

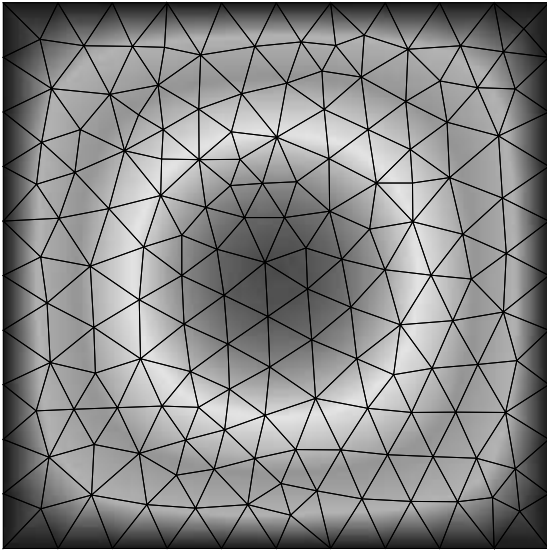
Here is some code to solve Poisson's Equation using the Finite Element Method (FEM) on arbitrary unstructured meshes:

```

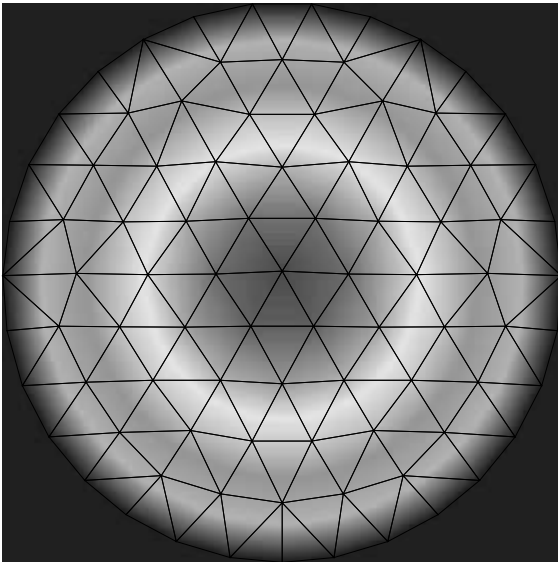
ImportMesh[folder_] := Module{}, nodes = Import[StringJoin[folder, "nodes.txt"], "CSV"];
  elements = Import[StringJoin[folder, "elements.txt"], "CSV"];
SolvePoissonFEM[] := Module{jmax = Length[elements]; xlist = Map#[[1] &, nodes]; ylist = Map#[[2] &, nodes];
  {x1, x2} = {Min[xlist], Max[xlist]}; {y1, y2} = {Min[ylist], Max[ylist]}; nxy = Length[nodes]; Q = Table[0, {nxy}];
  Klocal = Map[Module{}, Do{ $\Delta x_i$ ,  $\Delta y_i$ , void} = nodes[[#][i]] - nodes[[#][1]], {i, 2, 3};  $\lambda_2 = \Delta x_2^2 + \Delta y_2^2$ ;
   $\lambda_{23} = \Delta x_2 \Delta x_3 + \Delta y_2 \Delta y_3$ ;  $\lambda_3 = \Delta x_3^2 + \Delta y_3^2$ ; d =  $\Delta x_2 \Delta y_3 - \Delta x_3 \Delta y_2$ ; Do[Q[[#][n]] += d/6, {n, 1, 3}];
  Chop[ $\frac{1}{d} \begin{pmatrix} \lambda_2 + \lambda_3 - 2\lambda_{23} & \lambda_{23} - \lambda_3 & \lambda_{23} - \lambda_2 \\ \lambda_{23} - \lambda_3 & \lambda_3 & -\lambda_{23} \\ \lambda_{23} - \lambda_2 & -\lambda_{23} & \lambda_2 \end{pmatrix}$ ] &, elements]; Kglobal = Table[0, {nxy}, {nxy}];
  Do[element = elements[[j]]; Do[Kglobal[[element[[m]], element[[n]]] += Klocal[[j, m, n], {m, 1, 3}, {n, 1, 3}],
  {j, 1, jmax}]; Do{x, y, boundary} = nodes[[i]];
  If[boundary == 1, Kglobal[[i]] = Table[If[j == i, 1, 0], {j, 1, nxy}]; Q[[i]] = 0,
  {i, 1, nxy}]; q = Chop[LinearSolve[Kglobal, Q]]];
Interpolate[x_, y_] := Module{x1, y1}, {x1, y1} = plist[[1]]; Do{ $\Delta x_i$ ,  $\Delta y_i$ } = plist[[i]] - {x1, y1}, {i, 2, 3};
  { $\xi$ ,  $\eta$ } = ((x - x1) { $\Delta y_3$ , - $\Delta y_2$ } + (y - y1) {- $\Delta x_3$ ,  $\Delta x_2$ }) / ( $\Delta x_2 \Delta y_3 - \Delta x_3 \Delta y_2$ ); q[[element]].{1 -  $\xi$  -  $\eta$ ,  $\xi$ ,  $\eta$ };
PlotSolution[] := Module{}, r = (y2 - y1) / (x2 - x1); ny = 275; nx = Round[ny / r];
  image = Table[0, {ny}, {nx}]; Do[element = elements[[j]]; xlist = Map#[[1] &, nodes[element]];
  ylist = Map#[[2] &, nodes[element]]; plist = Transpose{xlist, ylist};
  xIntersect[{x1_, y1_}, {x2_, y2_}] := If[y1 == y2,  $\infty$ , x1 + (y - y1) (x2 - x1) / (y2 - y1)];
  Do[y = y1 + (y2 - y1) (i - 1) / (ny - 1); jlist = Round[(nx - 1)
  (Select[Map[xIntersect, Partition[plist, 2, 1, 1]], (Min[xlist] ≤ # ≤ Max[xlist]) &] - x1) / (x2 - x1) + 1];
  If[jlist != {}, Do[x = x1 + (x2 - x1) (jj - 1) / (nx - 1); image[[i, jj]] = Interpolate[x, y],
  {jj, Max[1, Min[jlist]], Min[nx, Max[jlist]]], {i, Max[1, Round[(ny - 1) (Min[ylist] - y1) / (y2 - y1) + 1]],
  Min[ny, Round[(ny - 1) (Max[ylist] - y1) / (y2 - y1) + 1]]], {j, 1, jmax}];
ListDensityPlot[image, AspectRatio → Automatic, Mesh → False, Frame → False,
  ColorFunction → (Hue[2 (1 - #) / 3] &), AspectRatio → r,
  ImageSize → {nx, ny}, Epilog → Table[element = elements[[j]];
  Line[Map[(x, y, boundary) = nodes[[#]]; {nx (x - x1) / (x2 - x1), ny (y - y1) / (y2 - y1)}] &,
  Append[element, element[[1]]], {j, 1, jmax}]]];
folder = "C:/School/Engineering/ME207 - Computer Modeling/Project/";
folder = "";

```

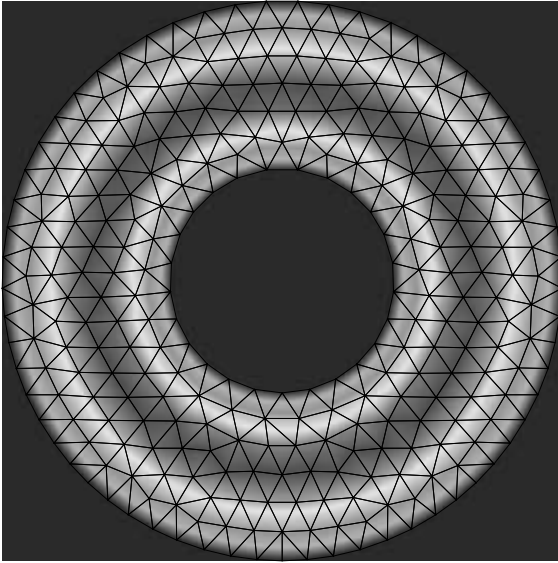
```
ImportMesh[StringJoin[folder, "Square/"]; SolvePoissonFEM[]; PlotSolution[];
```



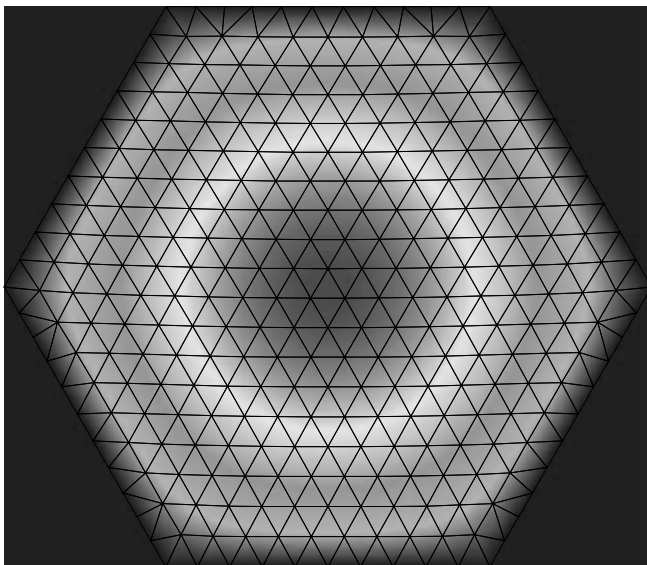
```
ImportMesh[StringJoin[folder, "Circle/"]; SolvePoissonFEM[]; PlotSolution[];
```



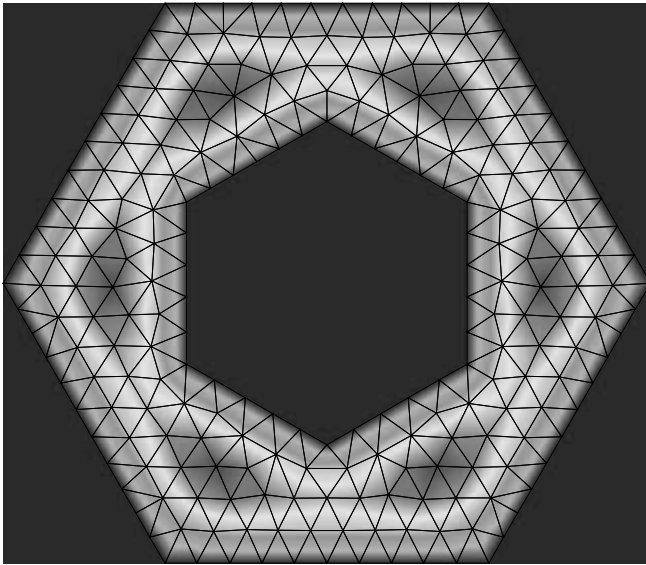
```
ImportMesh[StringJoin[folder, "Donut/"]; SolvePoissonFEM[]; PlotSolution[];
```



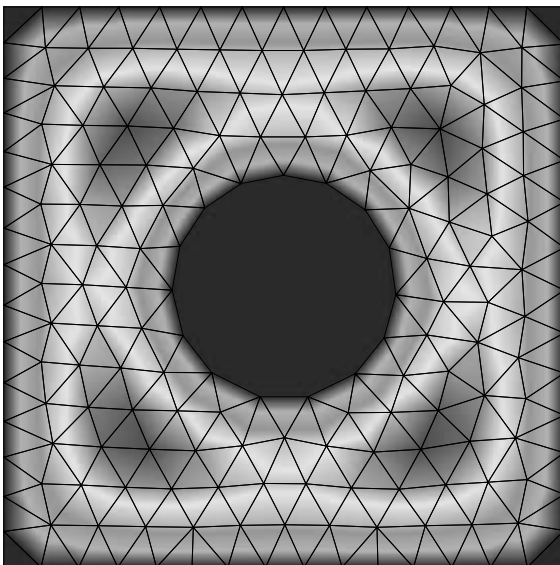
```
ImportMesh[StringJoin[folder, "Hex/"]; SolvePoissonFEM[]; PlotSolution[];
```



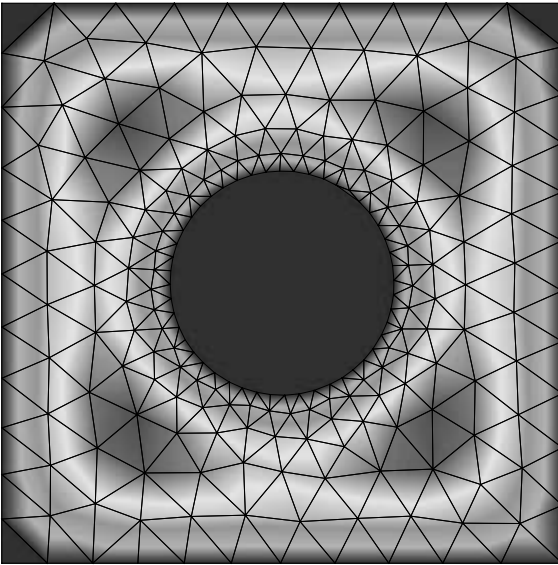
```
ImportMesh[StringJoin[folder, "HexDonut/"]; SolvePoissonFEM[]; PlotSolution[];
```



```
ImportMesh[StringJoin[folder, "SquareDonut/"]; SolvePoissonFEM[]; PlotSolution[];
```



```
ImportMesh[StringJoin[folder, "SquareDonut2/"]]; SolvePoissonFEM[]; PlotSolution[];
```



```
ImportMesh[StringJoin[folder, "Sunset/"]]; SolvePoissonFEM[]; PlotSolution[];
```

