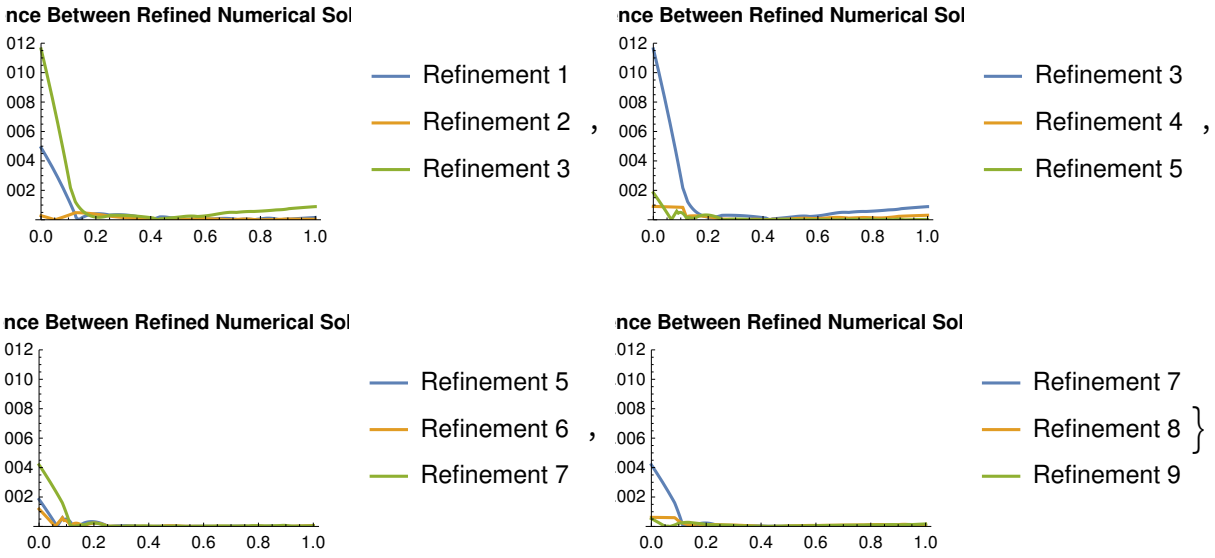
Out[•]=



Let's break this graph down into subgraphs to improve readability.

In[•]:=
```
Table[Plot[Evaluate[Table[
    Abs[xVel[[i]][0.75, y] - xVel[[i + 1]][0.75, y]]/(xVel[[i]][0.75, y] + xVel[[i + 1]][0.75, y])/2,
    {i, n, n + 2, 1}]], {y, 0, 1}, PlotRange → {0, 0.012},
  PlotLabel → "Difference Between Refined Numerical Solutions",
  PlotLegends → Table[StringForm["Refinement `1`", i], {i, n, n + 2, 1}]], {n, 1, 7, 2}]
```

Out[•]=

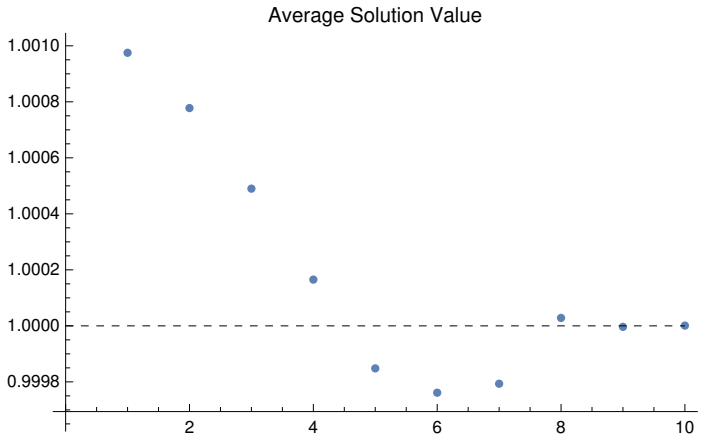


It appears that, from the above graphs, at least three refinements to the default mesh are necessary if our aim is to arrive at a stable solution.

Next, let's determine how the average solution value changes with each additional refinement.

In[•]:=
```
ASVdata = Table[NIntegrate[xVel[[n]][0.75, y], {y, 0, 1}], {n, 1, 10, 1}];
ListPlot[ASVdata, PlotLabel → "Average Solution Value",
 Epilog → {Dashed, Line[{{0, 1}, {10, 1}}]}]
```

Out[•]=

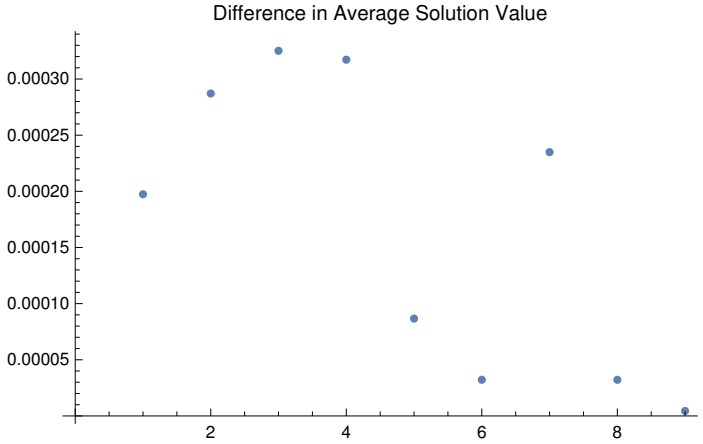


The above graph suggests that at least four refinements are necessary to arrive at a (semi) stable solution.

We could also determine the difference in the average solution value.

In[•]:=
```
Off[NIntegrate::ncvb]
Off[NIntegrate::slwcon]
ASVdiffdata =
  Table[Abs[NIntegrate[xVel[[n]][0.75, y] - xVel[[n + 1]][0.75, y], {y, 0, 1}]], {n, 1, 9, 1}];
ListPlot[ASVdiffdata, PlotLabel → "Difference in Average Solution Value"]
```

Out[•]=



Again, aside from the outlier value produced from the difference in the 7th and the 8th mesh refinements, it appears that four refinements to the default mesh are necessary.

Although it appears that the average value of our solution changes dramatically in the plot above, we

should keep the magnitude of the solution value in mind. For example, the largest difference in solution value in this plot is 0.0003:

In[•]:=
```
Max[ASVdiffdata]
```

Out[•]=
```
0.000325146
```

Relative to the actual solution value, this difference (0.03%) is insignificant

Analysis Conclusion

We can proceed one of two ways:

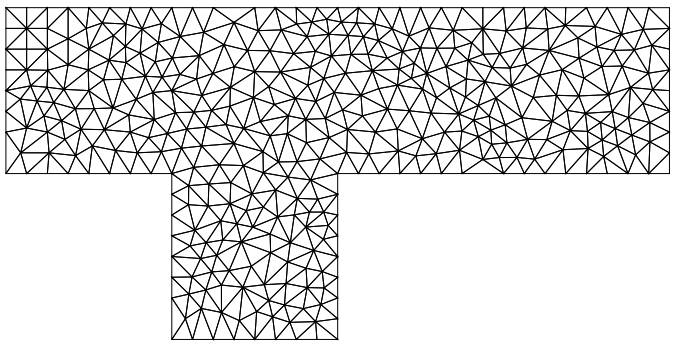
1. Use the coarsest mesh that results in the *most accurate* solution (i.e., the mesh that gives an average solution value of 1)
2. Use the coarsest mesh that *most accurately approximates* the solution (i.e., the mesh that gives a value *close* to 1)

If we go with the latter option, and **choose to proceed with *the fourth refinement*,** we can argue that additional refinements do not change the solution value enough to necessitate the increased computational demand. Otherwise, we would need to process with the eighth refinement.

The details of our interior mesh are as follows:

In[•]:=
```
Dmesh[[6]]["Wireframe"]
```

Out[•]=



In[•]:=
```
Dmesh[[6]]
```

Out[•]=
```
ElementMesh[{{0., 4.}, {-1., 1.}}, {TriangleElement[<698>]}]
```

With maximum and minimum element sizes:

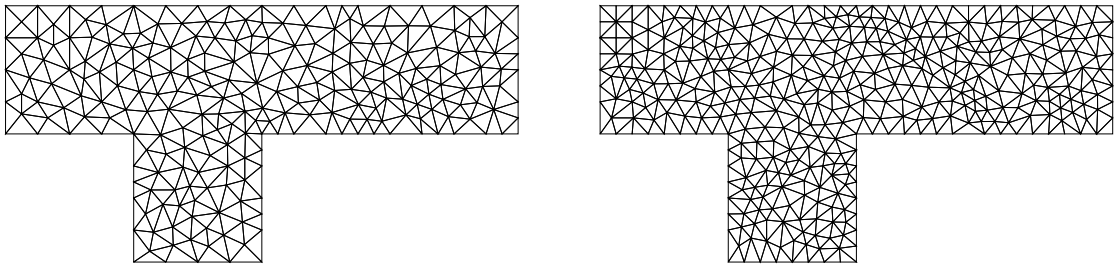
In[•]:=
```
{Max[Dmesh[[6]]["MeshElementMeasure"]], Min[Dmesh[[6]]["MeshElementMeasure"]]}
```

Out[•]=
```
{0.0119062, 0.0031391}
```

(i.e., our *MaxCellMeasure* → 0.012). Let's compare this to the default mesh:

In[•]:=
```
GraphicsGrid[{{Dmesh[[10]]["Wireframe"], Dmesh[[6]]["Wireframe"]}}]
```

Out[•]=



The average element size is approximated by the domain area and the number of mesh elements:

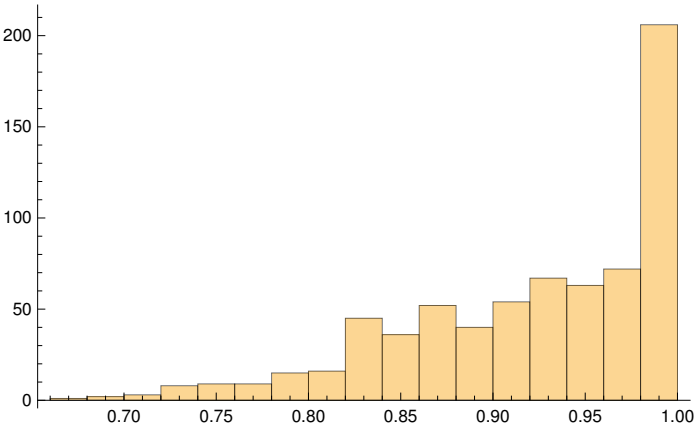
In[•]:=
```
N[5 / 698]
```

Out[•]=
```
0.00716332
```

The quality of the elements that make up this mesh:

In[•]:=
```
Histogram[Dmesh[[6]]["Quality"]]
```

Out[•]=



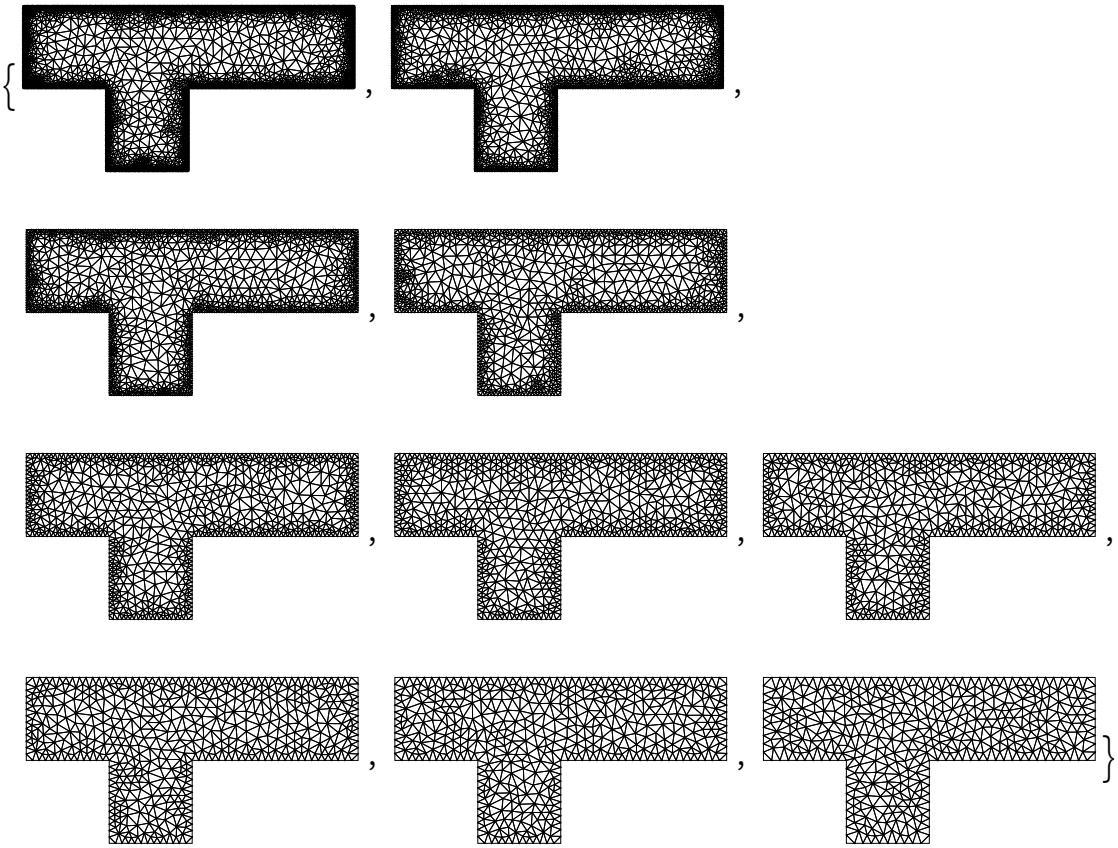
Boundary Mesh Refinement

Mesh Generation

To refine the boundary elements of our mesh, we will use the *MaxBoundaryCellMeasure* command.

In[ ]:=
```
DmeshB =
  Table[ToElementMesh[DEPnd, MaxCellMeasure → 0.012, "MaxBoundaryCellMeasure" → n,
    MeshQualityGoal → "Maximal"], {n, 0.01, 0.1, 0.01}];
Table[DmeshB[[n]]["Wireframe"], {n, 1, 10, 1}]
```

Out[ ]=



Solution Generation

In[ ]:=
```
Clear[u, v, p, x, y]

resultsB = Table[NDSolveValue[{EQs, BCs}, {u, v, p}, {x, y} ∈ DmeshB[[11 - n]],
    Method → {"FiniteElement", "InterpolationOrder" → {u → 2, v → 2, p → 1},
      "MeshOptions" → {AccuracyGoal → 16, PrecisionGoal → 16}}], {n, 1, 10, 1}];

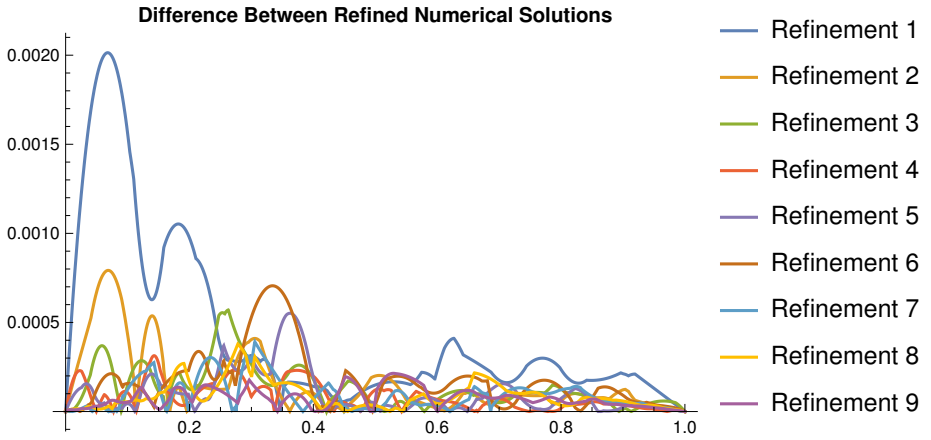
xVelB = resultsB[[All, 1]];
yVelB = resultsB[[All, 2]];
pressureB = resultsB[[All, 3]];
```

Convergence Analysis

Following, we repeat the same process we used for our interior mesh elements.

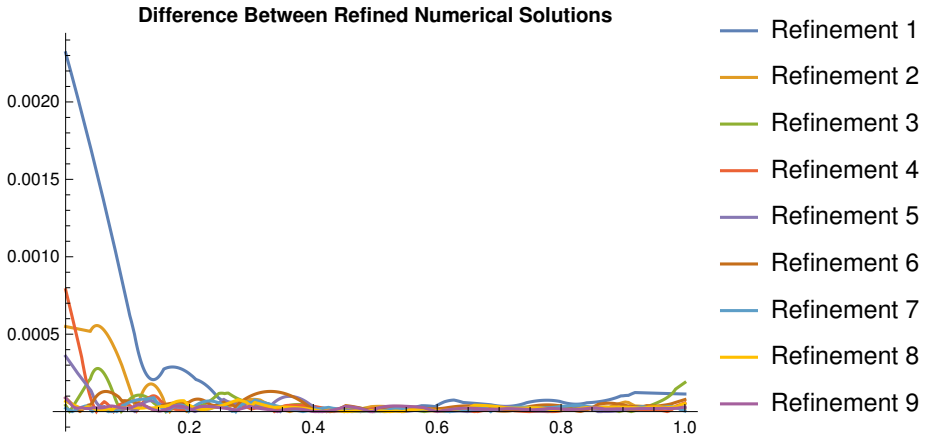
In[•]:=
```
Plot[Evaluate[Table[Abs[xVelB[[i]][0.75, y] - xVelB[[i + 1]][0.75, y]], {i, 1, 9, 1}]], {y, 0, 1},
 PlotRange → All, PlotLabel → "Difference Between Refined Numerical Solutions",
 PlotLegends → Table[StringForm["Refinement `1`", i], {i, 1, 9, 1}]]
```

Out[•]=



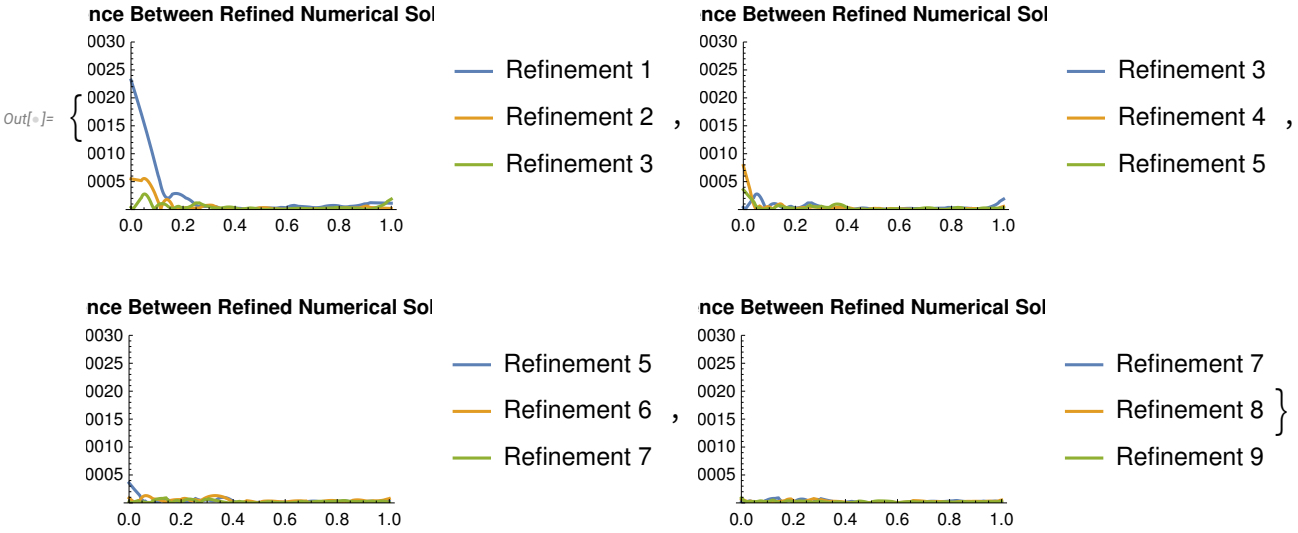
In[•]:=
```
Plot[Evaluate[Table[
   Abs[xVelB[[i]][0.75, y] - xVelB[[i + 1]][0.75, y]] / (xVelB[[i]][0.75, y] + xVelB[[i + 1]][0.75, y]) / 2,
   {i, 1, 9, 1}]], {y, 0, 1}, PlotRange → All,
 PlotLabel → "Difference Between Refined Numerical Solutions",
 PlotLegends → Table[StringForm["Refinement `1`", i], {i, 1, 9, 1}]]
```

Out[•]=



In[•]:=
```
Table[Plot[Evaluate[Table[
    Abs[xVelB[[i]][0.75, y] - xVelB[[i + 1]][0.75, y]] / (xVelB[[i]][0.75, y] + xVelB[[i + 1]][0.75, y]) /
     2, {i, n, n + 2, 1}]], {y, 0, 1}, PlotRange → {0, 0.003},
  PlotLabel → "Difference Between Refined Numerical Solutions",
  PlotLegends → Table[StringForm["Refinement `1`", i], {i, n, n + 2, 1}]], {n, 1, 7, 2}]
```

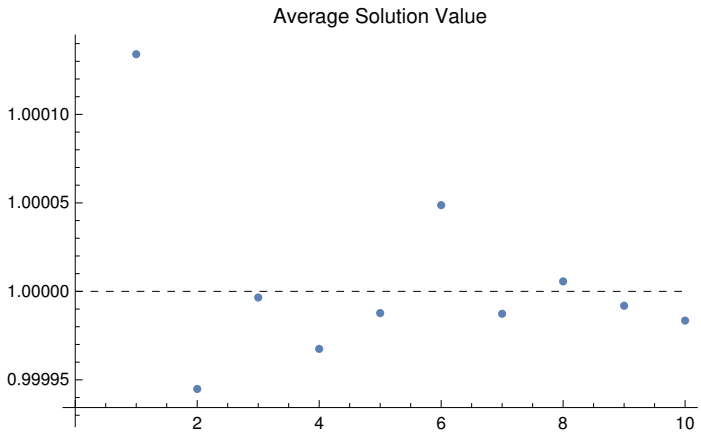
Out[•]=



It appears that two refinements of our boundary mesh elements is sufficient.

In[•]:=
```
ASVdataB = Table[NIntegrate[xVelB[[n]][0.75, y], {y, 0, 1}], {n, 1, 10, 1}];
ListPlot[ASVdataB, PlotLabel → "Average Solution Value",
 Epilog → {Dashed, Line[{{0, 1}, {10, 1}}]}]
```

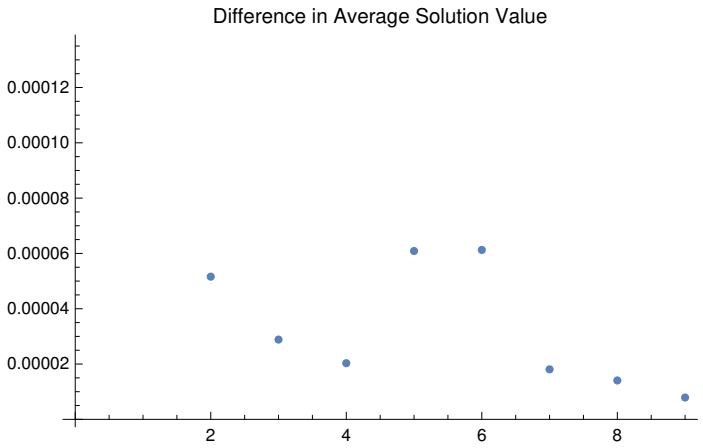
Out[•]=



Again, two refinements appears sufficient.

In[ ]:=
```
Off[NIntegrate::ncvb]
Off[NIntegrate::slwcon]
ASVdiffdataB =
  Table[Abs[NIntegrate[xVelB[[n]][0.75, y] - xVelB[[n + 1]][0.75, y], {y, 0, 1}]], {n, 1, 9, 1}];
ListPlot[ASVdiffdataB, PlotLabel → "Difference in Average Solution Value"]
```

Out[ ]=



Again, we're splitting hairs with this analysis. At most, the solution value changes by 0.01%.

In[ ]:=
```
Max[ASVdiffdataB]
```

Out[ ]=
```
0.000188042
```

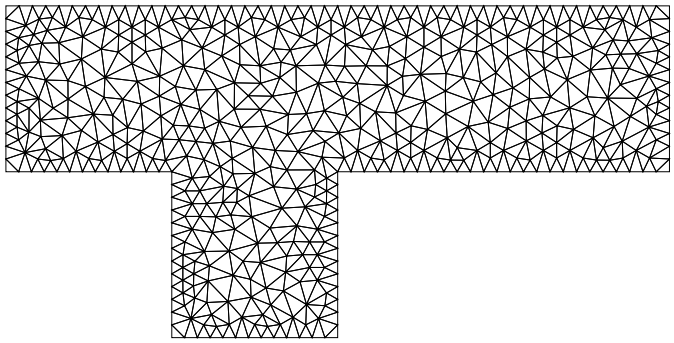
Analysis Conclusion

Moving forward, we will use a mesh with, at *most*, a *MaxBoundaryCellMeasure → 0.08.*

**We may choose to adjust this quantity down to ensure convergence of the numerical flow solver.*

In[ ]:=
```
DmeshB[[8]]["Wireframe"]
```

Out[ ]=



In[ ]:=
```
{Max[DmeshB[[8]]["MeshElementMeasure"]], Min[DmeshB[[8]]["MeshElementMeasure"]]}
```

Out[ ]=
```
{0.0118877, 0.00236317}
```

Summary

If we wanted too, we could also analyze the quality of our mesh.

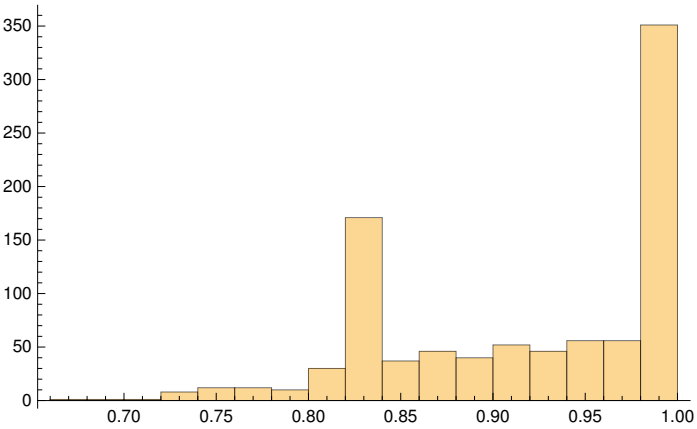
Mesh quality is described by Mathematica in the Element Mesh Generation Tutorial:
"The overall quality of a mesh can be expressed as a number between 0 and 1, with 1 being the best quality under a given quality estimate; 0 indicates the worst quality. A negative quality indicates that the incidents of an element are not in the proper order, that the incidents produce a self-intersecting element, or both."

In[•]:=
```
q = DmeshB[[8]]["Quality"];
{Min /@ q, Mean /@ q, StandardDeviation /@ q}
```

Out[•]=
```
{{0.663867}, {0.918508}, {0.0752608}}
```

In[•]:=
```
Histogram[q]
```

Out[•]=

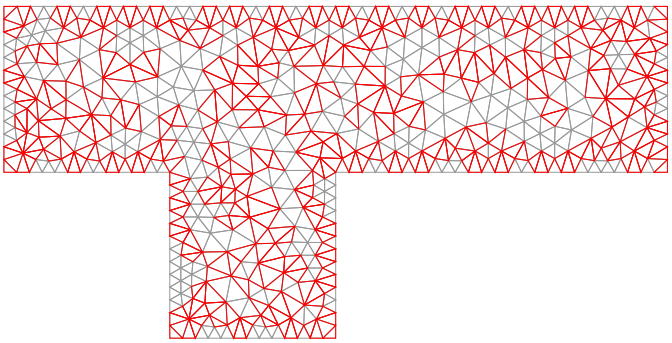


From these metrics, we can see that most of our mesh elements have a rather high quality (i.e., 0.92).

Mesh elements below the average are located randomly throughout our mesh, but mostly at the boundary.

In[•]:=
```
mean = # & @@ Mean /@ q;
pos = Position[q, _?(# ≤ mean &)];
Show[DmeshB[[8]][
  "Wireframe"["ElementMeshDirective" → Directive[EdgeForm[GrayLevel[.6]], FaceForm[]]],
 DmeshB[[8]]["Wireframe"[pos, "MeshElement" → "MeshElements",
   "ElementMeshDirective" → Directive[EdgeForm[Red], FaceForm[]]]]
```

Out[•]=



The boundary mesh must be refined such that the boundary layer is properly resolved and boundary conditions are enforced.

For laminar flow, the boundary layer thickness can be approximated by the following relationship:

$$\delta_{99}(x) = 5\,\frac{x}{\sqrt{Re_x}}$$

In non-dimensional space,

$$\delta^*_{99}(x^*) = 5\,\frac{x^*\,height}{\sqrt{Re_{x^*}}}$$

In[•]:=
```
delta99 = 5 1 * height / √RE
```

Out[•]=
```
0.00175445
```

Our mesh has minimum elements of *area* 0.002, which is on the same order of magnitude as the size of our boundary layer.

Addendum

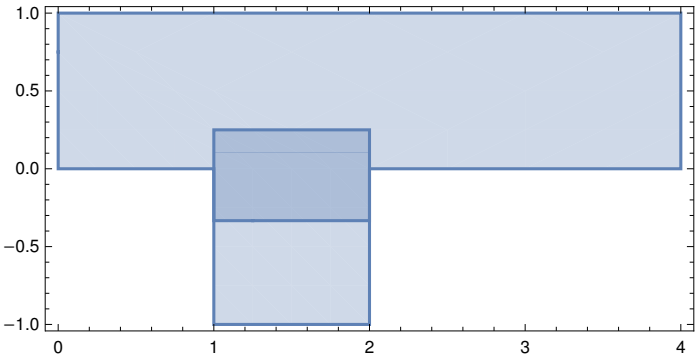
As part of our solver, we specify a refinement region to further reduce the mesh element area in the region of the separatrix.

To specify the refinement region, we use the follow commands:

```
refinementRegion =
  Rectangle[{height, (-1/3 * height)} / height, {2 * height, (1/4 * height)} / height];
Show[RegionPlot[Region[DEPnd]], RegionPlot[refinementRegion], AspectRatio → Automatic]
```

```
mrf = With[{rmf = RegionMember[refinementRegion]},
  Function[{vertices, area}, Block[{x, y}, {x, y} = Mean[vertices];
    If[rmf[{x, y}], area > 0.00025, area > 0.00025]]]]
```

Out[ ]=



Within the refinement region, our mesh elements have a minimum area of 0.00025.

We previously calculated that the average internal element size of our mesh was 0.00716332, making the elements within the refinement region, at most, 28x smaller.

```
0.0071633237822349575` / 0.00025
```

Out[ ]= 28.6533