Lecture #16b

Numerical Implementation of Transformation Optics

Lecture Outline

• Numerical solution of Laplace’s equation
• Calculating spatial transforms using Laplace’s equation
• Calculating permeability and permittivity from spatial transforms
Numerical Solution of Laplace’s Equation

(Not Yet TO)

Meaning of Laplace’s Equation

Laplace’s equation is

\[ \nabla^2 u = 0 \]

\( \nabla^2 \) is a 3D second-order derivative. A second-order derivative quantifies curvature. But, we set the second-order derivative to zero.

Functions satisfying Laplace’s equation vary linearly.
Laplace’s Equation as a “Number Filler Inner”

Given known values at certain points (e.g. physical boundary conditions), Laplace’s equation calculates the numbers everywhere else so they vary linearly.

Numerical Solution of Laplace’s Equation (1 of 3)

Step 1 – Use the finite-difference method to express Laplace’s equation in matrix form.

\[ \nabla^2 u(x, y) = 0 \rightarrow \frac{\partial^2 u(x, y)}{\partial x^2} + \frac{\partial^2 u(x, y)}{\partial y^2} = 0 \rightarrow D_x^2 u + D_y^2 u = 0 \rightarrow L u = 0 \]

\[ L = D_x^2 + D_y^2 \]

Step 2 – Build a function \( b(x, y) \) containing the boundary values.

\[ b(x, y) \]

This function contains the values at the locations that we wish to force \( u(x, y) \) to. The other points in \( b(x, y) \) can be set to anything, but zero is convenient.
Numerical Solution of Laplace’s Equation (2 of 3)

Step 3 – Build a diagonal force matrix.

\[ F = \text{diag}[f(x, y)] \]

This function contains 1’s at the positions where we wish to force values in \( u(x,y) \). It contains 0’s everywhere else. We set it to 1’s where we have our boundary values.

Step 4 – Incorporate the boundary values into Laplace’s equation.

\[ L' = F + (I - F)L \]

\[ b' = Fb \]

Note: both \( F \) and \( I \) should be stored as sparse matrices!

Numerical Solution of Laplace’s Equation (3 of 3)

Step 5 – Solve Laplace’s equation

\[ u = (L')^{-1} b' \]

The function \( u(x,y) \) has all of the forced values, but also contains all of the numbers in between.
Enclosed Problems

Sometimes we only wish to obtain a solution that is perfectly enclosed by the boundary values.

Reducing Enclosed Problems (1 of 3)

Step 1 – Make a map \( M(x,y) \) of where we wish to solve Laplace’s equation.
Reducing Enclosed Problems
(2 of 3)

Step 2 – Eliminate the rows and columns that correspond to points where we do not wish to solve.

\[ L'u = b' \]

\[ \begin{bmatrix} \vdots \end{bmatrix} = \begin{bmatrix} \vdots \end{bmatrix} \]

\[ L''u'' = b'' \]

\[ \begin{bmatrix} \vdots \end{bmatrix} = \begin{bmatrix} \vdots \end{bmatrix} \]

% REDUCE LAPLACE’S EQUATION
ind = find(M(:));
L = L(ind,ind);
b = b(ind);

Reducing Enclosed Problems
(3 of 3)

Step 3 – Solve Laplace’s equation.

\[ u'' = (L'')^{-1} b'' \]

% SOLVE LAPLACE’S EQUATION
u = L\b;

Step 4 – Insert solution back into grid.

% INSERT SOLUTION BACK INTO GRID
U = zeros(Nx,Ny);
U(ind) = u;
Calculating Spatial Transforms Using Laplace’s Equation

Step 1: Construct Object and Cloak

Cloak Shape

Object to Cloak

CLK

OBJ

0’s

1’s
Step 2: Identify Edges of Cloak and Object (1 of 2)

Edge of Cloak (ECLK)

Edge of Object (EOBJ)

0’s

1’s

Step 2: Identify Edges of Cloak and Object (2 of 2)

All edges are detected at positions outside of the cloak region.

This places the edge of the object to be inside of the object.
Step 3: Mask Meshgrid to Set Initial Boundary Values

Step 4: Force Coordinates on Object to Zero
Step 5: Fill In Missing Coordinates (1 of 4)

Solve \( \nabla^2 x' = 0 \)

Solve \( \nabla^2 y' = 0 \)

Step 5: Fill In Missing Coordinates (2 of 4)

The matrix equation is constructed in three steps:

Step 1/3 – Construct ordinary Laplacian matrix equation.
\[
L = D_x^2 + D_y^2
\]
\[\mathbf{b} \equiv \text{column vector of forced values } b(x, y)\]

Step 2/3 – Enforce physical boundary conditions.
\[
L' = F + (I - F)L
\]
\[\mathbf{b'} = F\mathbf{b}\]

Step 3/3 – Eliminate don’t care points.
\[
L'' \leftarrow L'
\]
\[\mathbf{b}'' \leftarrow \mathbf{b'}\]
Step 5: Fill In Missing Coordinates (3 of 4)

The matrix equation is solved in two steps:

Step 1/2 – Solve for $u''$.

$$u'' = (L'')^{-1} b''$$

Step 2/2 – Insert points in $u''$ back into full grid.

$$u \leftarrow u''$$

Alternate view of coordinate transform:

Grid Before Transform

Grid After Transform
Calculating Permittivity and Permeability from the Spatial Transform

Step 6: Initialize Background Permittivity and Permeability

\[
[\mu_r] = [\varepsilon_r] =
\]

Lecture 16b – Numerical TD
Step 7: Calculate Derivatives of Transformed Grid

\[ \frac{\partial x'}{\partial x} \rightarrow D_{x'} \]
\[ \frac{\partial x'}{\partial y} \rightarrow D_{y'} \]
\[ \frac{\partial y'}{\partial x} \rightarrow D_{y'} \]
\[ \frac{\partial y'}{\partial y} \rightarrow D_{y'} \]

Step 8: Build UR and ER (1 of 4)

This step loops through each point on the grid.

For each point in the cloak region...

a. Build \([\mu_r]\) and \([\epsilon_r]\) background tensors.
Step 8: Build UR and ER (2 of 4)

This step loops through each point on the grid. For each point...

b. Build Jacobian.

\[
\frac{\partial \varepsilon'}{\partial x} \rightarrow D_{x} x'
\]

\[
\frac{\partial \varepsilon'}{\partial y} \rightarrow D_{y} y'
\]

\[
\begin{bmatrix}
\frac{\partial x'}{\partial x} & \frac{\partial x'}{\partial y} \\
\frac{\partial y'}{\partial x} & \frac{\partial y'}{\partial y}
\end{bmatrix} = \begin{bmatrix} 1 & 0 \\
0 & 1 \end{bmatrix}
\]

\[
\begin{bmatrix}
0 & 0 & 0 & 1
\end{bmatrix}
\]

Step 8: Build UR and ER (3 of 4)

This step loops through each point on the grid. For each point...

c. Transform \([\mu_r]\) and \([\varepsilon_r]\) using inverse of Jacobian.

\[
G = J^{-1}
\]

\[
\begin{bmatrix}
\mu' \end{bmatrix} = \frac{G[\mu_r]G^T}{\det G}
\]

\[
\begin{bmatrix}
\varepsilon' \end{bmatrix} = \frac{G[\varepsilon_r]G^T}{\det G}
\]

% Transform UR and ER

\[
J = \text{inv}(J);
\]

\[
UR = J*UR*J.'/\text{det}(J);
\]

\[
ER = J*ER*J.'/\text{det}(J);
\]
Step 8: Build UR and ER (4 of 4)

This step loops through each point on the grid. For each point...

d. Populate grid with transformed values of $[\mu_r]$ and $[\varepsilon_r]$.

\[
\begin{pmatrix}
\mu_{rx} \\
\mu_{ry} \\
\mu_{rz}
\end{pmatrix}, \begin{pmatrix}
\varepsilon_{rx} \\
\varepsilon_{ry} \\
\varepsilon_{rz}
\end{pmatrix}
\]

Step 9: Done!

Numerical TO is done! The material tensors can be imported into a CEM code for simulation.

\[
\begin{pmatrix}
\mu_{rx} & \mu_{ry} & \mu_{rz} \\
UR_{rx} & UR_{ry} & UR_{rz}
\end{pmatrix}, \begin{pmatrix}
\varepsilon_{rx} & \varepsilon_{ry} & \varepsilon_{rz} \\
ER_{rx} & ER_{ry} & ER_{rz}
\end{pmatrix}
\]