Manufacturing an Electric Field Using a Random Walk With Cooling

Ian Friedrichs, Stevens Institute of Technology

Introduction

Suppose that the user wants to manufacture an electric field of a specific "shape" within a subset of the square plane spanning {{-1,-1}, {,-1,1}, {1,1}, {1,-1}}.

The "shape" of the field must be supplied as a vector field (e.g. {5x+y, 2x}).

The rectangular area of interest must be supplied supplied as 4 floating-point numbers corresponding to the minimum and maximum x and y values of the area.

For simplicity, the field will be manufactured using 8 stationary point charges (E-field sources). What's optimized are the sign and magnitude of the charge at each point, in order to produce an approxima - tion of the desired field.

The accuracy metric (or loss function) used is the average proportional difference in the field at ran - domly selected points within the region of interest. The points are replaced every time the loss function is called to avoid overfitting. NOTE that this resampling of points means that the loss will begin to jump around when there are a small number of points.

A neighbor to the current best solution is calculated by varying the charge on a user-specified number of the 8 field sources. The aggressiveness of this varying decreases linearly over time (hence the term "cooling" in the title, borrowed from simulated annealing).

User Input

User-Supplied Field & Region Input

In[2088]:=
```
userField := {Cos[x], -0.3 Sin[y]};
xMinUser := 0.5;
xMaxUser := 0.75;
yMinUser := 0.5;
yMaxUser := 0.75;
```

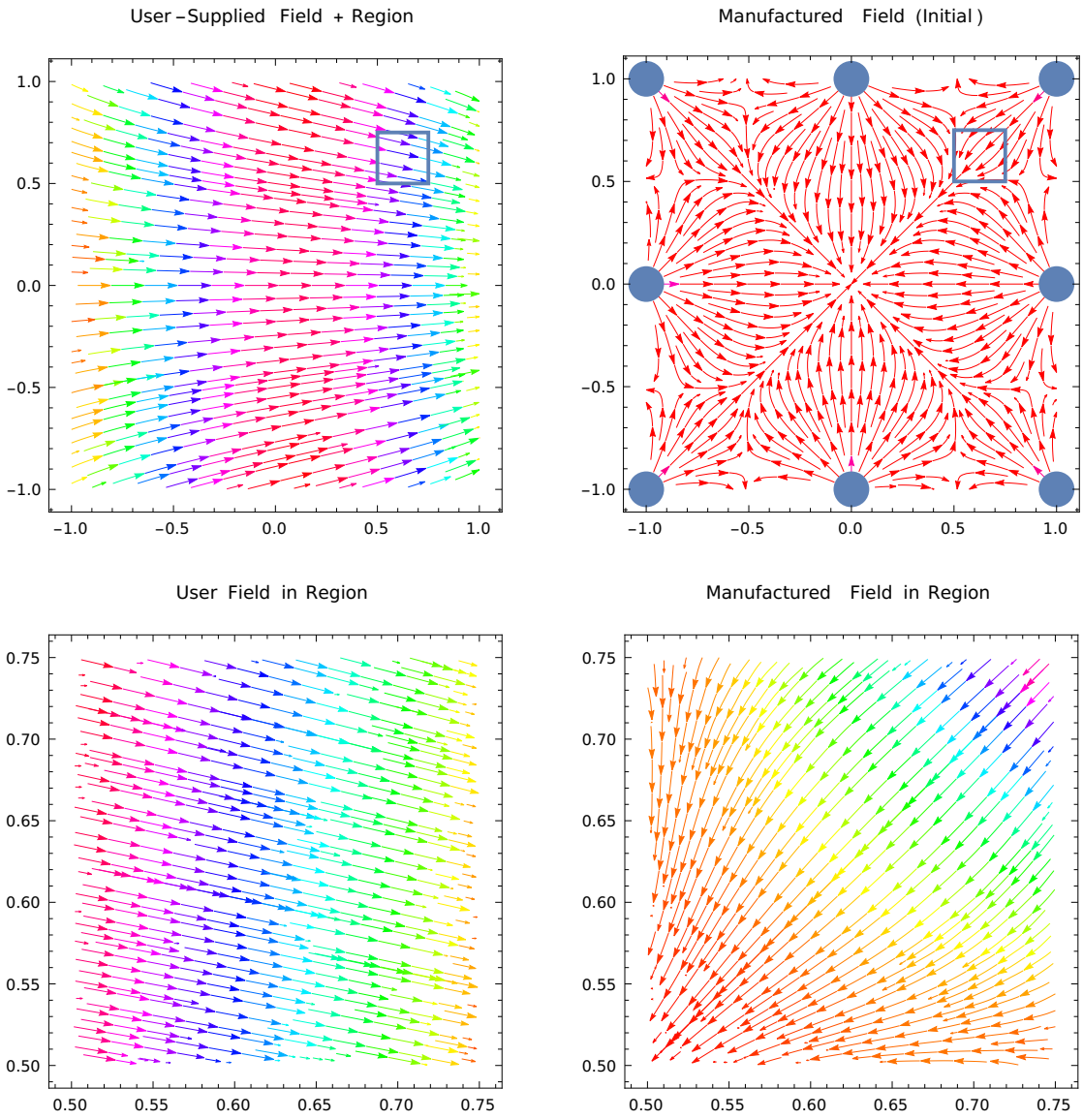
Other User-Supplied Inputs

In[2093]:=
```
(* the desired average accuracy out of 1
  (i.e. the average of the separate x and y accuracies) *)
userAcc = 0.95;
(* the number of points randomly sampled within the region of interest
 every time that the loss function is calculated *)
numPtsSA = 200;
(* the minimum and maximum charge that any single field source can hold *)
userMinChg = -10;
userMaxChg = 10;
(* the maximum number of simulated annealing steps to take *)
stepsSA = 10 000;
(* the number of field sources that will have their charges varied
 to calulate the next neighbor *)
numVarySA = 3;
(* the maximum (and starting) percentage to vary the field sources by (out of 1) *)
startPercVarySA = 0.7;
(* print out the loss every (numSteps / factor) steps *)
userPrintModFactor = 1000;
```

Display the Results of User-Supplied Field Input (Alongside the Initial Manufactured State)

In[2101]:=
```
(* for the user-supplied information *)
userFieldPlt = StreamPlot[userField, {x, -1, 1}, {y, -1, 1}, StreamPoints → Fine,
    StreamColorFunction → Hue, PlotLabel → "User-Supplied Field + Region"];
interestPointsShow := {
    {xMinUser, yMaxUser}, {xMaxUser, yMaxUser},
    {xMaxUser, yMinUser}, {xMinUser, yMinUser},
    {xMinUser, yMaxUser}};
interestPlt = ListLinePlot[interestPointsShow, PlotStyle → Directive[Thick]];
(* for the manufactured field *)
charges = {
    {-1, 1, 1}, {0, 1, 1}, {1, 1, 1},
    {-1, 0, 1}, {1, 0, 1},
    {-1, -1, 1}, {0, -1, 1}, {1, -1, 1}};
manPot = Total[ePot[{x, y}, #] & /@ charges];
manField = -Grad[manPot, {x, y}];
manFieldPlot = StreamPlot[manField, {x, -1, 1}, {y, -1, 1}, StreamPoints → Fine,
    StreamColorFunction → Hue, PlotLabel → "Manufactured Field (Initial)"];
manPoints = {{-1, 1}, {0, 1}, {1, 1},
    {-1, 0}, {1, 0},
    {-1, -1}, {0, -1}, {1, -1}};
manPointsPlt = ListPlot[manPoints, PlotMarkers → {Automatic, Large}];
(* Now show close-ups in the region of interest *)
manFieldPlotClose = StreamPlot[manField,
    {x, xMinUser, xMaxUser}, {y, yMinUser, yMaxUser}, StreamPoints → Fine,
    StreamColorFunction → Hue, PlotLabel → "Manufactured Field in Region"];
userFieldPltClose = StreamPlot[userField, {x, xMinUser, xMaxUser},
    {y, yMinUser, yMaxUser}, StreamPoints → Fine,
    StreamColorFunction → Hue, PlotLabel → "User Field in Region"];
GraphicsGrid[{{
    Show[userFieldPlt, interestPlt],
    Show[manFieldPlot, interestPlt, manPointsPlt]},
  {userFieldPltClose, manFieldPlotClose}}]
```

Out[2112]=



Optimization

Perform the Random Walk

In[2113]:=
```
(* evaluate the user's vector field at a certain point
    point → vector
    (x,y) → {x,y} *)
userFieldFunc[xIn_, yIn_] := userField /. x → xIn /. y → yIn;
```

In[2114]:=
```
(* calculate the error between the user-supplied and desired field *)
theErr[thePoint_, mff_] := Module[{tmpX, tmpY},
  tmpX = thePoint[[1]];
  tmpY = thePoint[[2]];
  Abs[(userFieldFunc[tmpX, tmpY] - mff[tmpX, tmpY]) / userFieldFunc[tmpX, tmpY]]
 ]
(* just takes the results of one call to theErr and averages it *)
avgErr[thePoint_] := Module[{},
  (thePoint[[1]] + thePoint[[2]]) / 2
 ]
(* practice for testing the loss function *)
pointsX = RandomReal[{xMinUser, xMaxUser}, numPtsSA];
pointsY = RandomReal[{yMinUser, yMaxUser}, numPtsSA];
pointsXY = Transpose @{pointsX, pointsY};
(* the loss function *)
lossFunc[mff_] := Total[avgErr /@ (theErr[#, mff] & /@ pointsXY)] / Length[pointsXY];
```

In[2120]:=
```
(* find a neighbor for a specific set of charges at a certain step *)
findNeighbor[theCharges_, theStep_] :=
 Module[{indices, maxVary, variations, chargesCpy},
  chargesCpy = theCharges;
  indices = RandomInteger[{1, Length[charges]}, numVarySA];
  maxVary = (1.0 * (stepsSA - theStep) / stepsSA) * startPercVarySA;
  variations = RandomReal[{-maxVary, maxVary}, numVarySA] + 1.0;
  For[i = 1, i ≤ numVarySA, i++,
   chargesCpy[[indices[[i]]]][[3]] = variations[[i]] * chargesCpy[[indices[[i]]]][[3]]
  ];
  chargesCpy
 ]
```

In[2121]:=
```
(* optimization loop *)
currentLoss = 1 * 10 ^ 10;
currentStep = 1;
(* the charges are what's bring optimized *)
Timing @ While[currentLoss > Abs[1 - userAcc] && currentStep < stepsSA,
    manPot = Total[ePot[{x, y}, #] & /@ charges];
    manField = -Grad[manPot, {x, y}];
    manFieldFunc[xIn_, yIn_] := manField /. x → xIn /. y → yIn;
    chargesTmp = findNeighbor[charges, currentStep];
    tmpPot = Total[ePot[{x, y}, #] & /@ chargesTmp];
    tmpField = -Grad[tmpPot, {x, y}];
    tmpFieldFunc[xIn_, yIn_] := tmpField /. x → xIn /. y → yIn;
    pointsX = RandomReal[{xMinUser, xMaxUser}, numPtsSA];
    pointsY = RandomReal[{yMinUser, yMaxUser}, numPtsSA];
    pointsXY = Transpose @ {pointsX, pointsY};
    manLoss = lossFunc[manFieldFunc];
    tmpLoss = lossFunc[tmpFieldFunc];

    If[tmpLoss < manLoss,
     (
      currentLoss = tmpLoss;
      charges = chargesTmp;
     ),
     (
      currentLoss = manLoss
     )
    ];
    If[Mod[currentStep, userPrintModFactor] == 0,
     Print["Current Loss ", currentLoss]];
    currentStep = currentStep + 1;
   ];
```

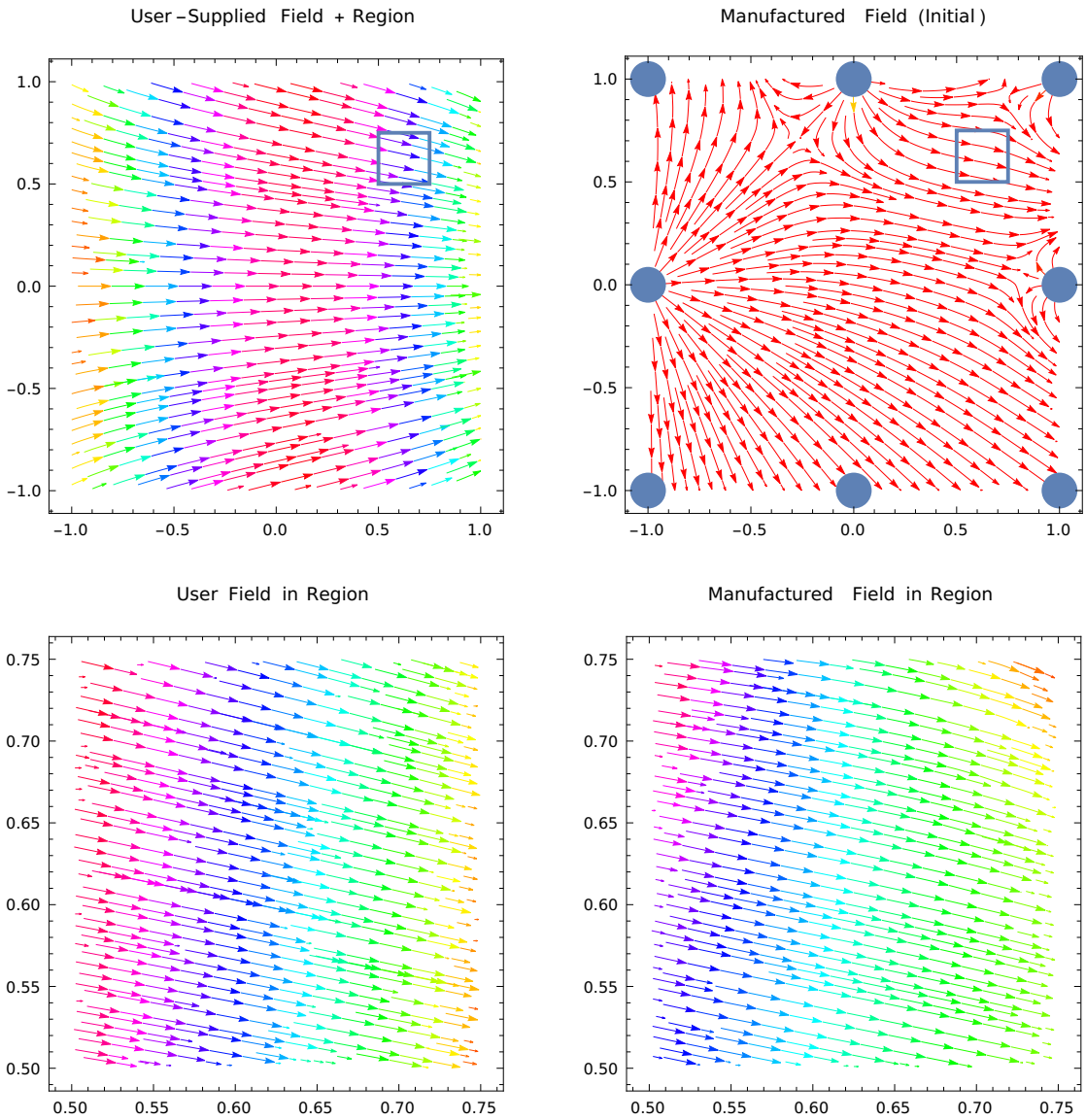
```
Current Loss 0.854269
Current Loss 0.262332
Current Loss 0.125581
Current Loss 0.102787
Current Loss 0.102571
Current Loss 0.104502
Current Loss 0.0984765
Current Loss 0.0977639
Current Loss 0.104099
```

Display the Results of the Random Walk

In[2124]:=
```
(* for the user-supplied information *)
userFieldPlt = StreamPlot[userField, {x, -1, 1}, {y, -1, 1}, StreamPoints → Fine,
    StreamColorFunction → Hue, PlotLabel → "User-Supplied Field + Region"];
interestPointsShow := {
    {xMinUser, yMaxUser}, {xMaxUser, yMaxUser},
    {xMaxUser, yMinUser}, {xMinUser, yMinUser},
    {xMinUser, yMaxUser}};
interestPlt = ListLinePlot[interestPointsShow, PlotStyle → Directive[Thick]];
(* for the manufactured field *)
manPot = Total[ePot[{x, y}, #] & /@ charges];
manField = -Grad[manPot, {x, y}];
manFieldPlot = StreamPlot[manField, {x, -1, 1}, {y, -1, 1}, StreamPoints → Fine,
    StreamColorFunction → Hue, PlotLabel → "Manufactured Field (Initial)"];
manPoints = {{-1, 1}, {0, 1}, {1, 1},
    {-1, 0}, {1, 0},
    {-1, -1}, {0, -1}, {1, -1}};
manPointsPlt = ListPlot[manPoints, PlotMarkers → {Automatic, Large}];
(* Now show close-ups in the region of interest *)
manFieldPlotClose = StreamPlot[manField,
    {x, xMinUser, xMaxUser}, {y, yMinUser, yMaxUser}, StreamPoints → Fine,
    StreamColorFunction → Hue, PlotLabel → "Manufactured Field in Region"];
userFieldPltClose = StreamPlot[userField, {x, xMinUser, xMaxUser},
    {y, yMinUser, yMaxUser}, StreamPoints → Fine,
    StreamColorFunction → Hue, PlotLabel → "User Field in Region"];
Print[charges];
GraphicsGrid[{{
    Show[userFieldPlt, interestPlt],
    Show[manFieldPlot, interestPlt, manPointsPlt]},
  {userFieldPltClose, manFieldPlotClose}}]
```

```
{{-1, 1, 0.0000402666}, {0, 1, 0.264996}, {1, 1, 0.0618607}, {-1, 0, 1.79291},
 {1, 0, 0.0236272}, {-1, -1, 1.48386 × 10^-11}, {0, -1, 3.11872 × 10^-14}, {1, -1, 5.06729 × 10^-10}}
```

Out[2135]=


User – Supplied Field + Region
Manufactured Field (Initial)
User Field in Region
Manufactured Field in Region