Formulation notes

# Introduction

I’m hoping to use this as just a quick overview and reference. The details of the formulation can be found in Chapter 10 in Computational Methods for Electromagnetics by Peterson, Raj, Mittra. This is more to a serve as a companion to my code, so that I don’t forget what decisions I made.

## The Plate

This is taken directly from Chapter 10 in Computational Methods for Electromagnetics by Peterson, Raj, Mittra. It is a method of moments approach for scattering from a flat perfectly conducting plate defined on rectangular cells. This will broken into two sections, getting the plate impedance (Z) and then feeding the system (E). Z and V are the matrix array we get when applying method of moments. Ie) E = Z J

The section will hopefully explain the thought process behind thePlate.m.

### Geometry

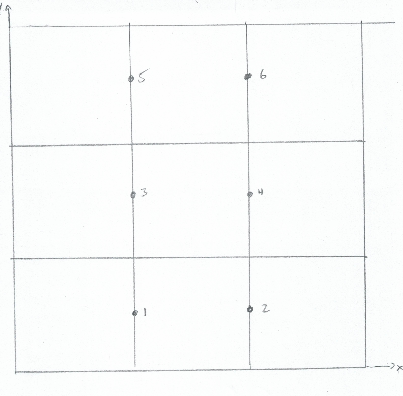
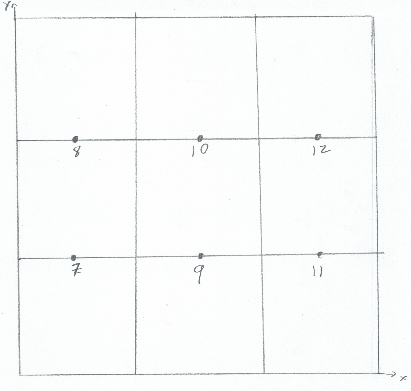
The plate is divided into rectangular cells. This allows for to break our model into its and components. The Electric Field Equation can then be written as

Where is the magnetic potential described by

The rooftop basis function will be used to describe the current. This basis function spans over two cells and is defined by at the center of the shared edge. In the direction of the current polarization, a triangle function used to force the current to die on the outer edges. A pulse function is used in the other direction to allow it to blow.

The important information out of here is that this mean and will be defined on different points. exists the shared edges of the two cells in x, but the center of the cell in y. x-position points are stored as Bxn\_xx and the corresponding y-position points stored as Bxn\_yy. Vice versa for which are stored as Byn\_xx and Byn\_yy.

I define first edge as the lower left hand. The idea was thinking in a cartesian plane where the lower left hand would be closets to (0,0). is defined first increases as we move right and upward. Then to define , is upward and to the left. This all is a bit hard to describe, so an example is provided for 3x3 cell division.

1a) center points defined 1b) center points defined

From here out, it’s pretty much normal method of moments process. The current is expanded as a set of basis function on the defined points. To create N-unknowns for the N-equations, the function is tested using a “razor blade” function (pulse in current direction and delta function in the other).

Going through all of this generates Z. The code stores this value as ZZ. The inverse is also stored as ZZinv. This is speed up “taking the inverse” when solving for the current. Only matrix multiplication will now be needed

### Feeding the plate

The code then feeds the system with theta polarized wave and phi polarized waves (TM and TE). The points are separated in and though. This means take the appropriate dot product ex) and then integrate against the testing function. shares ’s points and shares .

The wave and wave are stored separately as EE\_theta and EE\_phi. The current can now be calculated and are stored as JJ\_theta and JJ\_phi.

### Whats up with kk?

If while reading the code you notice kk=2\*pi scattered around, its just the wavenumber. Recall . The entire plate geometry is defined in lengths of for my sanity. This means that whenever we multiply k by distance we’ll get something proportional to .

### Initializing

The length is in wavelengths, not a ‘real’ length ex) 3. The incident angles are taken in as phiInc and thetaInc. NumCells describes how many cells across one row or column. For example, in the above figure 1, the would NumCells = 3. The division between cells should be at least .

## chaffElt

The plate is now defined. Holes need to be added to the plate to try and optimize the return.

### Initializing

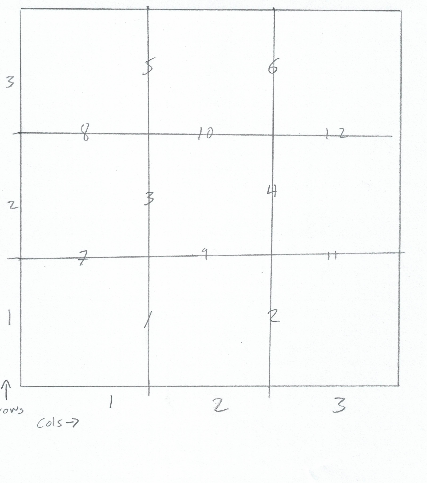
The code takes in the operating frequencies, the length of the plate, angles to be optimized (theta and phi), Number of cells across, and the number of pixels across. Cells refer to actual division of the plate while pixels are composed of cells. This necessary to fully represent the current when the code starts removing sections for optimization. Another way to think of it is that a pixel is composed of cells and should be thus smaller and be able to divide the number of cells. Ie. 20 cells across and 4 pixels would make each pixel 5 cells large.

These values are stored as freq, plateLength, thetaVals, phiVals, NumCells, pixelSize. Frequency, thetaVals, and phiVals can be an array of numbers. Frequency is in Hertz. Angles are in radians.

A plate for each frequency is generated with the code forcing at least for cell length. The plate is stored as plateFull and plateNull. Eventually, holes will be placed in the plate which involves removing elements from the impedance matrix. If a new design with different holes needs to be generated (and it will for optimization), a plate with no holes would be needed and different elements would be removed. Therefore two versions of the plate are stored. plateFull is complete metal while plateNull is the current design being examined.

### nullNew

nullNew takes in a matrix, nullPos, that defines rows and columns of cells to be removed. The first column of nullPos is the rows and second column that need to be removed. Here we’re talking about removing cells, but the plate is described in edges. This just means remove the surrounding edges. I think it’s easier to describe in example, so let return to 3x3 cell division.



To remove the middle cell (2,2), edges 3,4,9, and 10. To remove the middle bottom cell (1,2), edges 1,2, and 9 are removed.

For an edge to be removed, the corresponding column and row in the z matrix has to be removed. That edge is no longer interacting with any other edges. To continue the example, removing edge 3 means removing the third column and third row of the impedance matrix.

### NullPixels

This function takes in a matrix of row and column positions of pixels to be removed. A pixel is made up of cells. The main purpose of this function is to convert pixel positions into the cell position that will then be fed into nullNew.

We found that under this formulation, current could not flow across corners. If two pixels lye diagonal, other computational software showed that there should be current flowing across, because they are touching at the edges. This formulation does not allow for that. This function creates a fix by adding a single cell at the corner.

### SymmetricNullPix

In the end, we know the optimized chaff should be symmetric. The purpose of this function is just to force that. It takes in a square matrix of 1s and 0s, but only the lower triangle positions are used. The 1’s signify metal while 0 signifies a hole. The lower triangle is taken and reflected such that symmetric chaff design occurs.

From here, the idea is the same of NullPixels. The 0 positions are found and converted into row/column form needed for nullNew. The corner fix is applied and the position matrix is fed into nullNew.

### Optimization

Built-in matlab functions are used to maximize RCS. A series of 1 and 0’s are used to represent pixels being “on” (metal) or “off” (whole). Matlab then attempts to optimize null2minRCSAvgSym. The array is reshaped into a square matrix where only the lower triangle values matter (see nullPixels).

The function walks through each frequency, phi azimuthal angle, and theta elevation angle and calculates the RCS. Both polarizations are found averaged. At the end, all calculated RCS values are averaged. This is what the function attempts to maximize.