ECE 253 Homework 4

Shivani Bhakta A13832428

Contents

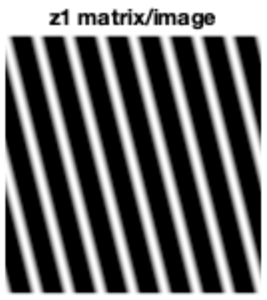
Problem 1 2D Sampling and Aliasing

Part(a)

```
clear; close all; clc;
[x y] = meshgrid(0:256,0:256);
figure, subplot(1,2,1), imshow(x, [0, 255])
subplot(1,2,2), imshow(y, [0, 255]), title('x, y ')
```

- We are given the sampling period T = 1, thus the sampling frequency F_s = 1.
- x and y are the spatial coordinates of the original image given by the mesh grid in matlab
- u and v are the spatial frequency coordinates of the Fourier transform of the image.

```
z1 = cos ( 2 * pi * 1/32 .* x - 2 * pi * 1/128 .* y);
figure, imshow(z1), title('z1 matrix/image');
```



Spatial Frequencies for z1.

- In x-direction = 1/32
- In y-direction = 1/128

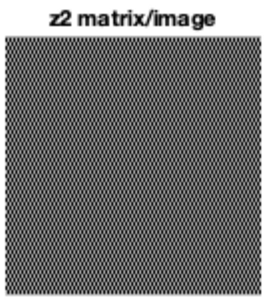
Image can be reconstructed exactly from it's samples if Cutoff freq ≤ 1/2 sampling freq F_s

- If equality holds, then the sampling is at the Nyquist rate
- If Δx and Δy are smaller than required, the image is called oversampled
- If they are larger than required, the image is undersampled

In terms of spatial frequency and the sampling frequency,

- 2 * 1/32 = 1/16 < F_s = 1 ==> oversampled in x-direction
- 2 * 1/128 = 1/64 < F_s = 1 ==> oversampled in y-direction

```
z2 = cos ( 2 * pi * 1/4 .* x - 2 * pi * 7/8 .* y);
figure, imshow(z2), truesize, title('z2 matrix/image');
```

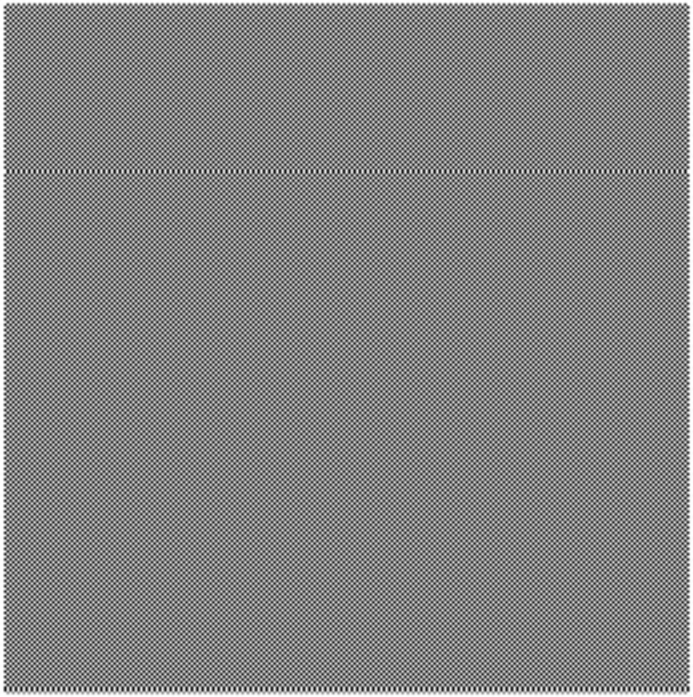


Spatial Frequencies for z2.

- In x-direction = 1/4
- In y-direction = 7/8

In terms of spatial frequency and the sampling frequency,

z3 matrix/image



- 2 * F_{xc} = 1/2 < F_s = 1 ==> oversamppled in x-direction
- 2 * F_{yc} = 7/4 > F_s = 1 ==> undersampled in y-direction

```
z3 = cos ( 2 * pi * 1/2 .* x - 2 * pi * 1/2 .* y);
figure, imshow(z3), title('z3 matrix/image');
```

Spatial Frequencies for z3.

- In x-direction = 1/2
- In y-direction = 1/2

In terms of spatial frequency and the sampling frequency,

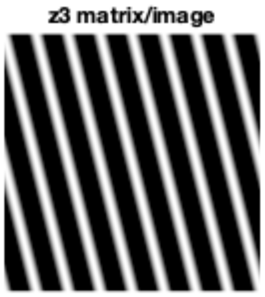
- 2 * F_{xc} = 1 = F_s = 1 ==> critically sampled in x-direction or sampling is at the Nyquist rate
- 2 * F_{yc} = 1 > F_s = 1 ==> critically sampled in y-direction or sampling is at the Nyquist rate

Part(b)

Since Cosine function is periodic with period 2π . we can use any frequencies which is generated by adding any multiple of 2 π to each of the xy frequency components in z1 to obtain a identical sampled function to that of the sampled function z1. Therefore let us consider a = F_{xc} + 2π and b = F_{yc} + 2π, taking the 2π in common we get a = 33/32 b = 129/128 Any a = 1/32 + k 2π where k $\in$ z, will give us the identical sampled function.

However, we have to note that the we want the sampled function to be aliased, this means we have to consider the inequality a>1/2 and b>1/2

```
z4 = cos ( 2 * pi * 33/32 .* x - 2 * pi * 129/128 .* y);
figure, imshow(z4), title('z3 matrix/image');
```



In the figure above it came be seen that z4 produces the same image as sampled in z1.

Published with MATLAB® R2022b

Contents

Problem 2 Pre-filtering to Reduce Aliasing before Sampling

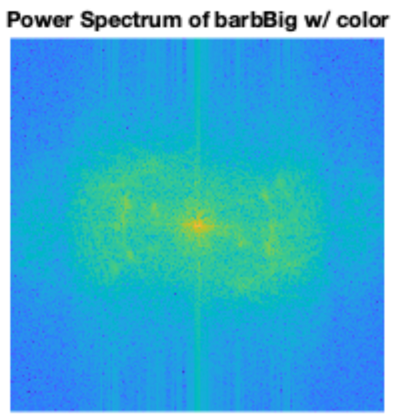
```
close all; clc; clear;
barbara = read(Tiff('barbara.tiff', 'r'));

figure,
subplot(1,2,1), imshow(barbara), title('barbara original');

baby = read(Tiff('baby.tiff', 'r'));
subplot(1,2,2), imshow(baby), title('baby original');
```

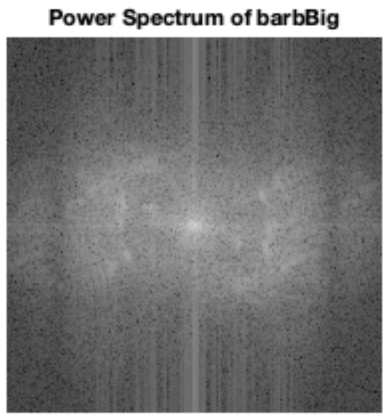
```
close all;
barbBig = barbara(1:512,37:548);
figure,
subplot(1,2,1), imshow(barbBig), title('barbBig 512 x 512 section');
babyBig = baby(201:712,201:712);
subplot(1,2,2), imshow(babyBig), title('babyBig 512 x 512 section');
```

barbBig 512 x 512 section

babyBig 512 x 512 section

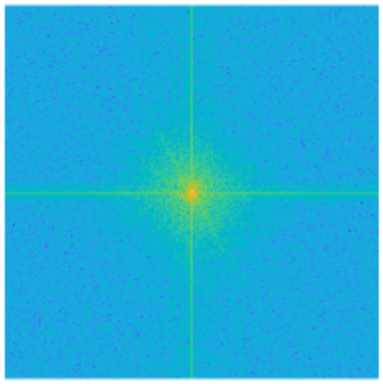
```
close all;
barbfft_1 = fftshift(abs(fft2(barbBig)).^2);
barbspectrum = log(barbfft_1 + 1);
figure,
subplot(1,2,1),
imshow(barbspectrum / max(max(barbspectrum))),
colormap("default"),
title('Power Spectrum of barbBig w/ color');

subplot(1,2,2),
imshow(barbspectrum / max(max(barbspectrum))),
title('Power Spectrum of barbBig');
```



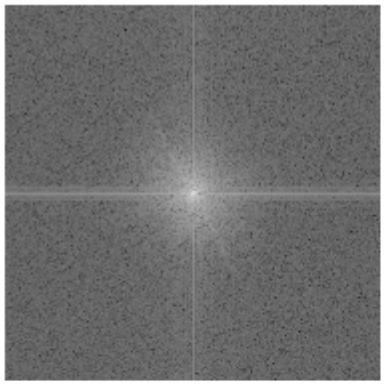


```
close all;
babyfft_1 = fftshift(abs(fft2(babyBig)).^2);
babyspectrum = log(babyfft_1 + 1);
figure, subplot(1,2,1),
imshow(babyspectrum / max(max(babyspectrum))),
colormap("default"),
title('Power Spectrum of babyBig w/ color');

subplot(1,2,2),
imshow(babyspectrum / max(max(babyspectrum))),
title('Power Spectrum of babyBig');
```

Power Spectrum of babyBig w/ color



Power Spectrum of babyBig



Part(a)

As can be seen from the above Power Spectrum the DC coefficient is located right at the center. Note that it would initially be in the corner but the fftshift brings it to the center. First fft transforms the signal to frequency domain, then the shift is basically rearanges the fourier transform by shifting the DC to center. Before the fftshift, the DC coefficient is located in the (1,1) position, and the low frequencies are out in the four corner areas. The fftshift swaps the quadrants, making the corner of each into the center sucuch that the quadrants are symetric at the center. The way we can check this is through the matlab documentation on how the fftshift function shifts the DC coefficent to be at the center of the 2D space, such that the spectrum looks symetric.

Part(b)

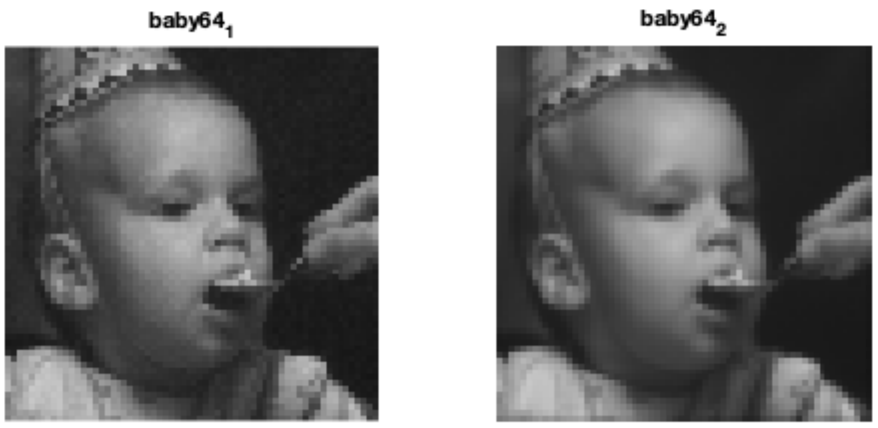
By just doing the visual comparision, we can say that barbbig has a higher frequency content because of the vertical and horizontal lines that we see in the image and power spectrum. Looking at the babyBig and it's power spectra, there are some high freq components, however not as high as barbbig. we just see the white circle in the middle which is a low frequency component. Barbara will have more of the aliasing problem from downsampling because of the high frequency components in the image.

Published with MATLAB® R2022b

Contents

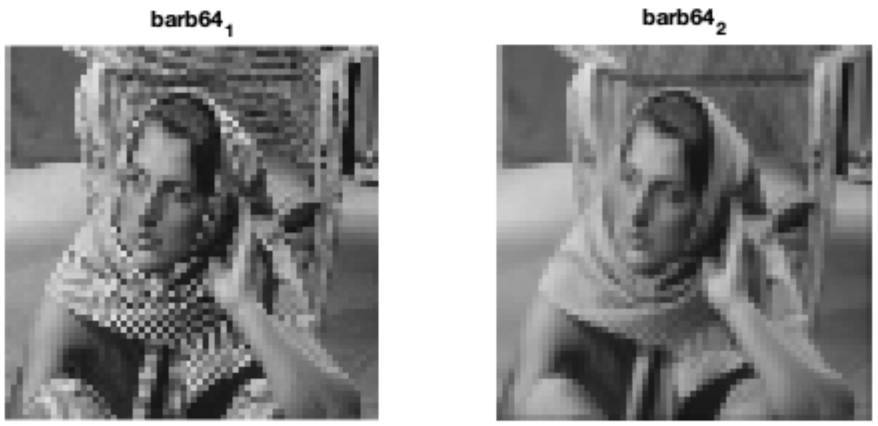
Part(c)

```
close all; clear; clc;
barbara = read(Tiff('barbara.tiff', 'r'));
baby = read(Tiff('baby.tiff', 'r'));

barbBig = barbara(1:512,37:548);
babyBig = baby(201:712,201:712);

baby64_1 = imresize(babyBig,0.125,'nearest');
figure,
subplot(1,2,1), imshow(baby64_1), title("baby64_1");

babylow = filter2(ones(8)/64,babyBig);
baby64_2 = imresize(babylow,0.125,'nearest');
subplot(1,2,2),
% imshow(baby64_2, [0 255]),
imshow(uint8(baby64_2)), title("baby64_2");
```

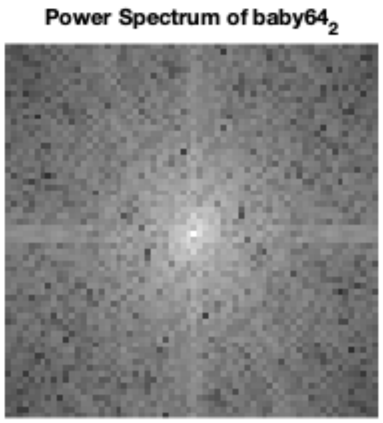


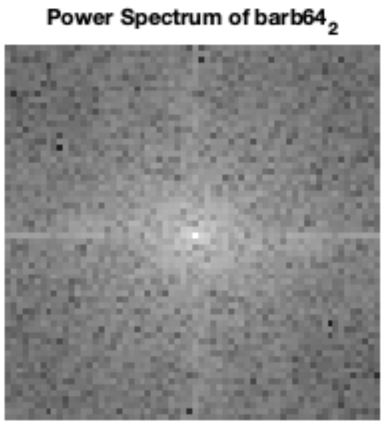
```
barb64_1 = imresize(barbBig,0.125,'nearest');
figure, subplot(1,2,1),
imshow(barb64_1), title("barb64_1")
```

```
barblow = filter2(ones(8)/64,barbBig);
barb64_2 = imresize(barblow,0.125,'nearest');
subplot(1,2,2),
imshow(barb64_2, [0 255]), title("barb64_2");
```



```
baby64fft_1 = fftshift(abs(fft2(baby64_2)).^2);
baby64spectrum = log(baby64fft_1 + 1);
figure, subplot(1,2,1),
imshow(baby64spectrum / max(max(baby64spectrum))),
title('Power Spectrum of baby64_2');


barb64fft_1 = fftshift(abs(fft2(barb64_2)).^2);
barb64spectrum = log(barb64fft_1 + 1);
subplot(1,2,2),
imshow(barb64spectrum / max(max(barb64spectrum)))
title('Power Spectrum of barb64_2');
```





By visually comparting the two 64x64 versions of baby and for barbara, looks like the pre-filtering helped barbara more because of the higher frequency components and aliasing as observed before. barb64_1 looks bad, like around the scarf area where we have vertical lines in the original image with a lot of alising because it has high freq which we supress using low freq filter and thus get a better image barb64_2 baby64_1 and baby64_2 looks identical, however, 1 is still little blury compared to 2. similarity between them is likely due to the low freq nature of the image. It didn't have aliasing which required the pre-filtering, and thus it didn't make any difference.

Part(d)

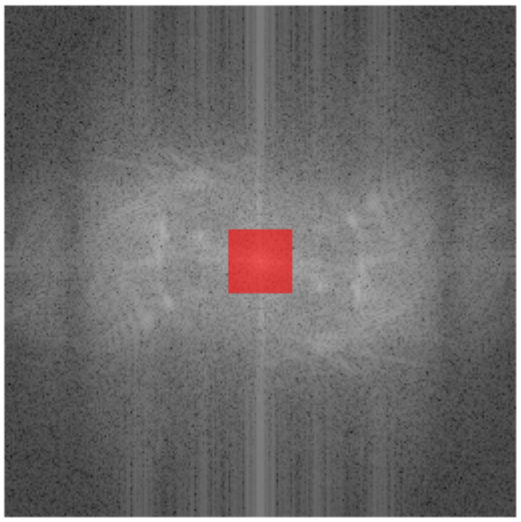
Cutoff frequency <= 1/2 * sampling frequency sampling frequency = 1/8 therefore, the cutoff freq is 1/16

```
barbBig = barbara(1:512,37:548);
babyBig = baby(201:712,201:712);

barbfft_1 = fftshift(abs(fft2(barbBig)).^2);
barbspectrum = log(barbfft_1 + 1);


barbfft_1 = fftshift(abs(fft2(barbBig)).^2);
barbspectrum = log(barbfft_1 + 1);
figure,
I = barbspectrum / max(max(barbspectrum));
rec = insertShape(I,"filled-rectangle",[225 225 64 64], 'Color' , 'red');
imshow(rec)
title('Power Spectrum of barbBig');

% not Aliased Energy
ERM = sum(barbfft_1(225:288,225:288),'all');

% Aliasing Energy
```

```
EAM = sum(barbfft_1,'all') - ERM;

fprintf('The not Aliased Energy is %i.\n', ERM);
fprintf('The Aliasing Energy %i.\n', EAM);
```

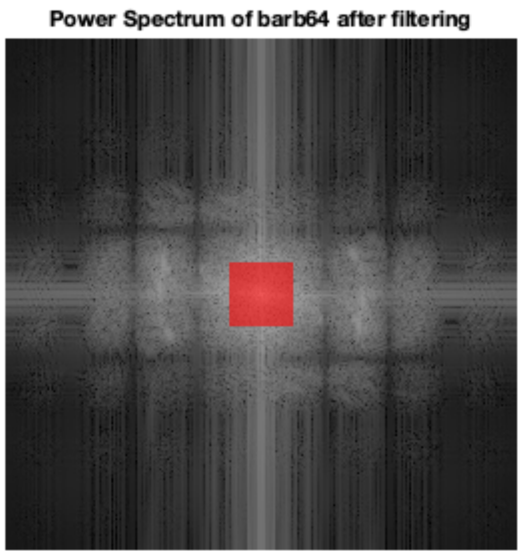
```
The not Aliased Energy is 1.022746e+15.
The Aliasing Energy 2.233135e+13.
```

Power Spectrum of barbBig



```
% filtered!
barbfft_2 = fftshift(abs(fft2(barblow)).^2);
barbspectrum = log(barbfft_2 + 1);
figure,
I_64_2 = barbspectrum / max(max(barbspectrum));
rec = insertShape(I_64_2,"filled-rectangle",[225 225 64 64], 'Color' , 'red');
imshow(rec)
title('Power Spectrum of barb64 after filtering');

% not Aliased Energy
ER = sum(barbfft_2(225:288,225:288),'all');

% Aliasing Energy
EA = sum(barbfft_2,'all') - ER;

% Aliasing Reduction
aliasing_reduction = (EAM-EA)/EAM;

% Resolution Error
resolution_error = (ERM-ER)/ERM;


fprintf('The not Aliased Energy after filter is %i.\n', ER);
fprintf('The Aliasing Energy after filter %i.\n', EA);
fprintf('Aliasing Reduction %i.\n', aliasing_reduction);
fprintf('Resolution Error %i.\n', resolution_error);
```

```
The not Aliased Energy after filter is 1.000850e+15.
The Aliasing Energy after filter 2.317592e+12.
Aliasing Reduction 8.962180e-01.
Resolution Error 2.140916e-02.
```

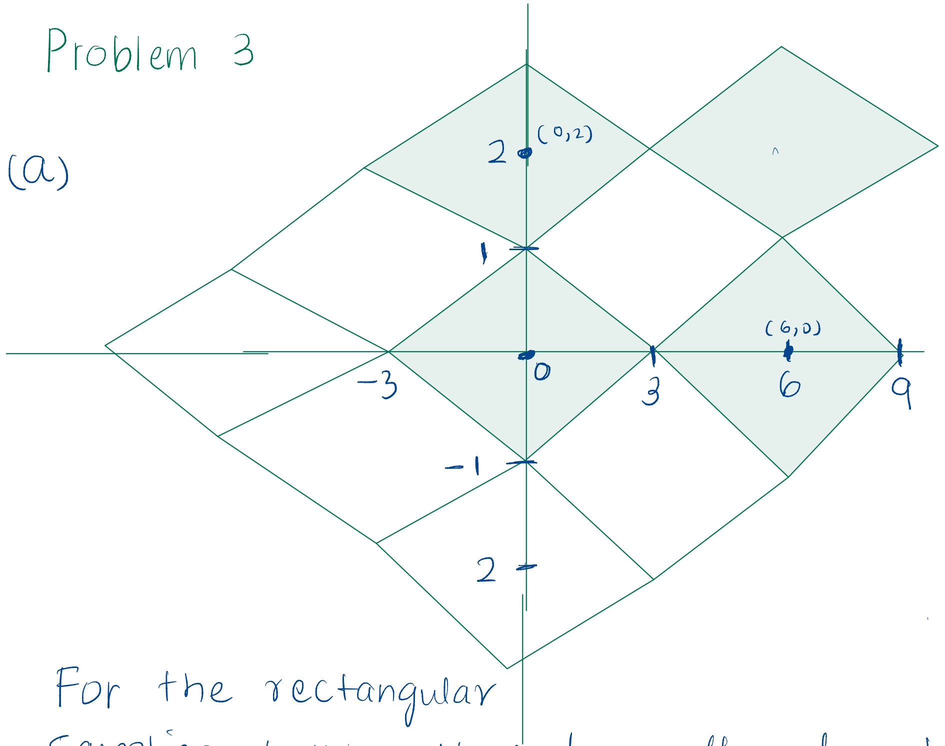


% In the above image of the power spectra, the image where it is aliased and not-aliased is shown. The region inside the inner box with a red boarder is where the spectrum showes part which is not aliased. The region outside isis aliased.

Published with MATLAB® R2022b

Problem 3

(a)



For the rectangular sampling lattice that has the lowest sampling density which can still ensure no aliasing (no overlap of spectral replications) has a generator matrix :

$$B = \begin{bmatrix} 6 & 0 \\ 0 & 2 \end{bmatrix} \qquad B = B^T$$

$\uparrow$ each column is a spectral replication location

Therefore,

$$B = (A^{-1})^T$$

$$B^T = A^{-1}$$

$$(B^T)^{-1} = A$$

$$A = \frac{1}{12}\begin{bmatrix} 2 & 0 \\ 0 & 6 \end{bmatrix}$$

$$= \begin{bmatrix} 1/6 & 0 \\ 0 & 1/2 \end{bmatrix} \leftarrow \text{Generator matrix for lattice}$$

$$\det(A) = 1/12 \leftarrow$$

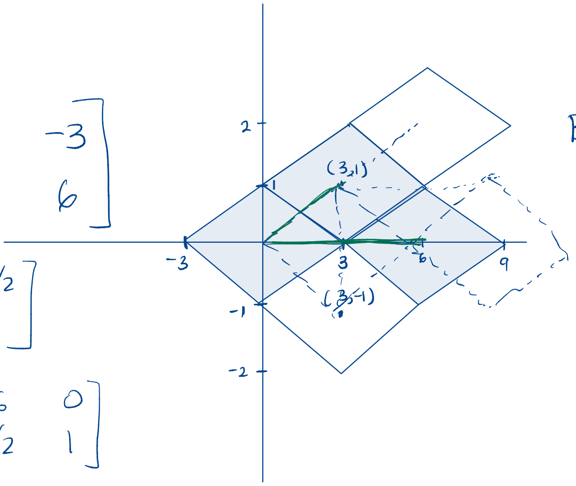
(b) Generator Matrix for non-rectangular sampling lattice w/ lowest sampling density & prevents spectral replications from overlapping.

$$B^{-1} = \frac{1}{6-0}\begin{bmatrix} 1 & -3 \\ 0 & 6 \end{bmatrix}$$

$$B^{-1} = \begin{bmatrix} 1/6 & -1/2 \\ 0 & 1 \end{bmatrix}$$

$$(B^{-1})^T = A = \begin{bmatrix} 1/6 & 0 \\ -1/2 & 1 \end{bmatrix}$$

$$\det(A) = \frac{1}{6}$$



$$B = \begin{bmatrix} 6 & 3 \\ 0 & 1 \end{bmatrix}$$

Hexagonal Lattice

c) Percentage reduction in samples by sampling on the non-rectangular ~~grid~~ grid as opposed to the rectangular grid.

$$\frac{\det(A)_{Rec}}{\det(A)_{NRec}} = \frac{1/12}{1/6} = \frac{1}{2} = 0.50.$$