FEM - Laplace

Gregory Javens 4/3/2015

Parameter setup.

```
Nodes = Import["Mathematica Projects\FEM\ChnlNodes.dat", "Table"];
Elems = Import["Mathematica Projects\FEM\ChnlElems.dat", "Table"];
N1[α_, β_, γ_, δ_, x_, y_] := ((γ - x) (δ - y)) / ((γ - α) (δ - β));
N2[α_, β_, γ_, δ_, x_, y_] := ((x - α) (δ - y)) / ((γ - α) (δ - β));
N3[α_, β_, γ_, δ_, x_, y_] := ((x - α) (y - β)) / ((γ - α) (δ - β));
N4[α_, β_, γ_, δ_, x_, y_] := ((γ - x) (y - β)) / ((γ - α) (δ - β));
```

Global K matrix and boundary condition on 61st node.

```
K = ConstantArray[0, {188, 188}];

For[k = 1, k < Length[Elems] + 1,
 (*Localizing the basis polynomials*)
 α = Nodes[[(Elems[[k]][[2]])]][[2]];
 β = Nodes[[(Elems[[k]][[2]])]][[3]];
 γ = Nodes[[(Elems[[k]][[4]])]][[2]];
 δ = Nodes[[(Elems[[k]][[4]])]][[3]];

 For[i = 1, i < 5,
   (*Taking gradients of localized basis polynomial*)
   Gi = Grad[Ni[α, β, γ, δ, x, y], {x, y}];

   For[j = 1, j < 5,
     (*Taking gradient of second localized basis polynomial and dotting*)
     Gj = Grad[Nj[α, β, γ, δ, x, y], {x, y}];
     dot = Gi.Gj;
     (*Retrieving the result of the integration
      and accumulating it in position (ni,nj) of the Global K *)
     nodei = Elems[[k]][[i + 1]];
     nodej = Elems[[k]][[j + 1]];
     K[[nodei]][[nodej]] =
      K[[nodei]][[nodej]] + Integrate[dot, {x, α, γ}, {y, β, δ}];

     j++;]
    i++;]
  k++;]

(*Setting 61st row of Global K as unit vector e61*)
unitVector = ConstantArray[0, 188];
unitVector[[61]] = 1;
K[[61]] = unitVector;
```

R Matrix

```
R = ConstantArray[0, 188];
(*Left-hand side boundary*)
For[h = 1, h < 10,

 α = Nodes[[Elems[[h]][[2]]]][[2]];
 β = Nodes[[Elems[[h]][[2]]]][[3]];
 γ = Nodes[[Elems[[h]][[4]]]][[2]];
 δ = Nodes[[Elems[[h]][[4]]]][[3]];


 position1 = Elems[[h]][[2]];
 R[[position1]] = R[[position1]] - (-Integrate[N1[α, β, γ, δ, α, y], {y, δ, β}]);


 position4 = Elems[[h]][[5]];
 R[[position4]] = R[[position4]] - (-Integrate[N4[α, β, γ, δ, α, y], {y, δ, β}]);

 h++;
]
(*Right-hand side boundary*)
For[h = 147, h < 156,
 α = Nodes[[Elems[[h]][[2]]]][[2]];
 β = Nodes[[Elems[[h]][[2]]]][[3]];
 γ = Nodes[[Elems[[h]][[4]]]][[2]];
 δ = Nodes[[Elems[[h]][[4]]]][[3]];

 position2 = Elems[[h]][[3]];
 R[[position2]] = R[[position2]] + (Integrate[N2[α, β, γ, δ, γ, y], {y, β, δ}]);


 position3 = Elems[[h]][[4]];
 R[[position3]] = R[[position3]] + (Integrate[N3[α, β, γ, δ, γ, y], {y, β, δ}]);


 h++;
]
```

Solution

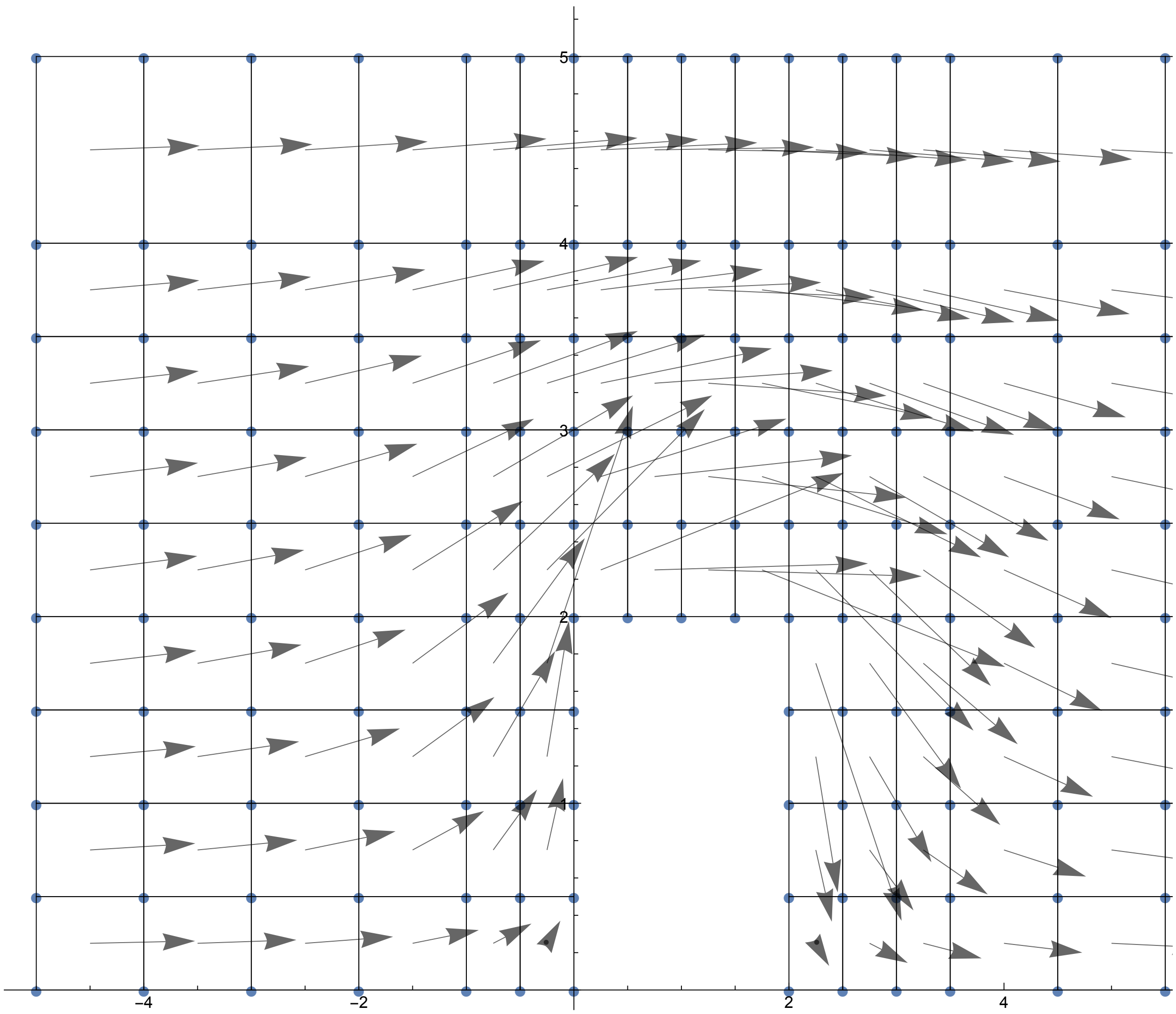
```
(*Solving global system for coefficients*)
coeffs = LinearSolve[K, R];

solution[a1_, a2_, a3_, a4_, x_, y_] := (a1 * N1[α, β, γ, δ, x, y]) +
    (a2 * N2[α, β, γ, δ, x, y]) + (a3 * N3[α, β, γ, δ, x, y]) + (a4 * N4[α, β, γ, δ, x, y]);

(*gathering the solution for each element in a list of solutions.*)
gatherSolutions = ConstantArray[0, 155];
For[i = 1, i < Length[Elems] + 1,
 (*localizing the element*)
 α = Nodes[[(Elems[[i]][[2]])]][[2]];
 β = Nodes[[(Elems[[i]][[2]])]][[3]];
 γ = Nodes[[(Elems[[i]][[4]])]][[2]];
 δ = Nodes[[(Elems[[i]][[4]])]][[3]];
 (*Accessing the appropriate coefficients*)
 a1 = coeffs[[Elems[[i]][[2]]]];
 a2 = coeffs[[Elems[[i]][[3]]]];
 a3 = coeffs[[Elems[[i]][[4]]]];
 a4 = coeffs[[Elems[[i]][[5]]]];

 gatherSolutions[[i]] = solution[a1, a2, a3, a4, x, y];

 i++;
]
```

Evaluating the gradient at the centroid of each element to yield the flow field.

```
(*Two lists to gather the gradients at
 each centroid and the vectors at each element*)
centroidGrad = ConstantArray[{0, 0}, 155];
arrows = ConstantArray[0, 155];

For[i = 1, i < Length[Elems] + 1,
 (*localizing the element*)
 α = Nodes[[(Elems[[i]][[2]])]][[2]];
 β = Nodes[[(Elems[[i]][[2]])]][[3]];
 γ = Nodes[[(Elems[[i]][[4]])]][[2]];
 δ = Nodes[[(Elems[[i]][[4]])]][[3]];

 (*Evaluating the gradient of the solution*)
 gradient = Grad[gatherSolutions[[i]], {x, y}];

 (*Finding x and y coordinates of the centroid of each element*)
 xCentroid = (α + γ) / 2;
yCentroid = (β + δ) / 2;

 (*Evaluating the gradient of the element at
  the centroid and sending it to the list of gradients*)
 centroidGrad[[i]] = gradient /. {x → xCentroid, y → yCentroid};

 (*Placing the vector of the element to a list of vectors*)
 arrows[[i]] = Arrow[{{xCentroid, yCentroid},
    {xCentroid, yCentroid} + {centroidGrad[[i]][[1]], centroidGrad[[i]][[2]]}}];

 i++
]
```

Post Processing - Visualization of the flow field.

```
(*Gathering the coordinate part of the Nodes list to listplot it.*)

NodeCoords = {{-5, 0}};
For[i = 2, i < 189,
 (*extracting the coordinate part of the Nodes list to use listplot*)
 AppendTo[NodeCoords, { Nodes[[i]][[2]], Nodes[[i]][[3]]}];
 i++
]
(*Drawing the obstruction*)
linesList = ConstantArray[{0, 0, 0, 0}, 155];
For[i = 1, i < 156,
 (*localizing the element*)
 α = Nodes[[(Elems[[i]][[2]])]][[2]];
 β = Nodes[[(Elems[[i]][[2]])]][[3]];
 γ = Nodes[[(Elems[[i]][[4]])]][[2]];
 δ = Nodes[[(Elems[[i]][[4]])]][[3]];
 (*For each element draw the line across each side*)
 linesList[[i, 1]] = Line[{{α, β}, {γ, β}}];
 linesList[[i, 2]] = Line[{{γ, β}, {γ, δ}}];
 linesList[[i, 3]] = Line[{{γ, δ}, {α, δ}}];
 linesList[[i, 4]] = Line[{{α, δ}, {α, β}}];
 i++
]
Show[ListPlot[NodeCoords, PlotMarkers → {Automatic, Small}],
 Graphics[linesList], Graphics[{Arrowheads[0.02], Opacity[0.6], arrows}]]
```


5
4
3
2
1
-4
-2
2
4