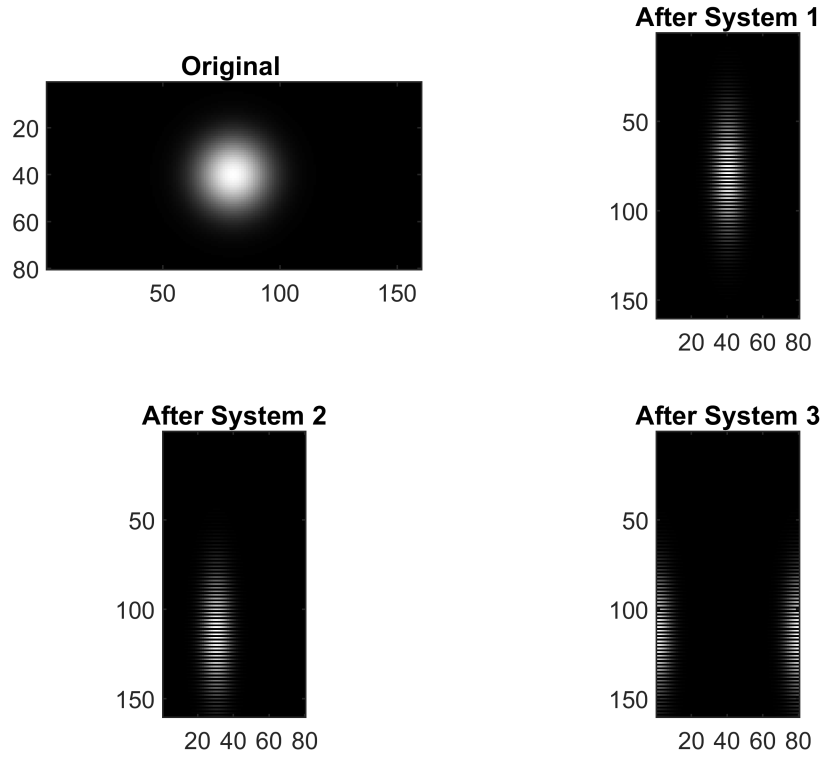
%%

Exercise 3.1

COMMENT Section

```
%%
% USER DEFINED VARIABLES
w = 15; % Width
x = 1:160; % Horiztonal Axis
y = 1:80; % Vertical Axis
% ==> Create the 'original' matrix with weights up to 127 <==
z = round (127* exp(-1/w.^2*((y.'-40).^2+(x-80) .^2)));
% ==> Operate on the image recursively <==
[xs ,ys ,zs] = image_system1(z, 2, 2);
za = image_system2(zs, -10, 35);
zb = image_system3(za, 50, 0);
% PLOT RESULT WITH SUBPLOT
figure (1);
subplot (2,2,1); % ==> Create a new plot on the 2x2 window matrix in position 1 <==
imagesc(x, y, z); % ==> Put the image on the plot <==
axis image; % ==> Aligns the image to the plot <==
title("Original ")
subplot (2,2,2); % ==> Create a new plot on the 2x2 window matrix in position 2 <==
imagesc(xs , ys , zs); % ==> Put the image on the plot <==
axis image; % ==> Aligns the image to the plot <==
title("After System 1")
subplot (2,2,3); % ==> Create a new plot on the 2x2 window matrix in position 3 <==
imagesc(xs , ys , za); % ==> Put the image on the plot <==
axis image; % ==> Aligns the image to the plot <==
title("After System 2")
subplot (2,2,4); % ==> Create a new plot on the 2x2 window matrix in position 4 <==
imagesc(xs , ys , zb); % ==> Put the image on the plot <==
axis image; % ==> Aligns the image to the plot <==
title("After System 3")
% ==> Create the 'image' in greyscale <==
colormap(gray);
```




```
%Functions image_system<X> defined and explanations at bottom of script
```

Exercise 3.2

SAMPLE section

A. Function at bottom of script

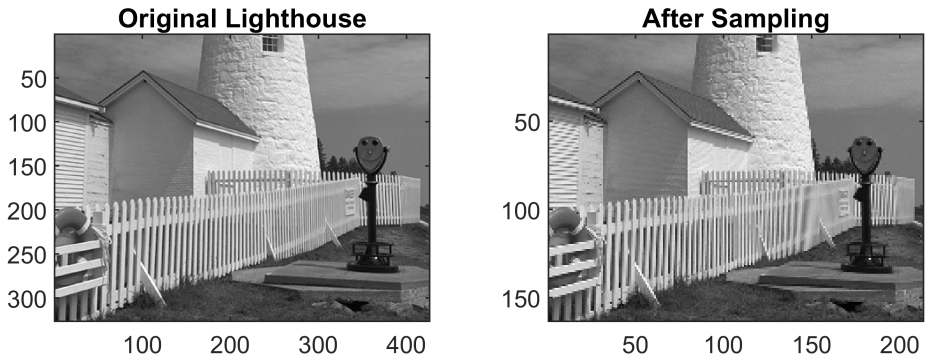
B.

```
[xs ,ys ,lighthouse_sampled] = image_sample(lighthouse, 2);
```

```
zs = 163×213
    0.4863    0.5059    0.4980    0.4941    0.4863    0.4824    0.4784    0.4824 ···
    0.4667    0.4863    0.4824    0.4863    0.4824    0.4863    0.4824    0.4667
    0.4824    0.4706    0.4745    0.4824    0.4863    0.4941    0.4784    0.4824
    0.4745    0.4784    0.4824    0.4824    0.4784    0.4784    0.4941    0.4667
    0.4784    0.4745    0.4745    0.4667    0.4588    0.4667    0.4588    0.4627
    0.4824    0.4745    0.4745    0.4667    0.4510    0.4392    0.4471    0.4392
    0.4667    0.4902    0.4627    0.4627    0.4588    0.4471    0.4549    0.4471
    0.4392    0.4588    0.4588    0.4784    0.4588    0.4431    0.4627    0.4588
    0.4275    0.4196    0.4471    0.4353    0.4431    0.4392    0.4392    0.4392
    0.4275    0.4431    0.4431    0.4314    0.4275    0.4353    0.4353    0.4471
      ⋮
```

```
clf;
figure (3);
subplot (1,2,1); % ==> Create a new plot on the 2x2 window matrix in position 1 <==
imagesc(lighthouse); % ==> Put the image on the plot <==
```

```
axis image; % ==> Aligns the image to the plot <==
title("Original Lighthouse")
subplot (1,2,2); % ==> Create a new plot on the 2x2 window matrix in position 2 <==
imagesc(lighthouse_sampled); % ==> Put the image on the plot <==
axis image; % ==> Aligns the image to the plot <==
title("After Sampling")
% ==> Create the 'image' in greyscale <==
colormap(gray);
```



It is clear to see that sampling causes the image to pixelate due to aliasing. Entities are difficult to make out in high detail, such as the binoculars.

This relates to the class topics because when a signal is sampled at too large of a sample, it can be difficult to determine the original signal. For example if a sinusoidal signal is sampled every x and the period of the signal is x, the result of the sampling is the same value for every sample.

ANTI ALIAS Section

C. Function at bottom of script

D.

```
% ==> Operate on the image recursively <==
lighthouse_aaxn = lighthouse;
for i = 1:6
    lighthouse_aaxn = image_antialias(lighthouse_aaxn);
```

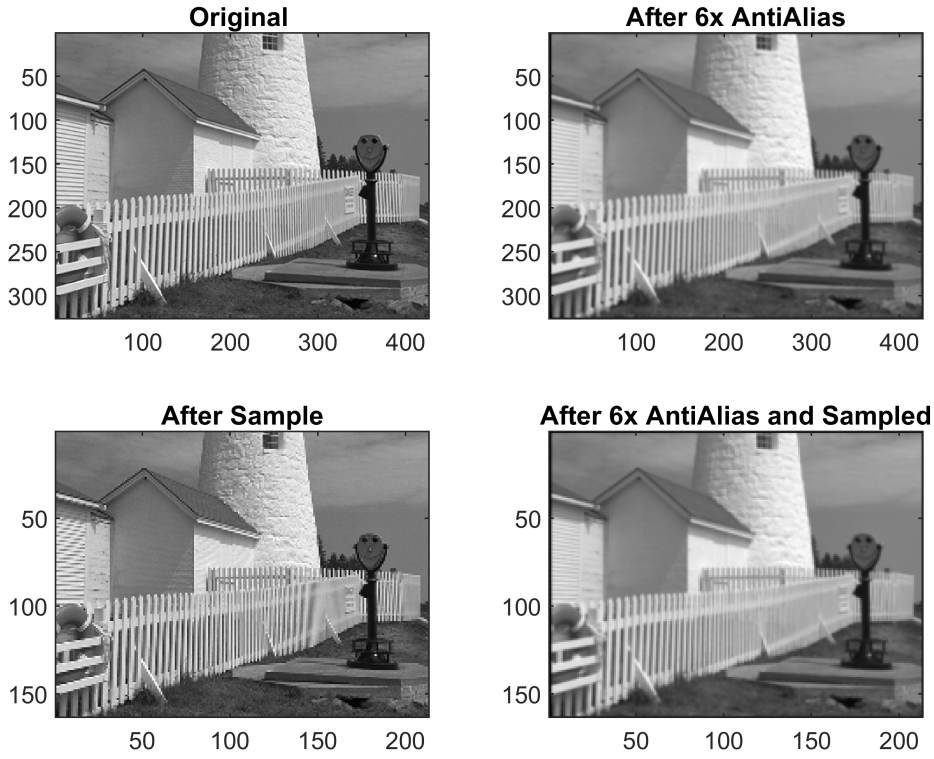
```
end
lighthouse_aax6 = lighthouse_aaxn;
[xs ,ys ,lighthouse_sampled] = image_sample(lighthouse, 2);
```

```
zs = 163×213
    0.4863    0.5059    0.4980    0.4941    0.4863    0.4824    0.4784    0.4824 ···
    0.4667    0.4863    0.4824    0.4863    0.4824    0.4863    0.4824    0.4667
    0.4824    0.4706    0.4745    0.4824    0.4863    0.4941    0.4784    0.4824
    0.4745    0.4784    0.4824    0.4824    0.4784    0.4784    0.4941    0.4667
    0.4784    0.4745    0.4745    0.4667    0.4588    0.4667    0.4588    0.4627
    0.4824    0.4745    0.4745    0.4667    0.4510    0.4392    0.4471    0.4392
    0.4667    0.4902    0.4627    0.4627    0.4588    0.4471    0.4549    0.4471
    0.4392    0.4588    0.4588    0.4784    0.4588    0.4431    0.4627    0.4588
    0.4275    0.4196    0.4471    0.4353    0.4431    0.4392    0.4392    0.4392
    0.4275    0.4431    0.4431    0.4314    0.4275    0.4353    0.4353    0.4471
     ⋮
```

```
[xs ,ys ,lighthouse_aax6_sampled] = image_sample(lighthouse_aax6, 2);
```

```
zs = 163×213
         0         0         0         0         0         0         0         0 ···
         0    0.2621    0.4331    0.4416    0.4416    0.4412    0.4348    0.4287
         0    0.2901    0.4760    0.4850    0.4858    0.4864    0.4813    0.4759
         0    0.2902    0.4736    0.4789    0.4789    0.4800    0.4811    0.4781
         0    0.2899    0.4682    0.4688    0.4663    0.4659    0.4661    0.4641
         0    0.2898    0.4659    0.4642    0.4561    0.4510    0.4492    0.4460
         0    0.2866    0.4618    0.4632    0.4554    0.4478    0.4499    0.4479
         0    0.2745    0.4516    0.4612    0.4533    0.4461    0.4516    0.4495
         0    0.2632    0.4355    0.4455    0.4432    0.4403    0.4435    0.4393
         0    0.2653    0.4333    0.4371    0.4370    0.4387    0.4406    0.4385
     ⋮
```

```
% PLOT RESULT WITH SUBPLOT
clf;
figure (4);
subplot (2,2,1); % ==> Create a new plot on the 2x2 window matrix in position 1 <==
imagesc(lighthouse); % ==> Put the image on the plot <==
axis image; % ==> Aligns the image to the plot <==
title("Original ")
subplot (2,2,2); % ==> Create a new plot on the 2x2 window matrix in position 2 <==
imagesc(lighthouse_aax6); % ==> Put the image on the plot <==
axis image; % ==> Aligns the image to the plot <==
title("After 6x AntiAlias")
subplot (2,2,3); % ==> Create a new plot on the 2x2 window matrix in position 2 <==
imagesc(lighthouse_sampled); % ==> Put the image on the plot <==
axis image; % ==> Aligns the image to the plot <==
title("After Sample")
subplot (2,2,4); % ==> Create a new plot on the 2x2 window matrix in position 2 <==
imagesc(lighthouse_aax6_sampled); % ==> Put the image on the plot <==
axis image; % ==> Aligns the image to the plot <==
title("After 6x AntiAlias and Sampled")% ==> Create the 'image' in greyscale <==
colormap(gray);
```



The antialiasing filter blurs the image such that everything is still recognizable but slightly less detailed, it reduces aliasing by making the changes across pixels much less pronounced. It is useful to find similarities in images (especially in artificial intelligence) where the tiny-scale details are not important.

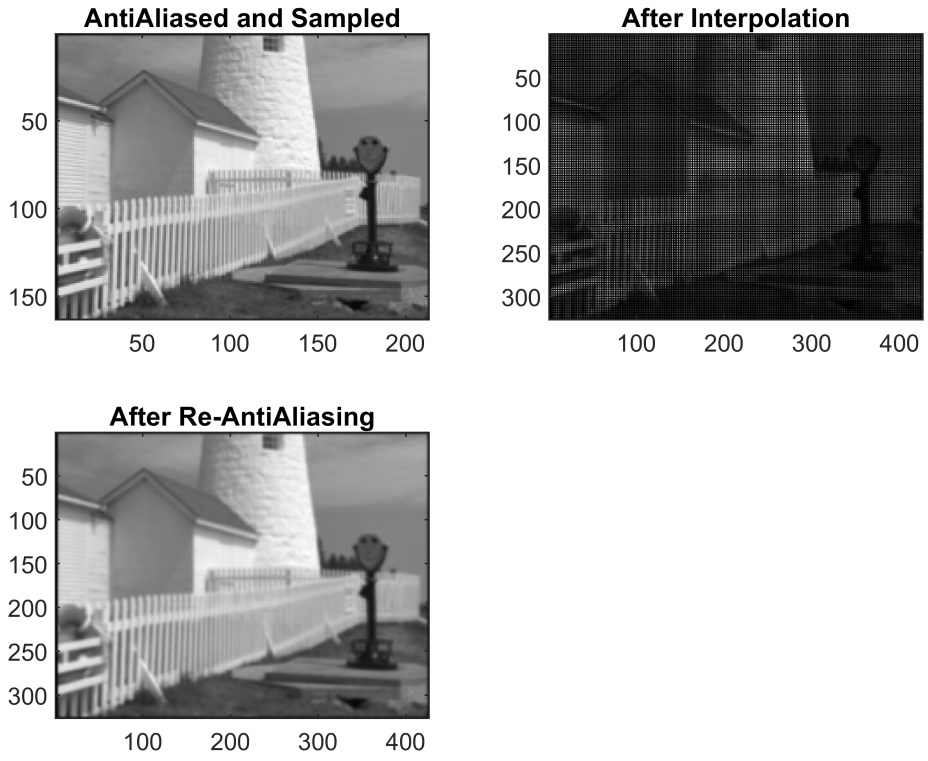
INTERPOLATION Section

E. Function at bottom of script

F.

```
[xs ,ys ,lighthouse_zeros] = image_insertzeros(lighthouse_aax6_sampled, 2);
lighthouse_aaxn = lighthouse_zeros;
for i = 1:7
    lighthouse_aaxn = image_antialias(lighthouse_aaxn);
end
lighthouse_interpolated = lighthouse_aaxn;
% PLOT RESULT WITH SUBPLOT
clf;
figure(5);
subplot (2,2,1); % ==> Create a new plot on the 2x2 window matrix in position 1 <==
imagesc(lighthouse_aax6_sampled); % ==> Put the image on the plot <==
axis image; % ==> Aligns the image to the plot <==
title("AntiAliased and Sampled")
subplot (2,2,2); % ==> Create a new plot on the 2x2 window matrix in position 2 <==
imagesc(xs, ys, lighthouse_zeros); % ==> Put the image on the plot <==
axis image; % ==> Aligns the image to the plot <==
title("After Interpolation")
```

```
subplot (2,2,3); % ==> Create a new plot on the 2x2 window matrix in position 2 <==
imagesc(lighthouse_interpolated); % ==> Put the image on the plot <==
axis image; % ==> Aligns the image to the plot <==
title("After Re-AntiAliasing")
colormap(gray);
```



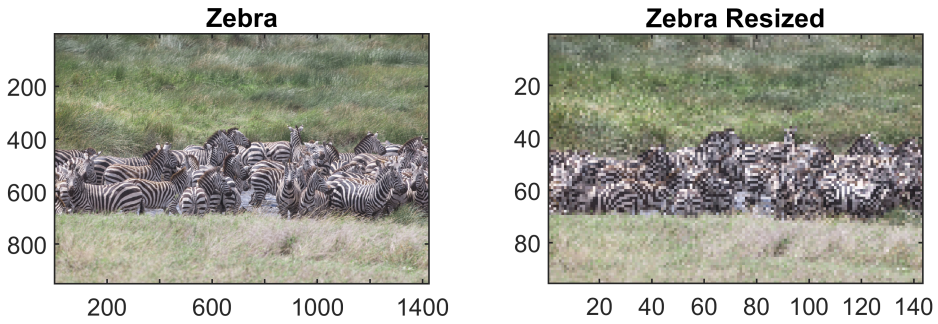
The dimensions of lighthouse_interpolated is the same size as the original lighthouse, which is double of the lighthouse_aax6_sampled, which is what we started with at the beginning of part F. The interpolation expands the size of the image, combining it with the antialiasing removes the disruptive black seperators that are used to add size. Simply put, this is useful in real world application to expand the size of any photo while preserving quality.

EXTRA CREDIT Section

B. Function at bottom of script

```
[xs,ys,zs] = rgbimage_sample(zebra,10);
% PLOT RESULT WITH SUBPLOT
clf;
figure(7);
subplot (1,2,1); % ==> Create a new plot on the 2x2 window matrix in position 1 <==
imagesc(zebra); % ==> Put the image on the plot <==
axis image; % ==> Aligns the image to the plot <==
title("Zebra")
subplot (1,2,2); % ==> Create a new plot on the 2x2 window matrix in position 1 <==
imagesc(zs); % ==> Put the image on the plot <==
axis image; % ==> Aligns the image to the plot <==
```

```
title("Zebra Resized")
```



D. Function at bottom of script

```
zebraaa = zebra;
for i = 1:60
    zebraaa = rgbimage_antialias(zebraaa);
end
% PLOT RESULT WITH SUBPLOT
figure(8);
subplot (2,2,1); % ==> Create a new plot on the 2x2 window matrix in position 1 <==
imagesc(zebra); % ==> Put the image on the plot <==
axis image; % ==> Aligns the image to the plot <==
title("Zebra")
subplot (2,2,2); % ==> Create a new plot on the 2x2 window matrix in position 1 <==
imagesc(zebraaa); % ==> Put the image on the plot <==
axis image; % ==> Aligns the image to the plot <==
title("Zebra AA")
```

E.

```
[xs ,ys ,zebra_zeros] = rgbimage_insertzeros(zebra, 2);

% PLOT RESULT WITH SUBPLOT
figure(10);
subplot (2,2,1); % ==> Create a new plot on the 2x2 window matrix in position 1 <==
imagesc(zebra); % ==> Put the image on the plot <==
```

```
axis image; % ==> Aligns the image to the plot <==
title("Original")
subplot (2,2,2); % ==> Create a new plot on the 2x2 window matrix in position 2 <==
imagesc(zebra_zeros); % ==> Put the image on the plot <==
axis image; % ==> Aligns the image to the plot <==
title("After Interpolation")
```

FUNCTION DEFINITIONS

```
function [xs, ys, zs] = image_system1(z,Dx,Uy)
    %IMAGE_SYSTEM1 ===> Transposes and scales the image <===
    % ==> Creeates matrix of the image dimension with all values equal to zero <==
    zs = zeros(ceil(Uy*size(z,1)),ceil(size(z,2)/Dx));
    % ==> Axis indexes, scale y axis by Uy, scale x axis by 1/Dx  <==
    ys = 1:ceil(Uy*size(z,1));
    xs = 1:ceil(size(z,2)/Dx);
    % ==> Copy z to zs step rows by Uy on zs and columns by Dx on z <==
    zs(1:Uy:end ,1:end) = z(1:end ,1:Dx:end);
end
function [za] = image_system2(z,Sx,Sy)
%IMAGE_SYSTEM2 ===> Shifts image by Sx on x axis and Sy on y axis <===
    %===> Creates an identical sized frame as the input image <===
    za = zeros(size(z,1), size(z,2));
    for nn = 1:size(z,1)
        for mm = 1:size(z,2)
        % ====> Checks that the given values are valid and in bound <====
            if nn>Sy && nn-Sy <= size(z,1) && mm > Sx && mm - Sx <=size(z,2)
                % ====> Copies the old images data to the new frame <====
                za(nn ,mm) = z(nn -Sy ,mm -Sx);
            end
        end
    end
end
function [zb] = image_system3(za,Sx,Sy)
    %IMAGE_SYSTEM3 ===> Shifts the image while allowing overflow to enter on the opposite side
    % ====> Create iterative vectors of the original image size, store y.size in x and x.size i
    x = 0:1:size(za ,2) -1;
    y = 0:1:size(za ,1) -1;
    % ====> Create the position shift, when overflow occurs, return remainder <====
    xs = mod(x-Sx , size(za ,2));
    ys = mod(y-Sy , size(za ,1));
    % ====> Put the original values into the result with the shift applied <====
    zb = za(ys+1,xs+1);
end

%Part A
function [xs,ys,zs] = image_sample(z,D)
    xs = 1:(size(z,2)/D);
    ys = 1:(size(z,1)/D);
    xsample = 1:D:size(z,2);
    ysample = 1:D:size(z,1);
    zs = z(ysample, xsample)
```

```
end

%Part C
function zaa = image_antialias(z)
    zaa = zeros(size(z,1), size(z,2));
    for zy = 1:size(z,1)
        for zx = 1:size(z,2)
            if zx-1>1 && zy-1>0 && zx+1<=size(z,2) && zy+1<=size(z,1)
                zaa(zy,zx) = ((z(zy,zx))/2) + ((z(zy,zx-1)+z(zy,zx+1)+z(zy-1,zx)+z(zy+1,zx))/8)
            end
        end
    end
end
```

EXTRA CREDIT FUNCTIONS

```
function [xs,ys,zs] = rgbimage_sample(z,D)
    zs = zeros(ceil(size(z,1)/D),ceil(size(z,1)/D),size(z,3));
    xs = 1:ceil(size(z,2)/D);
    ys = 1:ceil(size(z,1)/D);
    xsample = 1:D:size(z,2);
    ysample = 1:D:size(z,1);
    for i = 1:size(z,3)
        zs(ys, xs,i) = z(ysample, xsample, i);
    end
end

function zaa = rgbimage_antialias(z)
    zaa = zeros(size(z,1), size(z,2));
    for zy = 1:size(z,1)
        for zx = 1:size(z,2)
            for zz = 1:size(z,3)
                if zx-1>1 && zy-1>0 && zx+1<=size(z,2) && zy+1<=size(z,1)
                    zaa(zy,zx,zz) = ((z(zy,zx,zz))/2) + ((z(zy,zx-1,zz)+z(zy,zx+1,zz)+z(zy-1,zx
                end
            end
        end
    end
end

function [xz,yz,zz] = rgbimage_insertzeros(zaas, U)
    yz = 1:U*size(zaas,1);
    xz = 1:U*size(zaas,2);
    x = ceil(U*size(zaas,1));
    y = ceil(U*size(zaas,2));
    z = size(zaas,3);
    zz = zeros(x, y, z);

    zz(1:U:end,1:U:end,1:end) = zaas(1:end, 1:end, 1:end);
end
```