Table of Contents

Fresh Start

Author: Evan Krimpenfort Class: ECE 563-01 Purpose: Project Chapter 6

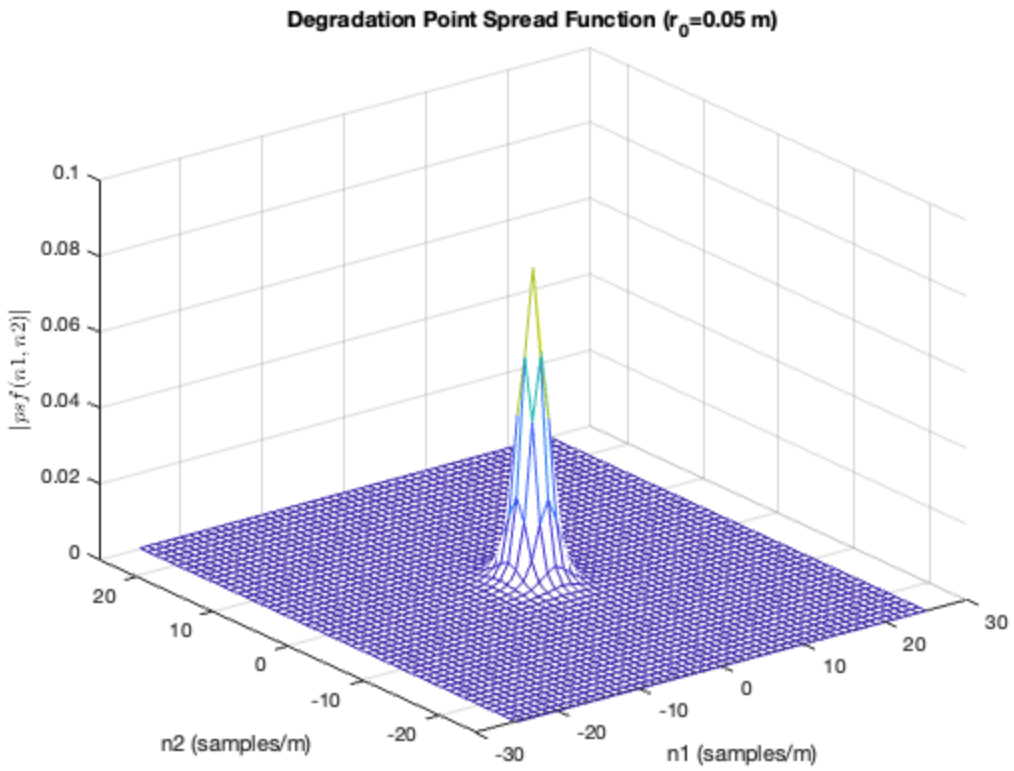
```
clc; clear all; close all;
```

Ideal Image

```
in = double(imread('Dream-Factory.JPEG'));  % Read image
in = double(in(:,:,2));                     % Make it gray
in = imresize(in, 0.4);                     % resize to get in the 255
 range

im(in); title('Input Image');
```

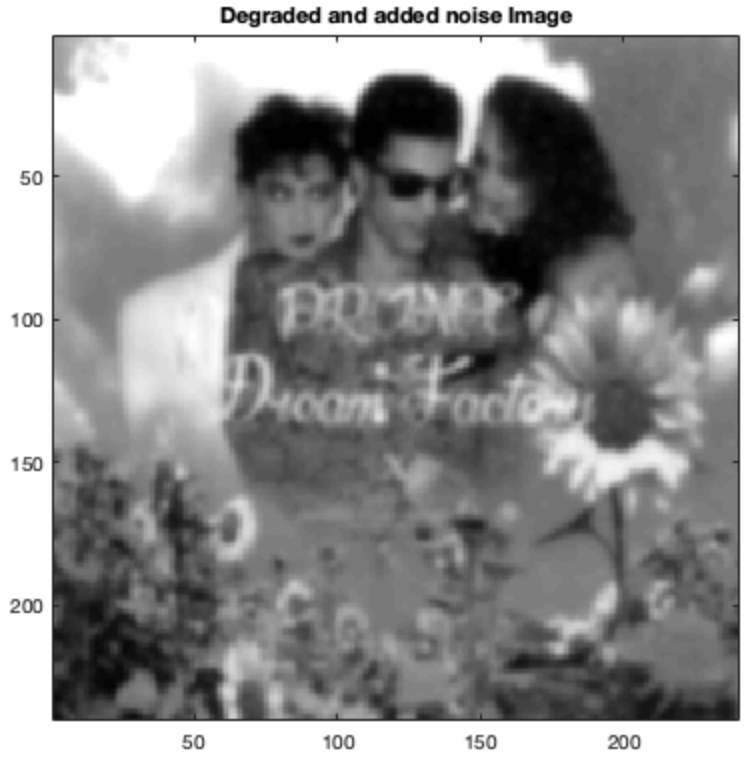
Mesh plot of the PSF

```
% setting up params for the psf
params.fnumber = 3;
params.focal_length = 150e-3;   % meters
params.wavelength = 0.5e-6;     % meters
params.rho_c = 1/(params.wavelength*params.fnumber); % Cutoff
 frequency
rho_s = 2*params.rho_c;         % sampling frequency
params.a = 1/rho_s;
params.b = 1/rho_s;
params.r0 = 0.05;       % SPECIFIED
params.dx = params.a;
params.dy = params.b;
params.M = 51;
params.N = 51;

out = model_psf( params );  % process the filter

% Overall PSF
figure;
mesh(out.n1, out.n2, out.psf);
title(sprintf('Degradation Point Spread Function (r_0=%2.2f m)',
 params.r0));
xlabel('n1 (samples/m)');
ylabel('n2 (samples/m)');
zlabel('$|psf(n1, n2)|$', 'interpreter', 'latex');
```



Degraded Image

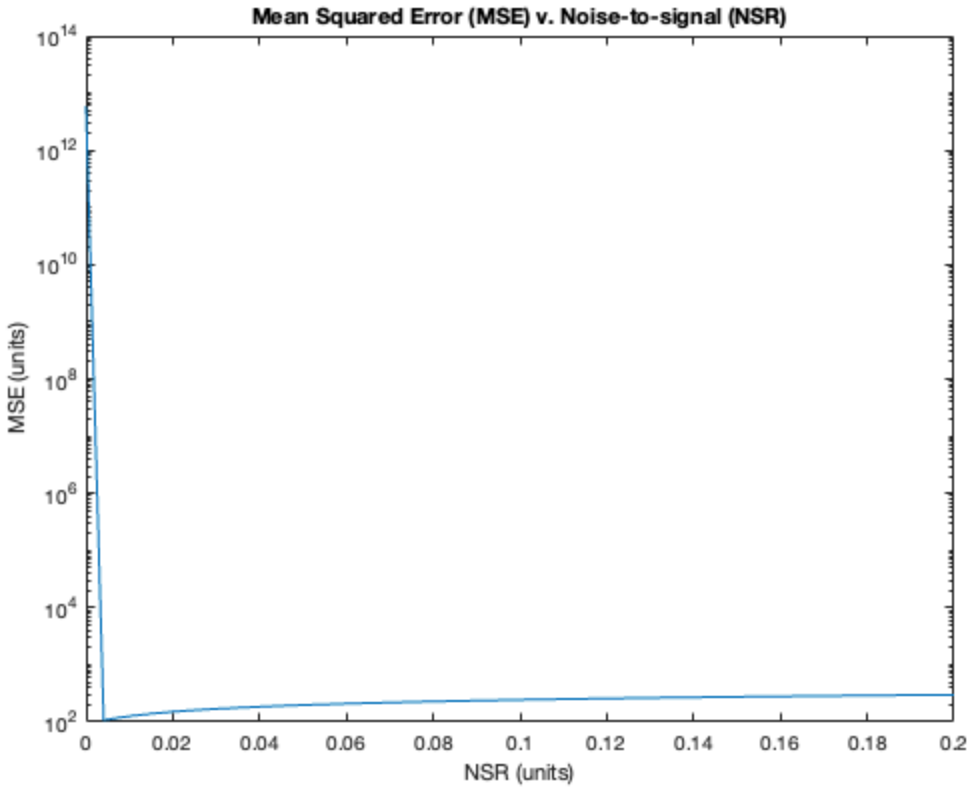
```
% size of psf
[L, K] = size(out.psf);

% padding of the psf
xpad = (K - 1)/2;
ypad = (L - 1)/2;

% pad array
psfpad = padarray(in, [xpad, ypad], 'symmetric', 'both');
y = conv2(psfpad, out.psf, 'valid');    % degrade through convolution

% Add gaussian noise with a standard deviation of 1
standard_deviation = 1;
y2 = y + randn(size(y)) * standard_deviation;

im(y); title('Degraded and added noise Image');
```



MSE v. NSR

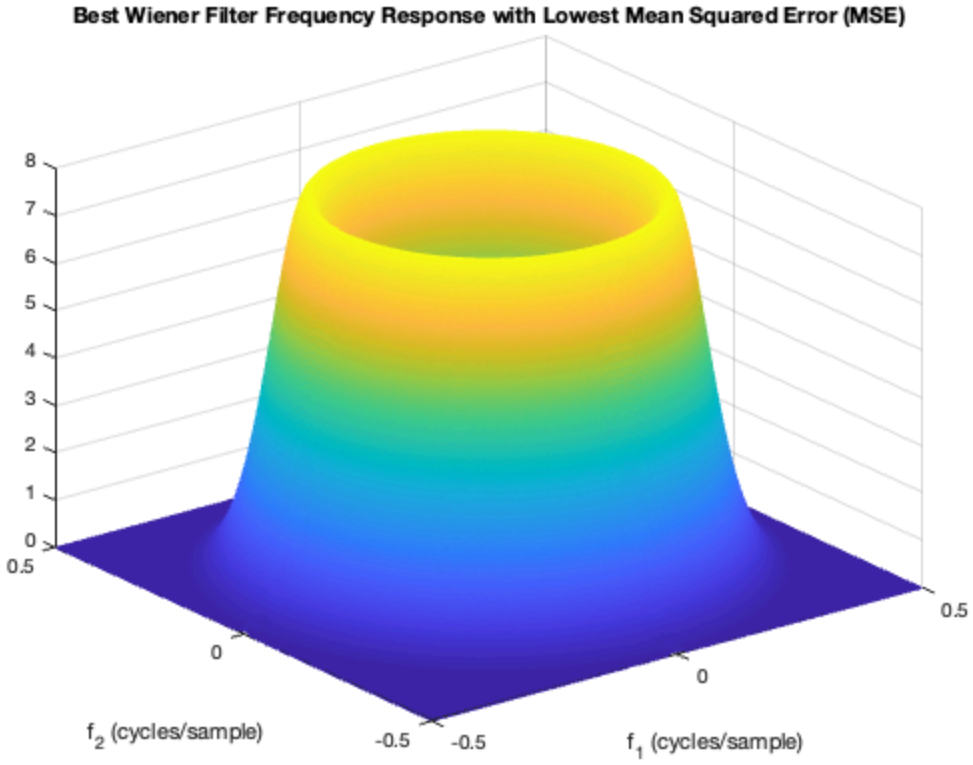
```
% constants
% array of Noise-to-singal values
% 0, 0.1, 100 was too large
% 0, 0.05, 50 was too small
% below seemed the best
NSR_array = linspace(0,0.2,50);

% Default pad size
pad = 100;

% empty array of Mean Standard Errors
% will be filled
MSE_array = zeros(1, length(NSR_array));

for i = 1:length(NSR_array)
   out2 = wiener_filter_2d(y2, out.psf, NSR_array(i), pad); % filter
   err = dif(in, out2.img); % find the error between the ideal and
 filtered
   MSE_array(i) = err.mse;  % store the mse value
end

% graph MSE v. NSR
figure;
```

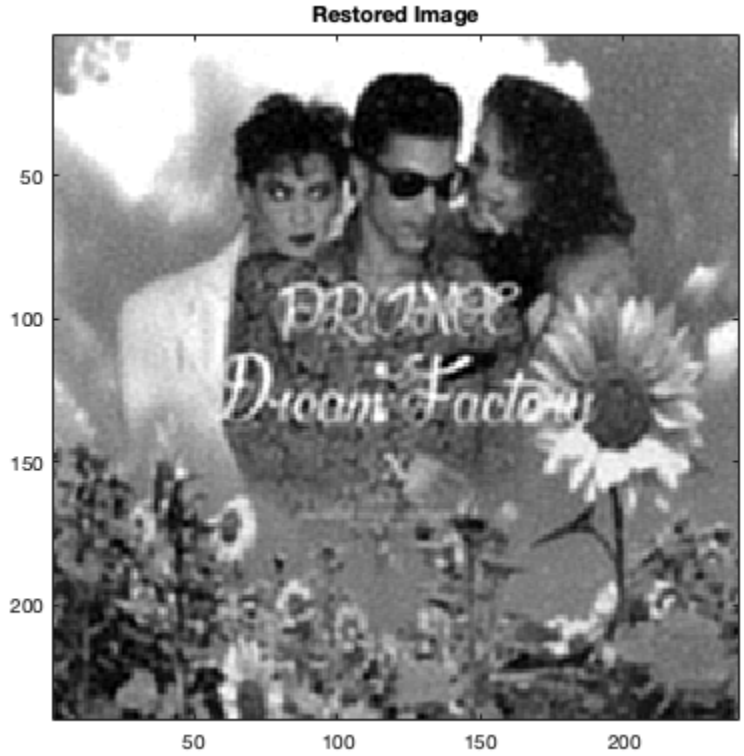
```
semilogy(NSR_array, MSE_array)
title('Mean Squared Error (MSE) v. Noise-to-signal (NSR)')
xlabel('NSR (units)')
ylabel('MSE (units)')

% Best MSE value
[min_value, index] = min(MSE_array);

% Best NSR index
best_NSR_value = NSR_array(index);
```



Best Wiener Filter Frequency Response

```
% filter with the best MSE index
out3 = wiener_filter_2d(y2, out.psf, best_NSR_value, pad);

% Plot the best Wiener filter frequency response
figure;
mesh(out3.f1, out3.f2, out3.HW_mag)
title('Best Wiener Filter Frequency Response with Lowest Mean Squared
 Error (MSE)')
xlabel('f_1 (cycles/sample)');
ylabel('f_2 (cycles/sample)');

im(out3.img); title('Restored Image');
```


Best Wiener Filter Frequency Response with Lowest Mean Squared Error (MSE)
8
7
6
5
4
3
2
1
0
0.5
0
-0.5
f2 (cycles/sample)
-0.5
0
0.5
f1 (cycles/sample)


Restored Image
50
100
150
200
50
100
150
200

Final error measurements

```
% MAE and MSE
err2 = dif(in, out3.img);
MAE = err2.mae;
MSE = err2.mse;

% SSIM
SSIM = ssim(in ,out3.img);

% list the erros
fprintf('The best NSR value is an NSR = %f\n', best_NSR_value)
fprintf('The Mean Absolute Error (MAE) for the best NSR value is %f
\n', MAE)
fprintf('The Mean Squared Error (MSE) for the best NSR value is %f\n',
 MSE)
fprintf('The SSIM for the best NSR is %f\n',SSIM)

% end of run_wiener_filter_2d.m
```

```
The best NSR value is an NSR = 0.004082
The Mean Absolute Error (MAE) for the best NSR value is 7.057613
The Mean Squared Error (MSE) for the best NSR value is 107.743929
The SSIM for the best NSR is 0.784438
```

Published with MATLAB® R2020a