Rough Edge Diffraction over a Perfect Electric Conducting Half Plane

Zak Costello

December 2009

I. **Introduction**

There are several well know solutions to the flat edge perfect electric conducting half plane diffraction problem. The Sommerfeld solution is the most accurate and uses Maxwell's equations to form its boundary conditions. In numerical simulations, it can be very computationally costly. Using the Huygens-Fresnel principle to approximate diffraction phenomenon can produce accurate results in the far field (more than 8 wavelengths) , and tolerable results in the near field. Additionally, it has a lower computational cost and can be evaluated using parallel computation methods by exploiting its natural linearity.

The goal of this paper is to show the diffracted field solutions using the Huygens-Fresnel approximation when a perfect electric conducting half plane has a rough edge. Then I will compare the rough edge solutions of different parameters to the normally predicted flat edge solution.

II.Methodology

All numerical simulation was done using the Matlab 7.8.0 software package. A code was written to simulate rough edges with Gaussian characteristics.

Rough Edge Generation

In order to generate rough edges with the correct Gaussian random statistics a vector containing Gaussian random data was filtered through a Gaussian probability density function. By filtering the original Gaussian through another Gaussian allows control over the absolute value of the rate of change of the edge. This is an important feature to have control over, as It allows the testing of highly jagged and serrated edges. Using this method, both standard deviation and curvature can be controlled when generating random edges.

Huygens-Fresnel Algorithm

An incident plane wave was generated in a matrix using the following function:

$$\text{Eincident}=\text{E0}*e\hat{}(-jk\rho*\cos(\phi-\phi\text{incident})) \quad (2.1)$$

This plane wave is independent of the Z direction. The Hugens-Fresel principle evaluated with respect to an incident plane wave on an aperture (2.2) can be thought of as man infinitesimal point sources at the same phase and frequency as the incident wave through the expected aperture. In the case of a half plane with rough edges, this aperture is the distance from the edge out to infinity. Because each infinitesimal point in the aperture contributes to the diffracted field linearly, we can treat each numerically generated point on the Gaussian edge as a slice. In doing so, it is possible to find when contributions from slices stop contributing appreciably to the diffracted field.

$$\text{Ediffracted}=\text{Eincident}(x',y')*(e\hat{}(-j|r-r'|))/(4\pi|r-r'|)dx'dy' \quad (2.2)$$

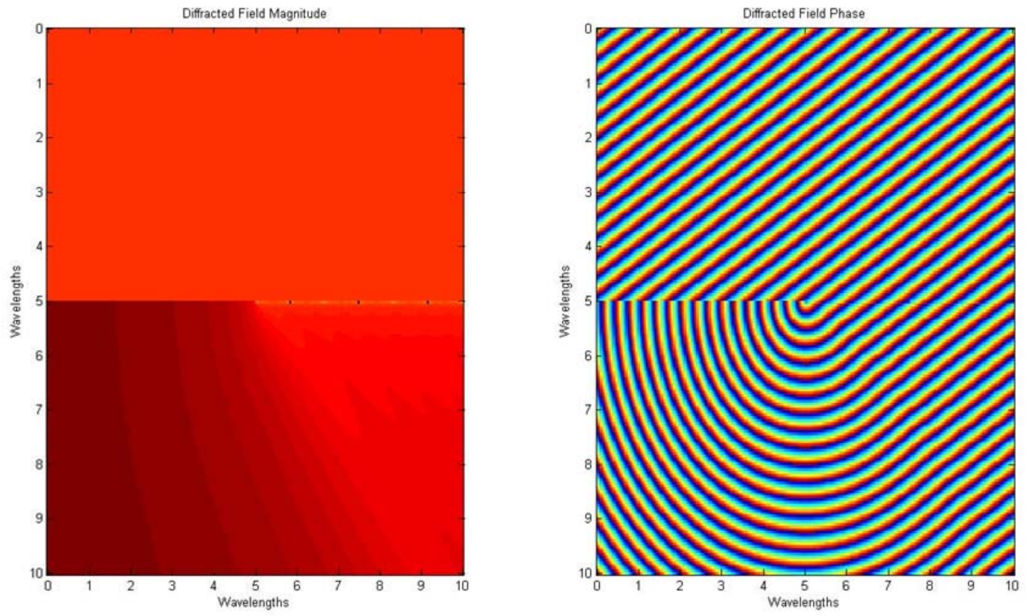
The algorithm uses the incident wave and the generated edge to create a field of points which launch waves that are projected on a central slice in the X Y plane. These field points are computed starting at the center slice, then moving outwardly one slice at a time until the points on the aperture no longer significantly contribute to the

Due to the computational load and time constraints an initial flat edge solution which went out to 500 wavelengths and had points along the aperture at intervals of 1/20 of a wavelength was computed as a baseline. Then the difference between the Flat edge and the roughness was chosen as a new aperture which launched a negative wave which projected onto the Flat Edge solution. This method sped up calculations and simultaneously increased the quality of the result.

III.Results and Conclusions

Diffracted Fields

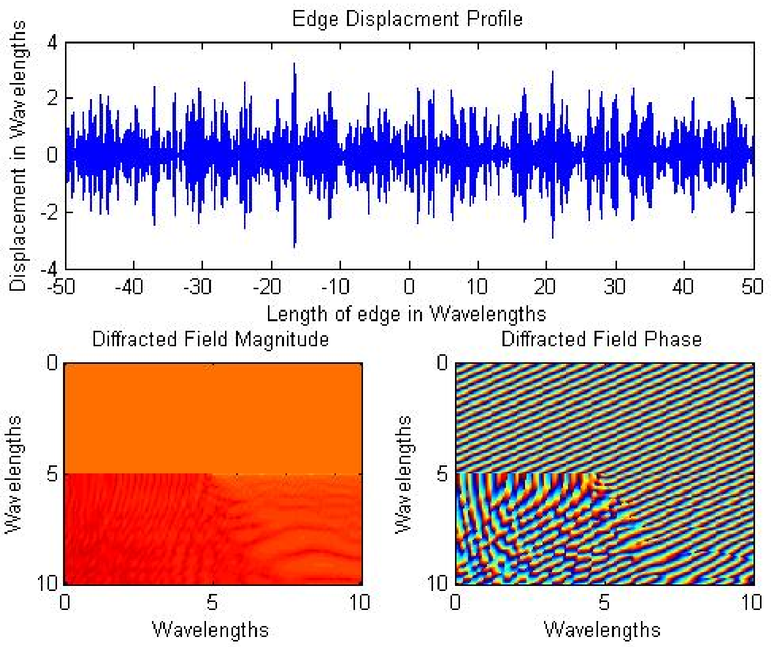
The simulations produced diffracted fields that were noticeably different from the smooth edge counterparts. The smooth edge solution has a predictable and periodic phase which seems to curl around the flat edge. The magnitude of the diffracted wave fades as it moves further into the shadow region. It stays relatively constant when in the unobstructed region as one would expect. Variations in



the edge introduced by a Gaussian noise profile disrupt these smooth features considerably.

Figure 1. The diffracted field of a perfect electric conducting half plane with a flat edge.

Figure 2 shows a disruption from the normally predictably smooth phase taper. The edge in this case has a standard deviation of 1 wavelength. Additionally it has a curvature of 5 wavelengths. In simulations with less roughness the differences in magnitude between the flat edge solution and a rough edge one is smaller than a higher roughness situation. Also, in a situation with low roughness and



high curvature, the features present in the phase seem to be more clear.

Figure 2. A 60 degree incident plane wave on a rough edge with a sigma of 1 and a curvature of 5.

A higher standard deviation seems to increase the choppiness of the phase in the shadow region, it also creates less fine and clear features in the magnitude. When comparing figure 2 and figure 3, the magnitude and phase appear to have less clear cut lines and take on a more blotchy look. In this case both the standard deviation and curvature increased. With a low standard deviation and curvature shown in figure 4, it appears that as the curvature decreases the diffracted image becomes more smooth.

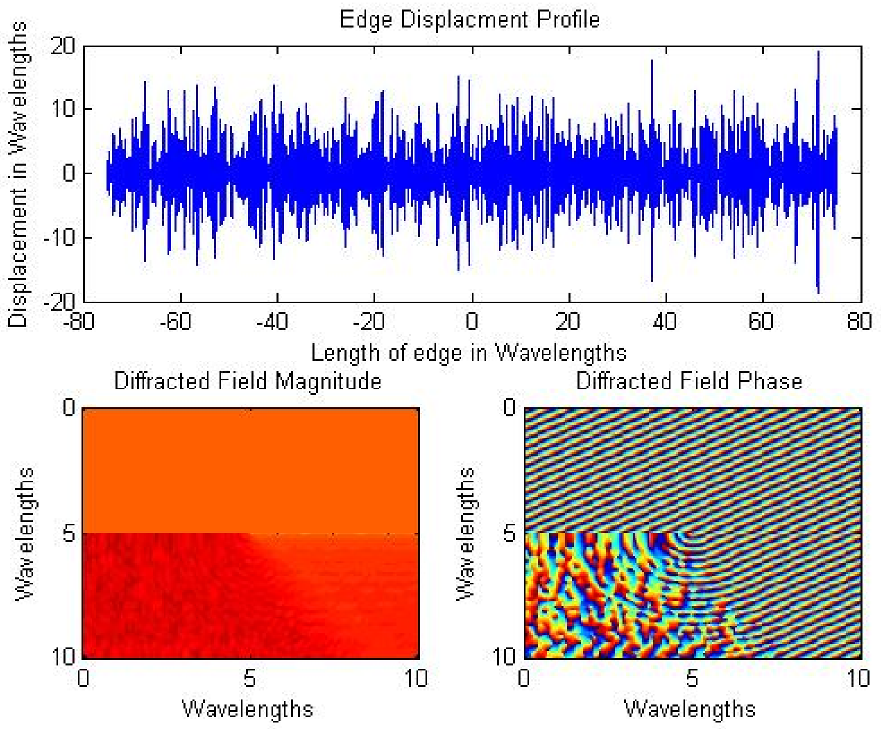


Figure 3. A 60 degree incident plane wave on a rough edge with a sigma of 10 and a curvature of 5.

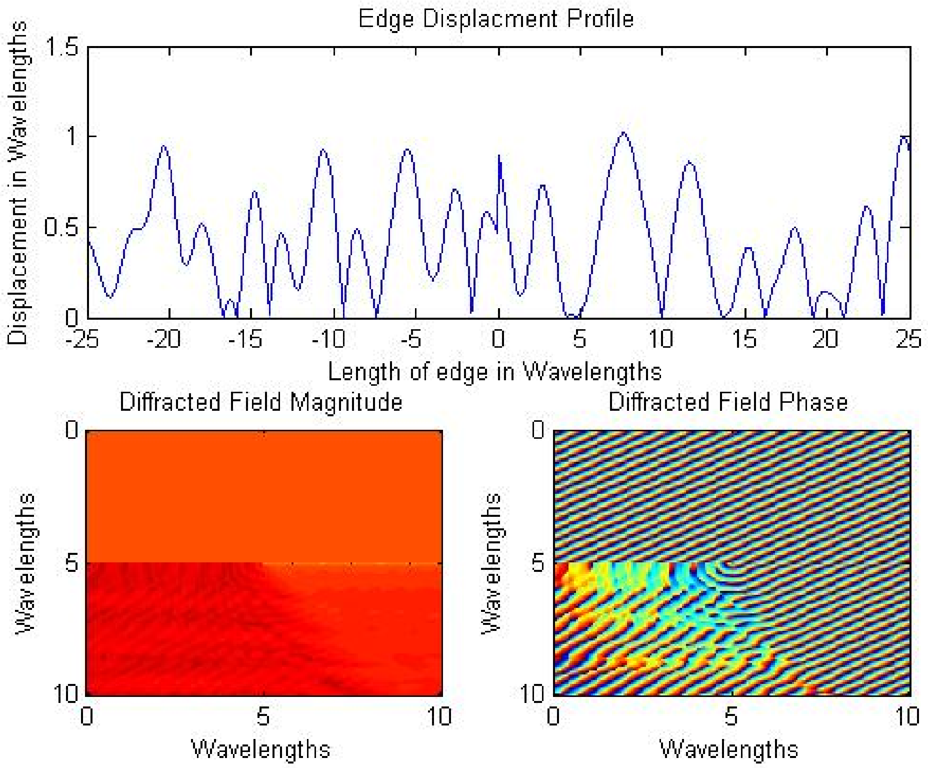
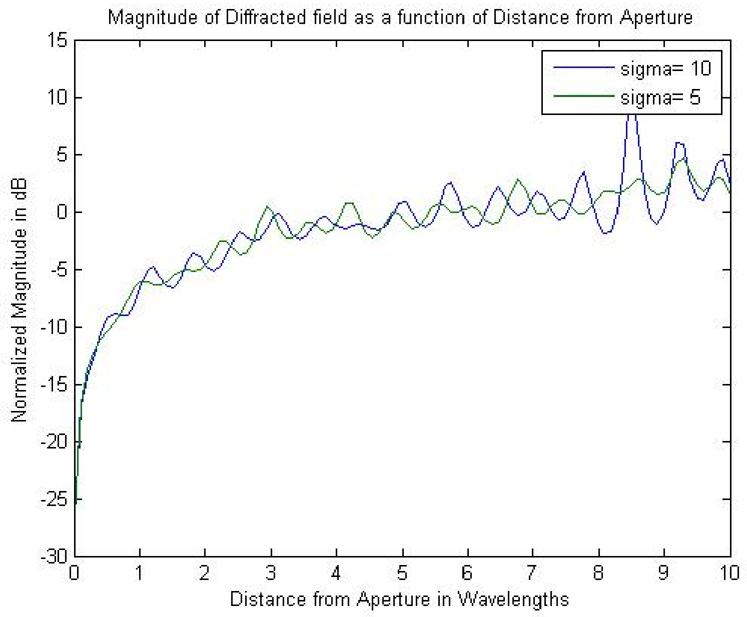


Figure 4. A 60 degree incident angle with a sigma of 0.5 and a curvature of 0.1.

These results appear to conform to what one would expect. In situations with a low roughness and a low change in curvature, there is a close resemblance to flat edge situations. As the roughness increases features from the roughness of the curve are introduced into the diffracted field. The more roughness a surface has the more fine the features in the diffracted field. Though even with high roughness decreasing the curvature can decrease the sharpness of the created features. Again, this makes sense, as curvature approaches zero the edge approaches smooth. As the standard deviation approaches zero the edge also approaches complete smoothness.

Rough edges have a large effect on the diffracted field within 5 wavelengths. Fine features are produced by edges with high standard deviations. Decreases in roughness or curvature appear to increase the smoothness of the features and bring the diffracted field closer to a flat perfectly



conducting edge.

Figure 5. Graph comparing my results to the Davis and Brown paper.

When comparing my results to those generated in the Davis and Brown paper, they appear to conform to a certain degree. In the Davis plot a smooth edge has a Fresnel Integral function power and it decreases with increasing roughness. This function does not directly conform to a Fresnel integral function, however it does smooth with an increasing roughness. Which is the expected behavior. This change may be due to the fact that the source in the paper was a beam and in this report, I use plane waves incident on the aperture. This appears to loosely confirm my results, though some more concerns should lead to additional testing before submitting for possible publication.

IV.Future Work and Concerns

In order to follow up with the work done, the next step is to use the Sommerfeld solution and modify it to compute the diffracted field solution. Due to the inaccuracies of the near field knife edge diffraction solution, it is unreliable under about 5 wavelengths in the shadow region. Therefore, a Sommerfeld solution would provide a rigorous solution to the problem.

This code also has the potential to be able to produce much higher resolution images with the addition of parallel processing. This could be achieved easily with a single instruction multiple data (SIMD) architecture. Currently the NVIDIA CUDA platform is offering easy access to this architecture using their GPUs. Significant speed ups in the code could be achieved due to the linearity of the algorithm making this an ideal candidate for such a switch.

Concerns

There are some lingering concerns I have about the validity of my results. While the phase variations in the shadow region look expected, the magnitude variations should be smoother. After running several simulations for extended periods of time, I continued to find this behavior. I believe it may be corrected by correctly determining the step between slices in the Z direction. This problem also effects aliasing of my edge generation. However after testing and retesting, it may just be part of a correct solution. More verification is needed.

Also when looking at a randomly generated edge, it seems to not be stationary. There are points which appear to have different statistics along the curve. In order for this work to be publishable, these concerns must be addressed and corrected. Further examination of the edge generation algorithm is necessary. I have spent a large part of the past several days working on it, and so far have been unable to find a satisfactory solution. Though it seems there is simply an osculation every other point and when tracing out the envelope of the function the correct edge profile appears to be created.

Apendix A

Simulation Code

```
%========================================================================%
% KED Rough Edge Field Simulation                                        %
% Author: Zak Costello                                                   %
%                                                                        %
% Description: Generates a Random Gausian Edge and uses the data to      %
% calculate a diffracted field solution.                                 %
%                                                                        %
% File Dependencies:                                                     %
%========================================================================%

clc
clear
close all

%------------------------------------------------------------------------%
% 1. Create Simulation Grid                                              %
%------------------------------------------------------------------------%

%Parameters
GridSize       =   10;    %XY cross sectionIn Wavelengths
GridLength     =   5;    %Z length of Grid in Wavelengths
GridResolution =   0.05;    %Minimum feature size In Wavelengths
GridZ = 1;

%Generate Grid
GridPoints= GridSize/GridResolution;
[GridX GridY] =
meshgrid(linspace(0,GridSize,GridPoints),linspace(0,GridSize,GridPoints));
GridRho = sqrt(GridX.^2+GridY.^2);
GridPhi = atan2(GridY,GridX);

%------------------------------------------------------------------------%
% 2. Generate Incident Wave                                              %
%------------------------------------------------------------------------%
% TODO: Add Polarization Effects

%Parameters
Eo = 8.85418*10^-12;           %Permitivity
wavelength  =      5;          %In mm
IncidentAngle =    90*pi/180;  %In Radians

%Generate Incident Wave
k = 1/GridResolution;
E_incident = @(rho,phi) Eo .* exp(1i.*k.*rho.*cos(phi-IncidentAngle));
E = E_incident(GridRho,GridPhi);

%------------------------------------------------------------------------%
% 3. Generate Rough Edge                                                 %
%------------------------------------------------------------------------%

%Parameters
EdgeRoughness  =   5;      %StdDev of Edge Roughness In WaveLength
EdgeCurvyness  =   20;     %Max rate of change of Edge In WaveLength

EdgeRoughness = EdgeRoughness/GridResolution;
```

```
EdgeCurvyness = EdgeCurvyness/GridResolution;

%Generate Rough Edge
Edge = makeEdge(GridLength,EdgeRoughness,EdgeCurvyness,100);
plot(Edge)

%-------------------------------------------------------------------------%
% Flat Edge Baseline Slice                                                %
%-------------------------------------------------------------------------%

% check to see if data for this exists and use it if it does
if exist('FlatEdgeSlice90Wav.mat','file')
    load 'FlatEdgeSlice90Wav.mat'
else
    Yp = GridSize/2;
    Z = 0;
    SOL = zeros(GridPoints/2,GridPoints);
    [ X Y ] =
meshgrid(linspace(0,GridSize,GridPoints),linspace(GridSize/2,GridSize,GridPoi
nts/2));
    count = 1;
    for Xp = linspace(GridSize/2,GridSize*50,40000)
        KED = E_incident(sqrt((Xp).^2+ (Yp).^2), atan2(Yp,Xp))  .*
exp(1i*k*sqrt((X-Xp).^2+(Y-Yp).^2+Z.^2)) ./ (4*pi*sqrt((X-Xp).^2+(Y-
Yp).^2+Z.^2));
        SOL = SOL + sum(KED,3)*GridResolution;
        imagesc(angle([E(1:GridPoints/2,:); SOL]));
        %pause(0.5)
        disp(count);
        count = count+1;
    end
    save('FlatEdgeSlice90Wav','SOL');
end
FlatEdge = SOL;

%-------------------------------------------------------------------------%
% Slice Algorithm                                                         %
%-------------------------------------------------------------------------%

tic
%Generate initial Edge of sufficient length
EdgeL = makeEdge(GridZ*500,EdgeRoughness,EdgeCurvyness,500);
EdgeR = makeEdge(GridZ*500,EdgeRoughness,EdgeCurvyness,500);
SOL = zeros(GridPoints/2,GridPoints);
SOL = FlatEdge;
DisplacementL = EdgeL + GridSize/2;
DisplacementR = EdgeR + GridSize/2;

TotalEdge = [EdgeL EdgeR];

Yp = GridSize/2;
X = GridX(GridPoints/2+1:GridPoints,:);
Y = GridY(GridPoints/2+1:GridPoints,:);

count = 1;
```

```
countL = 1;
countR = 1;
EdgesL = 0;
EdgesR = 0;
side = 0;
Slice = 1;
SliceV = [];
while abs(median(median(Slice))) > 7*10^-19
   if side==0
       D = DisplacementL(countL);
       side = 1;
       Zp = countL * GridZ + 500 * EdgesL;
       countL = countL + 1;
       if countL == 500
           EdgeL = makeEdge(GridZ*500,EdgeRoughness,EdgeCurvyness,500);
           countL = 1;
           EdgesL = EdgesL + 1;
           TotalEdge = [EdgeL TotalEdge];
       end
   else
       D = DisplacementR(countR);
       %SOL = SOL + FlatEdge;
       side = 0;
       Zp = countR * GridZ  + 500 * EdgesL;
       countR= countR + 1;
       if countR == 500
           EdgeR = makeEdge(GridZ*500,EdgeRoughness,EdgeCurvyness,500);
           countR = 1;
           EdgesR = EdgesR + 1;
           TotalEdge = [TotalEdge EdgeR];
       end
   end

   count = count + 1;
   disp(count*GridZ);
   disp(num2str(abs(median(median(Slice)))))

   Points = round((GridSize - D)/GridResolution);
   Points = 100;
   [X Y Xp] = 
meshgrid(linspace(0,GridSize,GridPoints),linspace(GridSize/2,GridSize,GridPoi
nts/2),linspace(D,GridSize/2,Points));
   KED = E_incident(sqrt((Xp).^2+ (Yp).^2), atan2(Yp,Xp))  .* 
exp(1i*k*sqrt((X-Xp).^2+(Y-Yp).^2+(Zp).^2)) ./ (4*pi*sqrt((X-Xp).^2+(Y-
Yp).^2+(Zp).^2));
   Slice = sum(KED,3)*GridResolution;
   imagesc(angle([E(1:GridPoints/2,:); Slice]));
   SOL = SOL - (Slice);
   imagesc(angle([E(1:GridPoints/2,:); SOL]));
   %SliceV = [SliceV sum(sum(abs(Slice)))];
end
toc

%-------------------------------------------------------------------------%
% 5. Format and Display Results                                           %
%-------------------------------------------------------------------------%
```

```
figure
subplot(2,2,1:2)
plot(TotalEdge)

subplot(2,2,3)
imagesc(-20*log10(abs([E(1:GridPoints/2,:); SOL])));

subplot(2,2,4)
imagesc(angle([E(1:GridPoints/2,:); SOL]
```