# 6. Image Processing - Filtering in Frequency Domain

Institute for Biomedical Imaging, Hamburg University of Technology

- Lecture: Prof. Dr.-Ing. Tobias Knopp
- Exercise: Konrad Scheffler, M.Sc.

## Table of Contents



## 6.1 Fourier Transformation

The Fourier transform plays a very important role in signal and image processing. In particular it allows to:

- analyse image characteristics in a different (often better) way than in image space
- understand sampling artifacts much better
- separate noise signal from object signal in a more natural way
- apply filters more efficiently

And on top of that the Fourier transformation is a lot of fun!

> **Note**
>
> In this lecture we try to capture the most important things about the Fourier transformation and its application. If you have not learned about the Fourier transformation we encourage that you get a book on basic signal processing. Basic knowledge on Fourier series that you learned about in an Analysis lecture is assumed.

> **Note**
>
> We will introduce most mathematics in 1D since the extension to multiple dimensions is straight forward. Considering it in 1D allows for a more concise presentation.

### 6.1.1 Fourier Series

We start with a basic example. Suppose that you have a 1-period function $\mathbf{rect} : [0,1) \to \mathbb{R}$ with

$$\mathbf{rect}(x) = \begin{cases} 1 & \text{if} \quad \frac{1}{4} < x < \frac{3}{4} \\ 0 & \text{if} \quad x = \frac{1}{4} \vee x = \frac{3}{4} \\ -1 & \text{otherwise} \end{cases}$$

and you want to express this function using a sum of sine functions. Then, one can show that the $\mathbf{rect}$ function can be expressed as

$$\begin{aligned} \mathbf{rect}(x) &= \frac{4}{\pi}\left[\sin(2\pi x) + \frac{1}{3}\sin(6\pi x) + \frac{1}{5}\sin(10\pi x) + \ldots\right] \\ &= \frac{4}{\pi}\sum_{k=1}^{\infty}\frac{\sin((2k-1)2\pi x)}{2k-1} \end{aligned}$$

The shown sum is the so-called Fourier series and for a general $P$-period function it can be expressed as

$$f(x) = \sum_{n=-\infty}^{\infty} c_n \, e^{i\frac{2\pi}{P}nx}.$$

The Fourier coefficients $c_n$ can be calculated by

$$c_n = \frac{1}{P}\int_0^P f(x)\, e^{-i\frac{2\pi}{P}nx} \mathrm{d}x.$$

Here, it does not matter over which part of the function is integrated as long as it is a full period.

**Note**

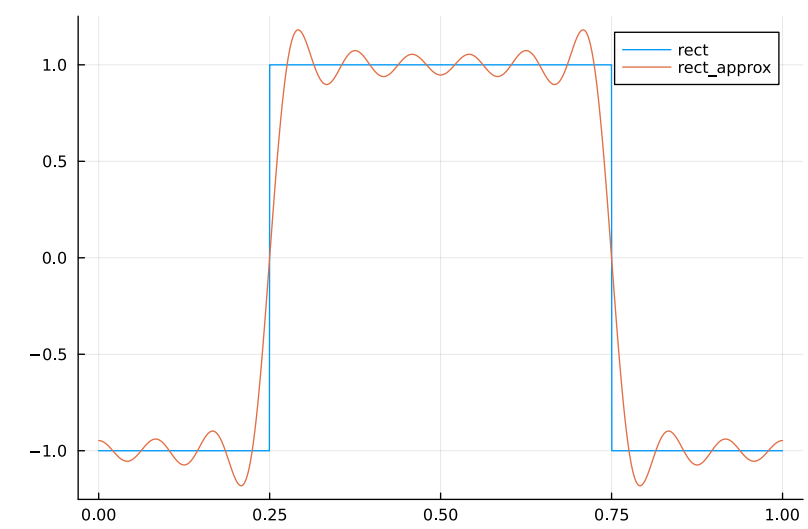
Not every P-periodic function has a Fourier series. A sufficient criterion is the fulfillment of the Dirichlet_conditions that require:
➔ $f$ must be absolutely integrable over a period.
➔ $f$ must be of bounded variation in any given bounded interval.
➔ $f$ must have a finite number of discontinuities in any given bounded interval, and the discontinuities cannot be infinite.

## 6.1.2 Example

The following shows the Fourier approximation of the rectangular function taking only $n < N$ summands into account. With this slider

N = 11

you can control the number of summands being included:




```
let
    M = 2000
    t = range(0,1,length=M)

    rect(t) = t < 0.25 ? -1.0 : (t > 0.75 ? -1.0 : 1.0)

    rect_approx(t,N) = sum([ 4/(π*n)*sin(2*pi*(t-0.25)*n) for n=1:2:N])

    p = plot(t,rect.(t),label="rect")
    plot!(p,t,rect_approx.(t,N),label="rect_approx")
end
```

### Observations

1. The more terms we use, the better is the approximation.
2. A Fourier approximation often show a *ringing* effect. This ringing is more pronounced at dis discontinuities (i.e. edges) and called Gibbs ringing. We will investigate this later.

**Question**

Why is the Fourier series still exact although all(!) finite approximations show an almost constant *jump height* of the ringing?

Hint: Think about what happens in the limit $N \to \infty$.

# 6.2 Continuous Fourier Transform

The Fourier series is restricted to periodic functions, which is clear since the base functions are periodic. But it can be generalized to non-periodic functions resulting the the continuous Fourier transform that is defined as

$$(\mathcal{F}f)(\xi) = \hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)\, \mathrm{e}^{-2\pi i x\xi} \mathrm{d}x.$$

It also has an associated inverse that can be calculated by

$$(\mathcal{F}^{-1}\hat{f})(x) = f(x) = \int_{-\infty}^{\infty} \hat{f}(\xi)\, \mathrm{e}^{2\pi i x\xi} \mathrm{d}\xi$$

One can derive this by considering the limit $P \to \infty$ in the definition of the Fourier series, i.e. one is basically stretching one period over the entire real line.

The Fourier transformation and its inverse only exists under certain circumstances. A sufficient criterion for this is that $f$ and its inverse are both functions from the space of integrable functions $L^1(\mathbb{R})$, i.e. $f$ and its inverse fulfill

$$\|f(x)\|_1 := \int_{-\infty}^{\infty} |f(x)| \mathrm{d}x < \infty$$

This criterion directly derives from the definition of the Fourier transform when considering $\xi = 0$ (or $x = 0$ for the inverse), which basically integrates the function over the entire real line.

**Note**

The existence of Fourier integrals goes far beyond the scope of the present lecture and requires proper knowledge of functional analysis. But note that there are cases where the sufficient criterion is not fulfilled but a Fourier transform can still be applied.

The functions $f$ and $\hat{f}$ are often denoted as a Fourier transformation pair which is graphically shown as

$$f(x) \overset{\mathcal{F}}{\longleftrightarrow} \hat{f}(\xi).$$

Here is a table of some important Fourier relations:

| $f(x)$ | $\hat{f}(\xi)$ |
|---|---|
| $1$ | $\delta(\xi) = \lim_{p\to\infty} p\, \mathrm{rect}(\frac{\xi}{p})$ |
| $\mathrm{rect}(x)$ | $\mathrm{sinc}(\xi) = \frac{\sin(\pi\xi)}{\xi}$ |
| $\mathrm{e}^{2\pi i\xi_0 x}$ | $\delta(\xi - \xi_0)$ |
| $\cos(2\pi\xi_0 x)$ | $\frac{1}{2}(\delta(\xi+\xi_0) + \delta(\xi-\xi_0))$ |
| $\sin(2\pi\xi_0 x)$ | $\frac{i}{2}(\delta(\xi+\xi_0) - \delta(\xi-\xi_0))$ |
| $\mathrm{e}^{-ax^2}$ | $\sqrt{\frac{\pi}{a}}\mathrm{e}^{-\frac{\pi^2\xi^2}{a}}$ |
| $\frac{\mathrm{d}^d}{\mathrm{d}x^d} f(x)$ | $(2\pi\xi_0)^d \hat{f}(\xi)$ |
| $\sum_{n=-\infty}^{\infty} \delta(x - nX)$ | $\frac{1}{X}\sum_{n=-\infty}^{\infty} \delta(\xi - \frac{n}{X})$ |
| $f(x - x_0)$ | $\mathrm{e}^{-i2\pi x_0\xi}\hat{f}(\xi)$ |
| $f(ax)$ | $\frac{1}{\lvert a\rvert}\hat{f}(\frac{\xi}{a})$ |

**Note**

We encourage you to prove some of these transformation pairs on your own. It gives you a much better understanding of the Fourier transform. You will need some trigonometric identities for some proves.

## 6.2.1 Convolution Theorem

One of the most important Fourier relationships is the convolution theorem:

**Theorem**

Let $g, s, h \in L^1(\mathbb{R})$ be functions that satisfy $\hat{g}, \hat{s}, \hat{h} \in L^1(\mathbb{R})$. Then, the convolution $g(x) = (s * h)(x)$ in spatial space corresponds to a multiplication in Fourier space:

$$\hat{g}(\xi) = \hat{s}(\xi)\hat{h}(\xi).$$

**Proof**

$$\begin{aligned}
\hat{g}(\xi) &= (\mathcal{F}g)(\xi) = (\mathcal{F}(s*h))(\xi) \\
&= \int_{-\infty}^{\infty} \left( \int_{-\infty}^{\infty} s(\tilde{x}) h(x-\tilde{x}) \mathrm{d}\tilde{x} \right) \mathrm{e}^{-2\pi i x\xi} \mathrm{d}x \\
&= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} s(\tilde{x}) h(x-\tilde{x}) \mathrm{e}^{-2\pi i(x-\tilde{x})\xi} \mathrm{e}^{-2\pi i\tilde{x}\xi} \mathrm{d}\tilde{x}\mathrm{d}x \\
&= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} s(\tilde{x}) h(z) \mathrm{e}^{-2\pi i\tilde{x}\xi} \mathrm{e}^{-2\pi i z\xi} \mathrm{d}\tilde{x}\mathrm{d}z \\
&= \int_{-\infty}^{\infty} s(\tilde{x}) \mathrm{e}^{-2\pi i\tilde{x}\xi} \mathrm{d}\tilde{x} \int_{-\infty}^{\infty} h(z) \mathrm{e}^{-2\pi i z\xi} \mathrm{d}z \\
&= \hat{s}(\xi)\hat{h}(\xi)
\end{aligned}$$

What this basically tells us is that we can apply a convolution in Fourier space much easier by just a multiplication. This multiplication is what we originally understand by *filtering*.

## 6.2.2 Filtering

The convolution theorem gives us a general pattern how we can apply convolution/filter operation in Fourier space. It consists of the following three steps to compute $(s * h)(x)$:

1. Calculate the Fourier transforms $\hat{s}(\xi)$ and $\hat{h}(\xi)$
2. Apply the filter by calculating $\hat{d}(\xi) = \hat{s}(\xi)\hat{h}(\xi)$
3. Apply the inverse Fourier transform to $\hat{d}(\xi)$ yielding $d(x) = (s * h)(x)$

**Note**

While this general pattern also holds in the discrete setting you will need to account for some additional things like fftshift and padding.

### 6.2.3 Multi-Dimensional Fourier Transform

The multidimensional Fourier transform is defined as

$$(\mathcal{F}f)(\mathbf{k}) = \hat{f}(\mathbf{k}) = \int_{-\infty}^{\infty} f(\mathbf{r})\, e^{-2\pi i \mathbf{r}^T \mathbf{k}} d\mathbf{r}.$$

It also has an associated inverse that can be calculated by

$$(\mathcal{F}^{-1}\hat{f})(\mathbf{r}) = f(\mathbf{r}) = \int_{-\infty}^{\infty} \hat{f}(\mathbf{k})\, e^{2\pi i \mathbf{r}^T \mathbf{k}} d\mathbf{k}.$$

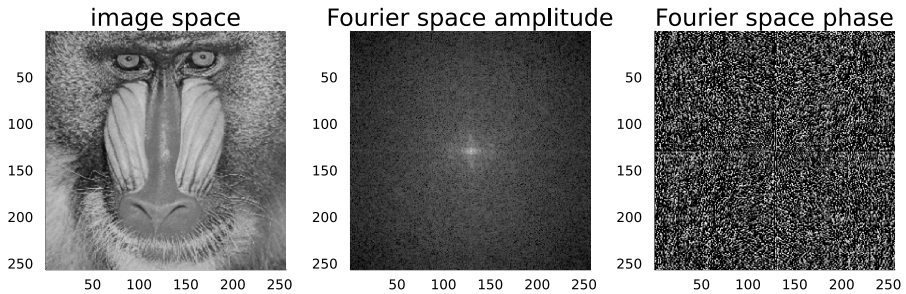
So the difference is that we have the inner product $\mathbf{r}^T\mathbf{k}$ in the exponent and that the integration is multidimensional. Since

$$e^{a+b} = e^a e^b$$

one can see that the term $\mathbf{r}^T\mathbf{k}$ naturally arises from a tensor product approach of the exponential base functions.

## 6.3 Fourier Transform on Images

Before we discuss the mathematics further, we want to look at some actual data:

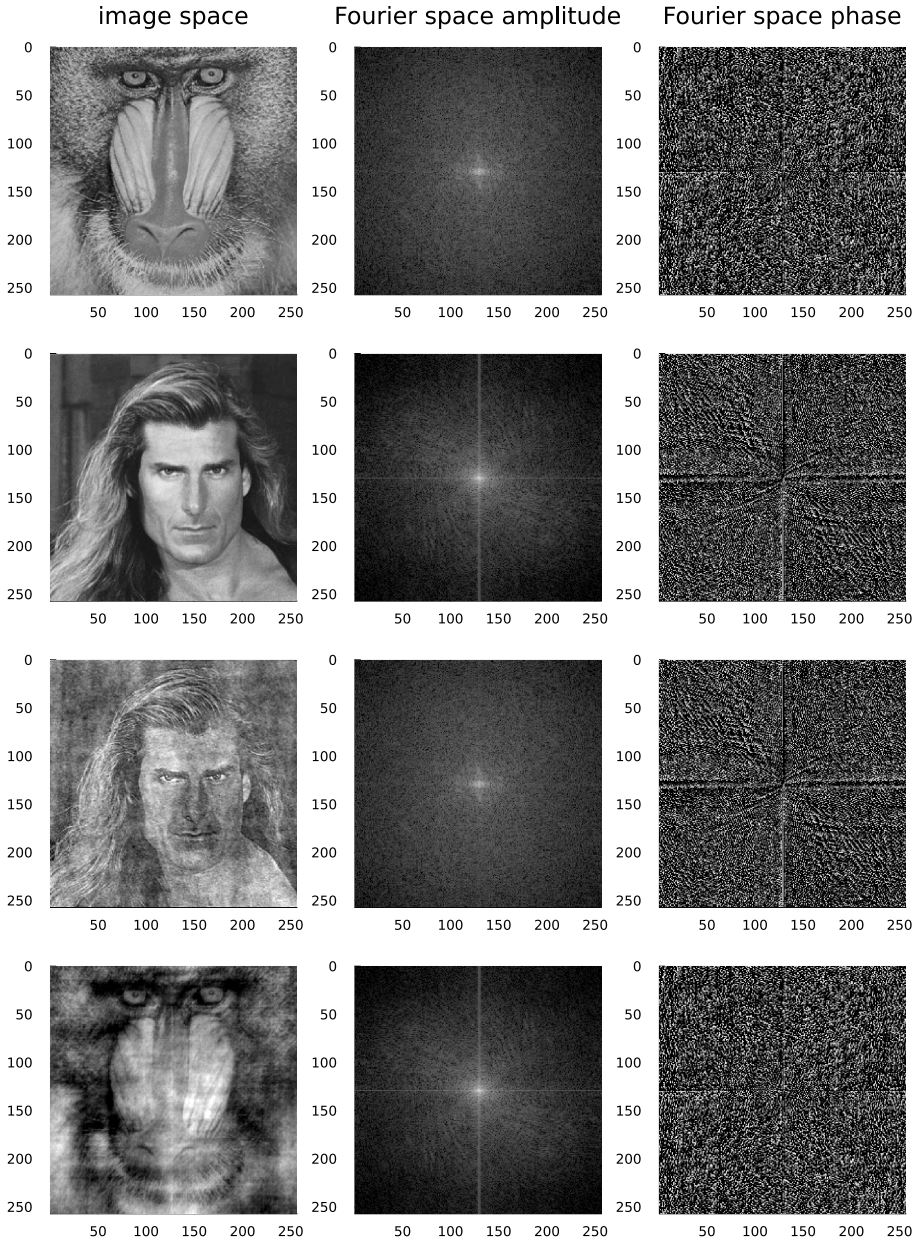



#### Observations

- Fourier space has a higher signal in the center
- It needs to be shown in a logarithmic fashion since otherwise only the center would be visible.

### 6.3.1 Amplitude vs Phase

It is an interesting question whether the main image information is encoded in the phase or in the amplitude. This can be tested as follows:

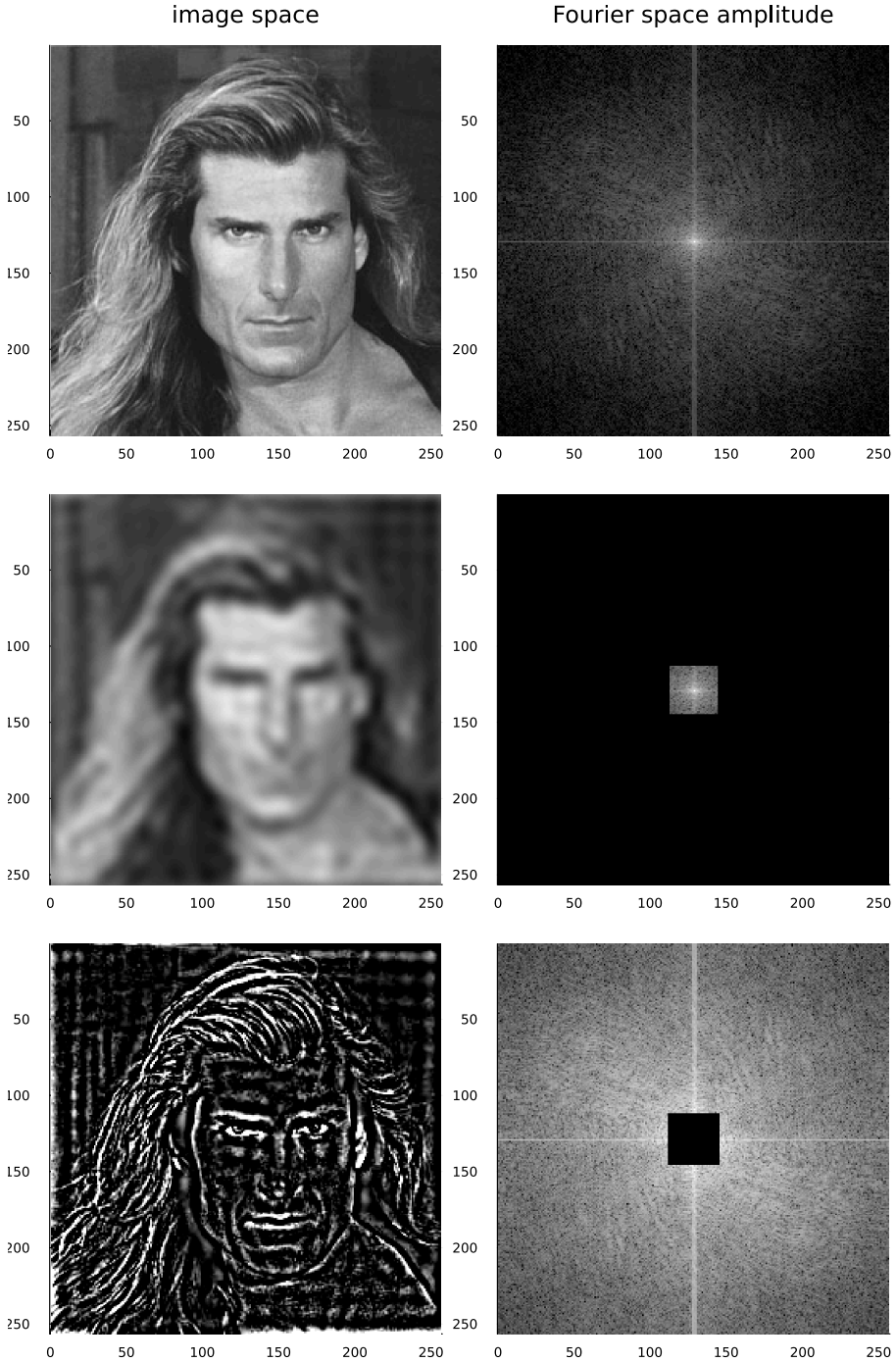



Thus, phase is actually very important and cannot be simply left away. This is actually not too much of a surprise because in the phase the actual positioning of pixels is encoded. If you take a Dirac delta in image space and shift it around this will change only the phase in Frequency space:

$$\delta(x - x_0) \overset{\mathcal{F}}{\longleftrightarrow} e^{-i2\pi x_0 \xi}.$$

## 6.3.2 Encoding Properties

We next want to have a look what different parts of the Fourier space actually encode. To this end we set certain parts of frequency space to zero and look at the effect on the resulting image in image space.




### Observations

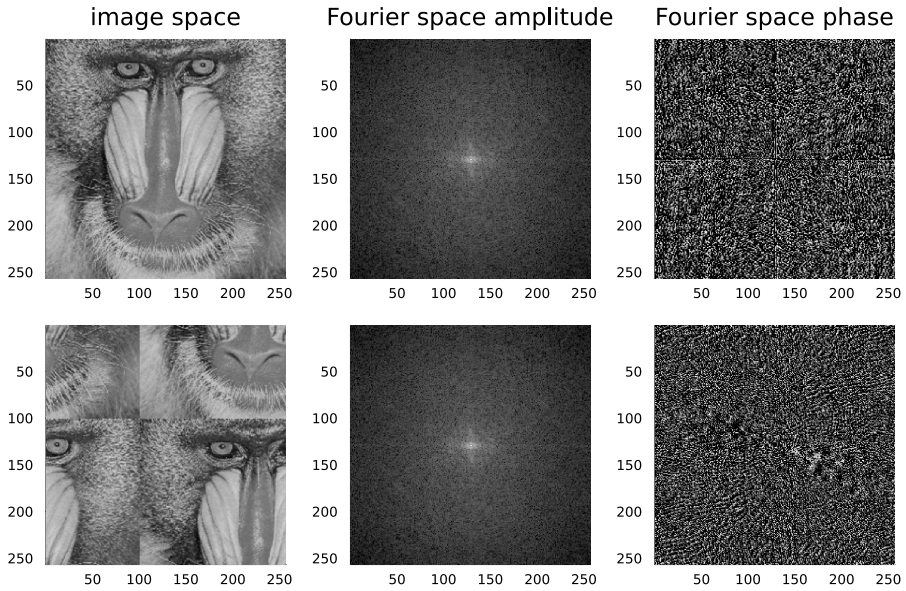
- Preserving the central part of the frequency space (low frequency part) leads to a blurred image.
- Preserving the outer part of the frequency space (high frequency part) leads to an edge image.

### 6.3.3 Geometric Transformations

Let us have a look what happens if we apply geometric transformations. If we shift our image in image space, the Fourier transform will be weighted with a phase:

$$f(x + x_0, y + y_0) \overset{\mathcal{F}}{\longleftrightarrow} \mathrm{e}^{2\pi i(x_0 u + y_0 v)} \hat{f}(u, v).$$

In images this looks like this:

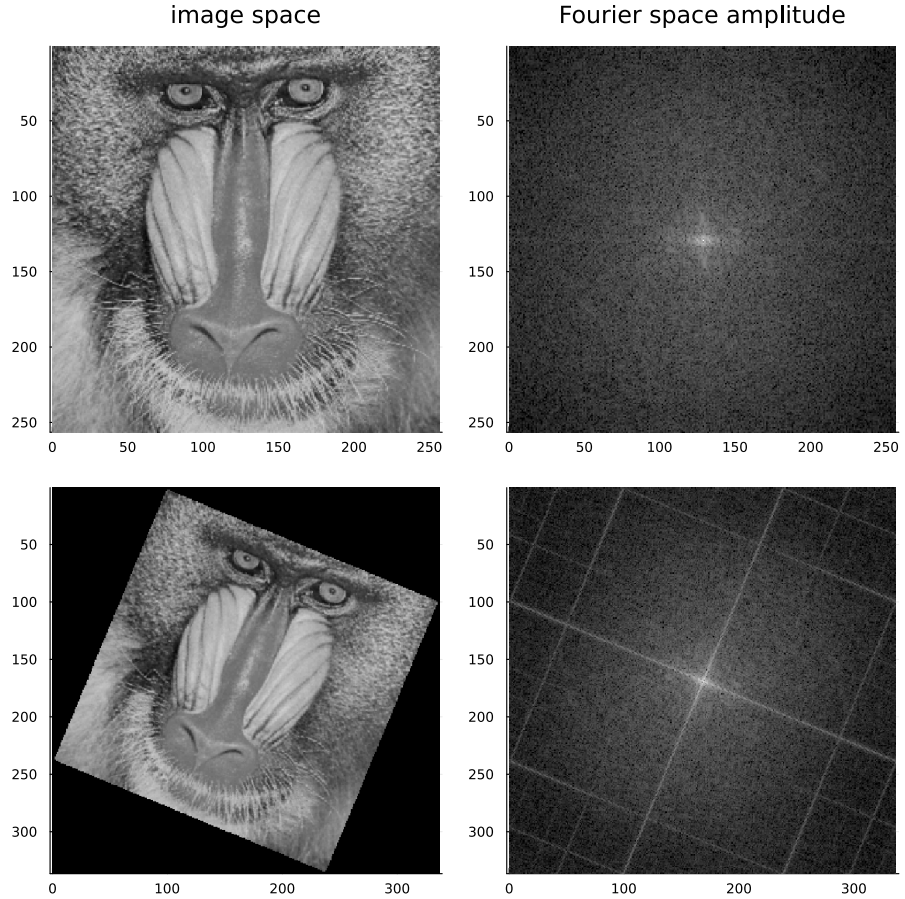



For an image rotation with the rotation matrix $\mathbf{R}_\alpha$ and its inverse $\mathbf{R}_\alpha^{-1} = \mathbf{R}_\alpha^T$ we can calculate the Fourier transform of $f(\mathbf{R}_\alpha \mathbf{r})$:

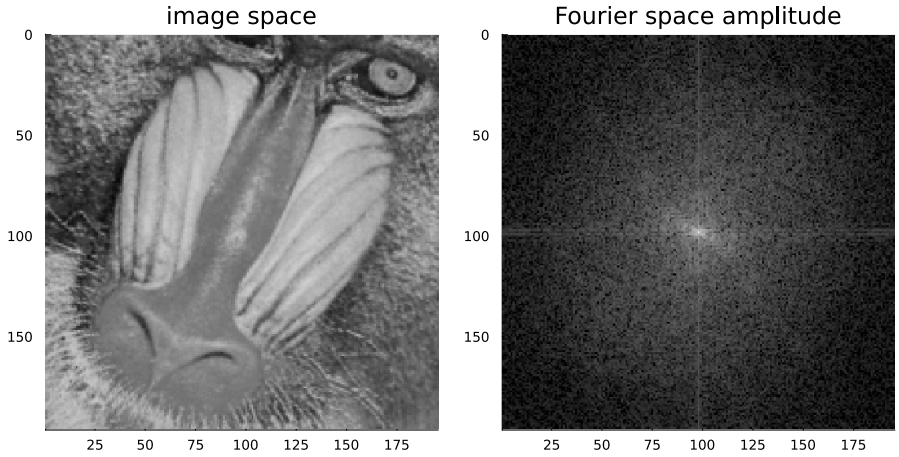
$$\begin{aligned}
(\mathcal{F} f(\mathbf{R}_\alpha \mathbf{r}))(\mathbf{k}) &= \int_{\mathbb{R}^2} f(\mathbf{R}_\alpha \mathbf{r})\, \mathrm{e}^{2\pi i \mathbf{k}^T \mathbf{r}} \mathrm{d}\mathbf{r} \quad \left| \text{subst. } \tilde{\mathbf{r}} = \mathbf{R}_\alpha \mathbf{r} \right. \\
&= \int_{\mathbb{R}^2} f(\tilde{\mathbf{r}})\, \mathrm{e}^{2\pi i \mathbf{k}^T \mathbf{R}_\alpha^T \tilde{\mathbf{r}}} \mathrm{d}\tilde{\mathbf{r}} \\
&= \int_{\mathbb{R}^2} f(\tilde{\mathbf{r}})\, \mathrm{e}^{2\pi i (\mathbf{R}_\alpha \mathbf{k})^T \tilde{\mathbf{r}}} \mathrm{d}\tilde{\mathbf{r}} \\
&= (\mathcal{F} f(\mathbf{r}))(\mathbf{R}_\alpha \mathbf{k})
\end{aligned}$$

We have used that the determinant of the Jacobian during the substitution is equal to 1.

Consequently, a rotation in image space corresponds to a rotation in the Fourier space. Lets see, if we can verify this on images:




On a first sight the spectrum is indeed rotated. On a second sight one can see several additional lines appearing. The simple reason for this is that the rotation invariance only holds in the continuous setting. The edges appear due to the sharp edges present in the rotated image shown on the left. These can be removed by cropping the rotated image but the fact remains that the rotation invariance does not hold in the discrete setting:

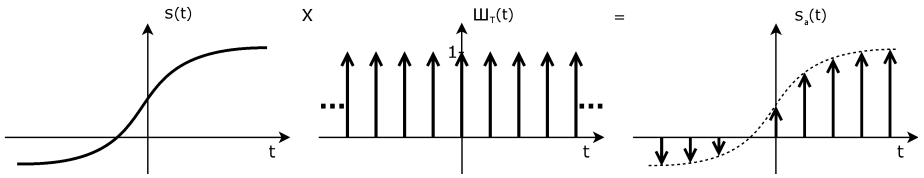



## 6.4 Discrete Fourier Transform

We have until now considered the continuous Fourier transformation on the real line. In practice, images are discrete and the question is what effect sampling has on the signal and its Fourier transform. Sampling can be expressed as a multiplication of a the signal $s(x)$ with the Dirac comb, i.e.

$$s_{\text{sampled}}(x) = s(x) \sum_{n=-\infty}^{\infty} \delta(x - nX)$$

This is illustrated in the following picture:




**Note**

The Dirac distribution is here basically only a handy concept. One could also express the discretized function as a step function. The difference is just that the sampled value is either smeared across the entire pixel, or within a single point.

If we look closely at the sampling relation we see that we have a multiplication in the spatial domain and thus we can express the sampling in Fourier space as convolution:

$$\begin{aligned}
\hat{s}_{\text{sampled}}(\xi) &= \hat{s}(\xi) * \frac{1}{X} \sum_{n=-\infty}^{\infty} \delta(\xi - \frac{n}{X}) \\
&= \int_{-\infty}^{\infty} \hat{s}(\tilde{\xi}) \left( \frac{1}{X} \sum_{n=-\infty}^{\infty} \delta(\xi - \tilde{\xi} - \frac{n}{X}) \right) \mathrm{d}\tilde{\xi} \quad | \text{subst. } \tilde{\xi} = \xi' + \frac{n}{X} \\
&= \int_{-\infty}^{\infty} \hat{s}(\xi' + \frac{n}{X}) \left( \frac{1}{X} \sum_{n=-\infty}^{\infty} \delta(\xi - \xi') \right) \mathrm{d}\xi' \\
&= \frac{1}{X} \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} \hat{s}(\xi' + \frac{n}{X}) \delta(\xi - \xi') \mathrm{d}\xi' \\
&= \frac{1}{X} \sum_{n=-\infty}^{\infty} \hat{s}(\xi + \frac{n}{X})
\end{aligned}$$

This sum is called the *periodization* of $\hat{s}(\xi)$. In fact $\hat{s}_{\text{sampled}}(\xi)$ is $1/X$-periodic

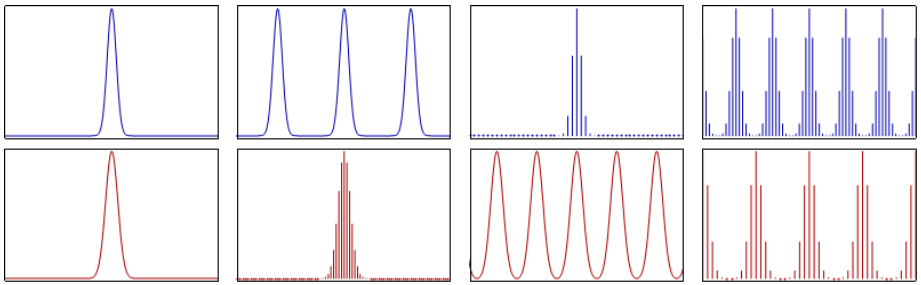
$$\begin{aligned}
\hat{s}_{\text{sampled}}(\xi + \frac{\alpha}{X}) &= \frac{1}{X} \sum_{n=-\infty}^{\infty} \hat{s}(\xi + \frac{\alpha}{X} + \frac{n}{X}) \\
&= \frac{1}{X} \sum_{n=-\infty}^{\infty} \hat{s}(\xi + \frac{\alpha + n}{X}) \quad | \text{subst. } \tilde{n} = n + \alpha \\
&= \frac{1}{X} \sum_{\tilde{n}=-\infty}^{\infty} \hat{s}(\xi + \frac{\tilde{n}}{X}) \\
&= \hat{s}_{\text{sampled}}(\xi)
\end{aligned}$$

**Note**

This implies: A periodization in spatial (Fourier) domain leads to a periodization in Fourier (spatial) domain.

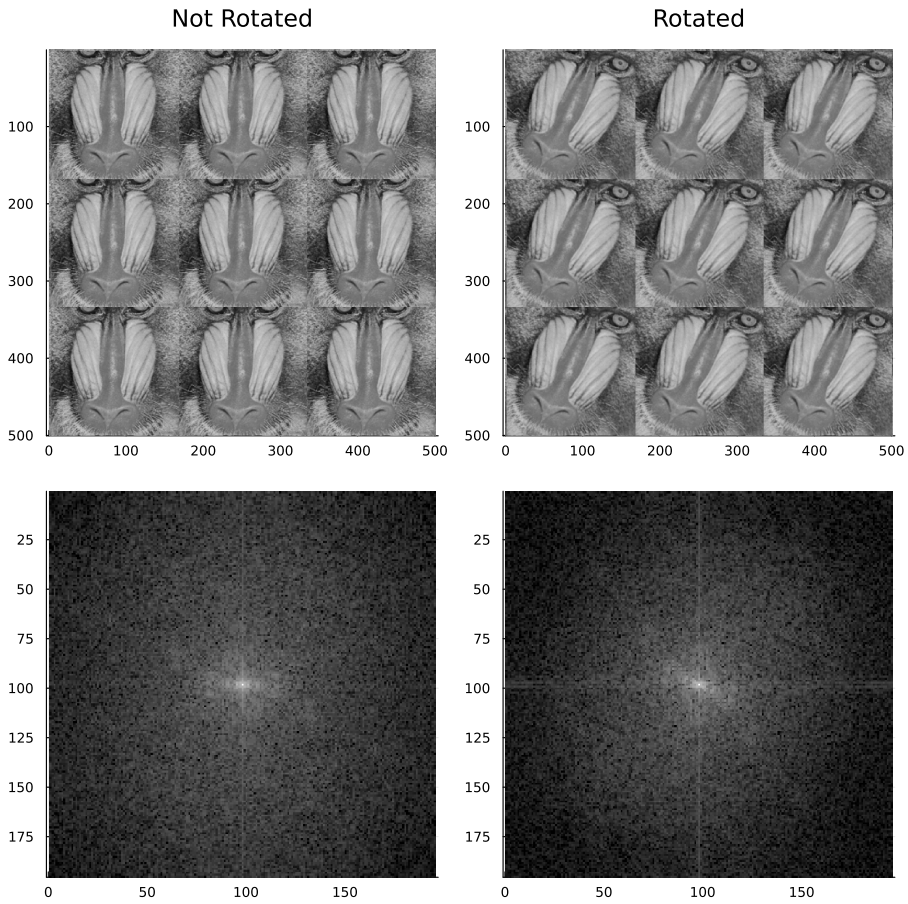
We have, by the way, also seen this between when going from the Fourier series to the continuous Fourier transform.

If we now sample the periodic domain as well we end up at a periodic and discrete signal in both domains leading to the Discrete Fourier Transform (DFT). The following picture summarizes the four different combinations of Fourier transforms:



### 6.4.1 Example

Let us shortly revise the rotation of image, which we proved to be invariant in Fourier space. When looking at periodizations of the images, they look like this:




Clearly the rotation invariance cannot be fulfilled in the discrete case if the image goes not to zero at the image boundaries. Also this image explains why the cross in Fourier space is not rotated as well. The cross is caused by the edges.

### 6.4.2 Definition

The DFT is defined as

$$\hat{s}(\xi) = \sum_{n=0}^{N-1} s(x) \mathrm{e}^{-2\pi i \frac{x\xi}{N}}.$$

It has an associated inverse transform that is defined as

$$s(x) = \frac{1}{N} \sum_{n=0}^{N-1} \hat{s}(\xi) \mathrm{e}^{2\pi i \frac{x\xi}{N}}$$

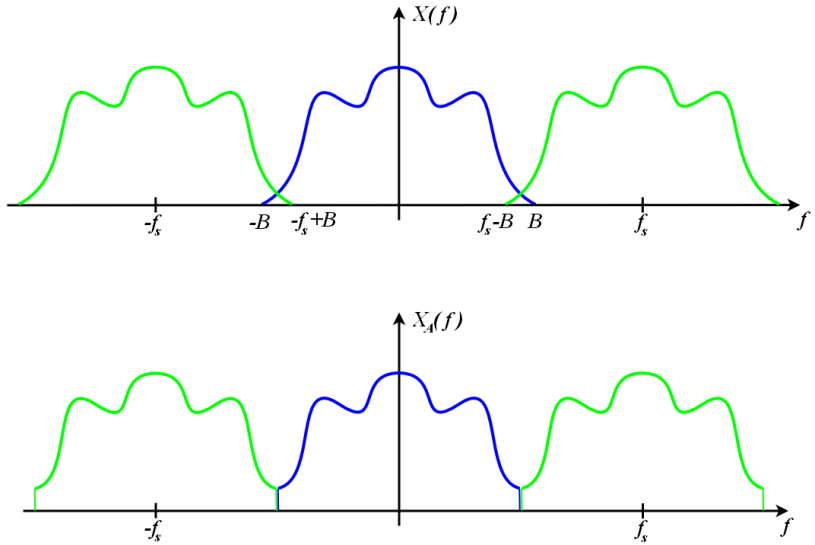
The DFT is a linear transform that can be written in matrix-vector notation $\mathbf{Fs} = \hat{\mathbf{s}}$. The matrix fulfills $\mathbf{F}^H\mathbf{F} = N\mathbf{I}$, i.e. with an appropriate scaling factor the Fourier matrix is unitary.

## 6.5 Sampling Artifacts

It is important to understand that sampling leads to errors. This means that the DFT is not automatically a good approximation of the Fourier series or the continuous Fourier transform. There are two common errors occurring.

### 6.5.1 Aliasing Error

Aliasing error happen when function that has no limited support in Fourier space is sampled. Due to the periodization it can happen that the signals fold into each other, which is illustrated in the following picture:



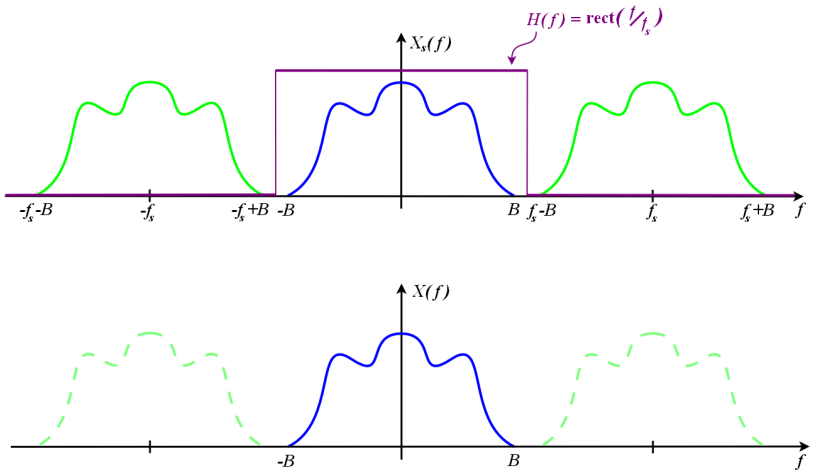

For bandlimited functions, which fulfill the so-called Nyquist Shannon sampling theorem sampling errors can be avoided:

**Theorem**

A signal $s(x)$ that has a bandlimited spectrum $\hat{s}(\xi)$ with bandwidth $B$ can be perfectly reconstructed after sampling $s$ with a sampling frequency $\xi_s > 2B$.

This perfect reconstruction is achieved by multiplying the Fourier spectrum with a rectangular function, which cuts out exactly one period of the periodic signal.

$$\hat{s}_{\text{reconstructed}}(\xi) = \hat{s}_{\text{sampled}}(\xi)\,\text{rect}(\frac{\xi}{\xi_s})$$



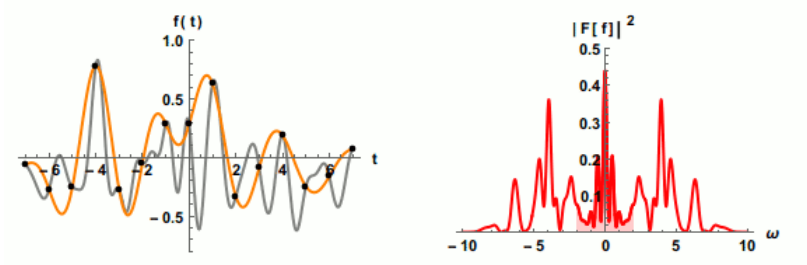

If we consider this filtering in spatial domain the multiplication becomes a convolution with the Fourier transform of the rect function, which is the sinc function:

$$\begin{aligned} s_{\text{reconstructed}}(x) &= (s_{\text{sampled}} * \xi_s \text{sinc}(\cdot\,\xi_s))(x) \\ &= \xi_s \int_{-\infty}^{\infty} s_{\text{sampled}}(\tilde{x})\text{sinc}((x-\tilde{x})\xi_s)\text{d}\tilde{x} \\ &= \xi_s \int_{-\infty}^{\infty} \left( \sum_{n=-\infty}^{\infty} s(\tilde{x})\delta(\tilde{x} - \frac{n}{\xi_s}) \right) \text{sinc}((x-\tilde{x})\xi_s)\text{d}\tilde{x} \\ &= \xi_s \sum_{n=-\infty}^{\infty} s(\frac{n}{\xi_s})\text{sinc}(\frac{x\xi_s - n}{\xi_s})\text{d}\tilde{x} \end{aligned}$$

This means that the discrete signal is replaced by shifted sinc functions. This is in fact an interpolation. However, the oscillating behavior of the sinc function leads to ringing behavior in image space.

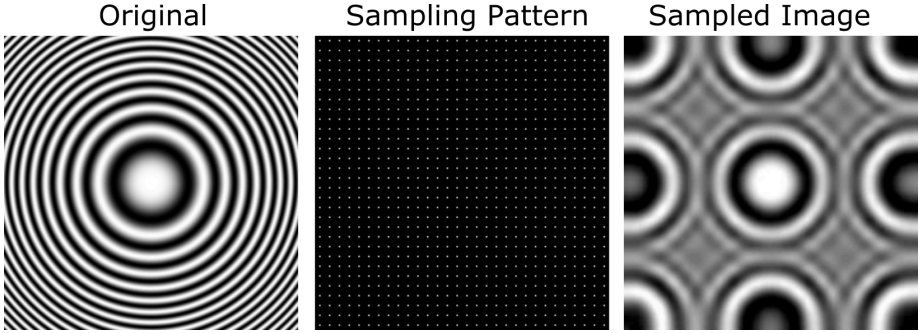
## Examples

We next look at some examples where aliasing error happen. The first example shows what happens to a 1D signal that is subsampled below the Nyquist theorem. While the sampling points are exactly on the gray function, the interpolating orange function looks only similar if the sampling frequency is chosen high enough.




### Moire Effect

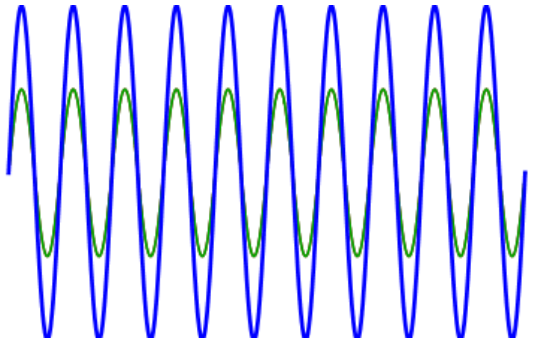
In the next example we look at the zone plane, which contains increasing frequency in radial direction. One can see that the discretized version looks fine in the center but towards the edges one can see strong artifacts since the wavehill frequency is higher then the sampling frequency. This generation of structural patterns because of aliasing errors is also called the Moire effect.



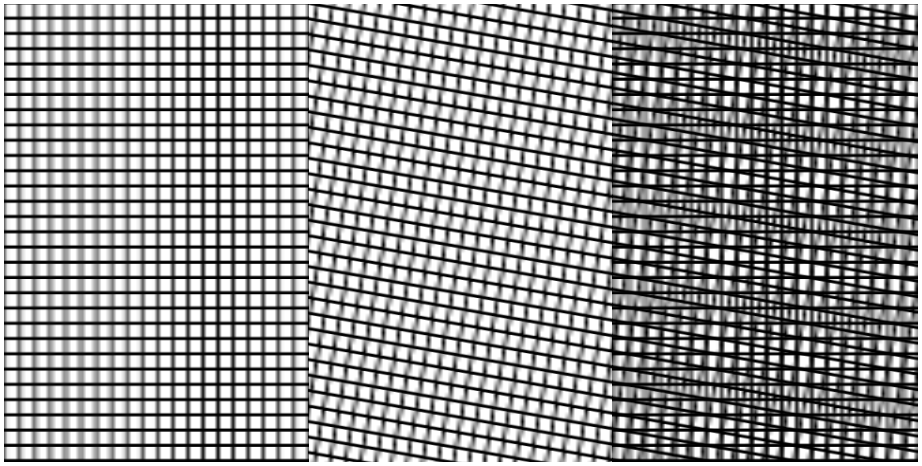

The moire effect happens in particular if regular patterns with a certain frequency are undersampled. There are various ways to showcase the Moire effect in image space.

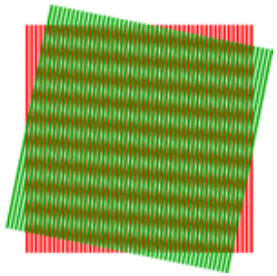
To analyse it better one can take two grids and rotate or shift them. Although both original grids are structured and have a homogeneous spatial frequency, the overlayed image shows a different frequency which we is due to interference, i.e. the sum of two similar sine functions leads to a modulation with a sine of a different frequency.

### Example: Interference Pattern



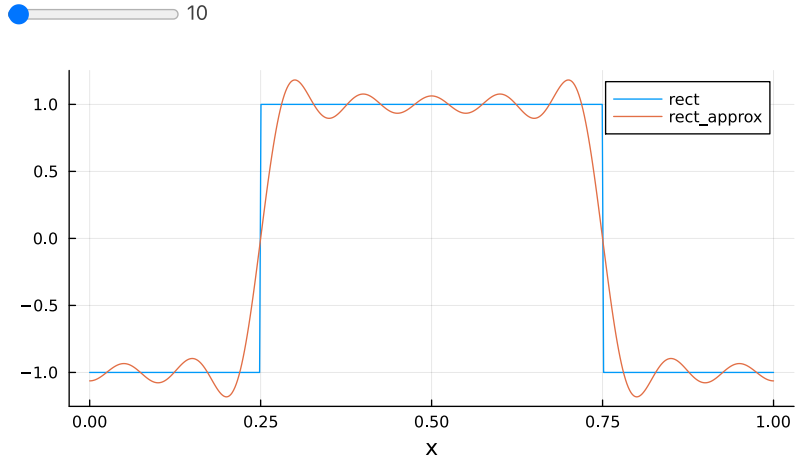
**Example: Rotated Grids**



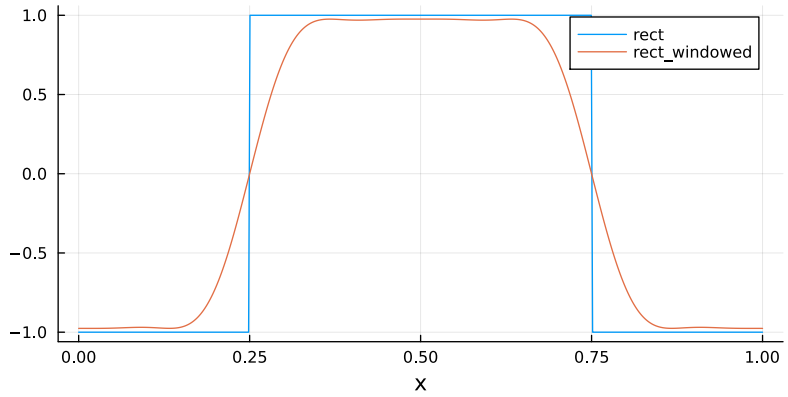


## 6.5.2 Avoiding Aliasing Errors

One way to mitigate aliasing errors is to sample high enough, and/or explicitly bandlimit the function *before* sampling or resampling. This can be done by the application of a rectangular function in Fourier space. However, this leads to the ringing artifact that we before:




Whenever we cut of in frequency space *sharply*, there will be ringing artifacts. But this can be avoided by using a different window function which is not so sharp as the rect function:




In this example we have used the Hann Window, which is defined as

$$w(n) = \frac{1}{2}\left[1 - \cos\left(\frac{2\pi n}{N-1}\right)\right]$$

where $n = 0, \ldots, N-1$ and $N$ is the number of samples in Fourier space. In our code example we used the sine representation of the Fourier series and thus only need to apply half of the window.

**Note**

Windowing is an important technique that you should always consider when using Fourier techniques in practice. Windowing enforces that Fourier coefficients go smoothly to zero, which avoids the truncation artifacts.

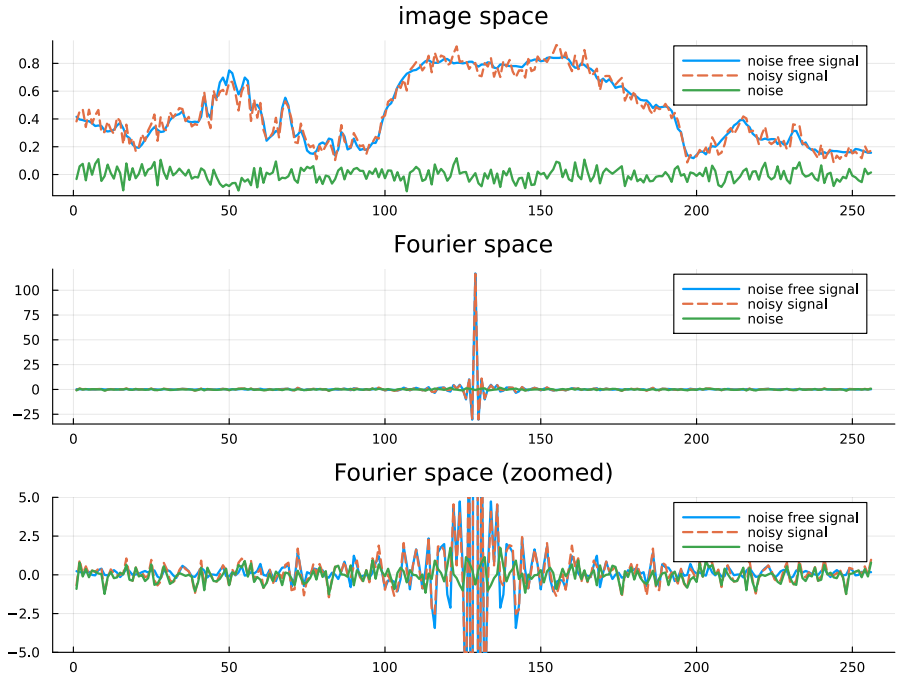
## 6.5.3 Truncation Error

The second error that we make when transitioning from the continuous to the discrete Fourier transform is that only a finite number of coefficients is spend for the Fourier space signal. This corresponds to cutting out the image in space, i.e. it is the reciprocal to the bandlimitation.

# 6.6 Smoothing Filters

In the next two section we investigate some concrete filters that we considered in the spatial filtering lecture and discuss how they look like in Fourier space. First we discuss smoothing filters.

Recall that we used smoothing filters in order to reduce the noise. Let us take a line from the Fabio image and put some noise on it:




### Observations

- In image space, the noise and the signal spread over the entire axis.
- In Fourier space, the signal is located around frequency zero and then decreases rapidly.
- Noise on the other hand is distributed equally in Fourier space.

All this means that the higher frequency parts mainly contain noise, while the central part of the frequency space contains the signal. The higher the noise, the smaller is the region we can trust. Consequently we can try separating the low and high frequency part using a filter. There are various ways to do so.

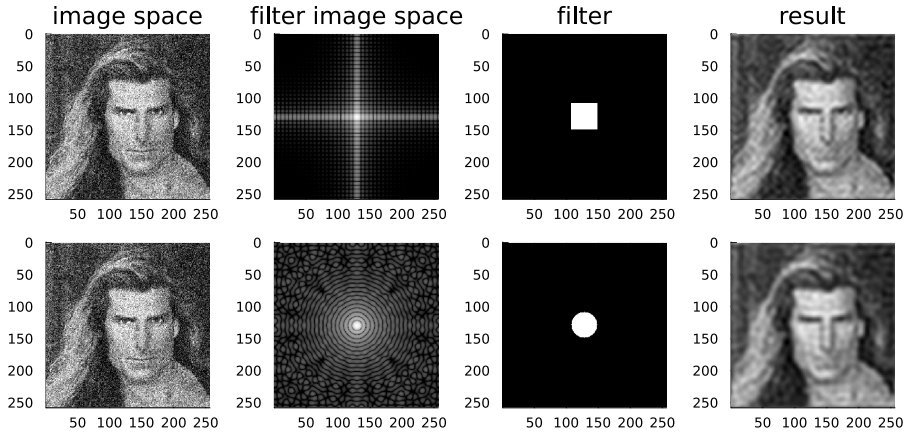
## 6.6.1 Ideal Low Pass

The ideal low pass can be written as

$$\hat{h}_D(u,v) = \begin{cases} 1 & \text{if } \|(u,v)^T\|_\infty < D \\ 0 & \text{otherwise} \end{cases}$$

One may also use an isotropic form of that

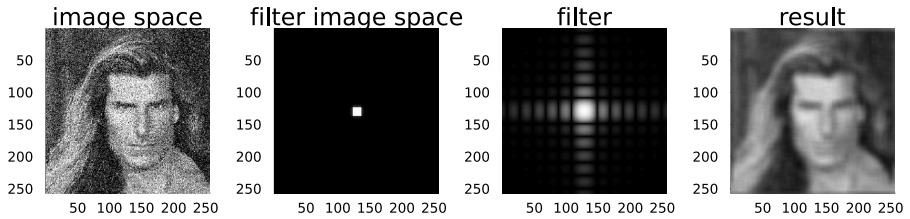
$$\hat{h}_D^{\text{isotropic}}(u,v) = \begin{cases} 1 & \text{if } \|(u,v)^T\|_2 < D \\ 0 & \text{otherwise} \end{cases}$$




So what we can see is the both filters do their job and reduce the noise. Interesting is how the filters look in image space. since $\mathbf{sinc}(x) \overset{\mathcal{F}}{\longleftrightarrow} \mathbf{rect}(\xi)$ it is clear that the 2D variant is either a tensor product of the sinc or an isotropic version of the sinc. This means:

- In principle the ideal low pass leads to a local averaging as we expect from a smooth kerne in image space.
- They are, however, oscillating, and spread over the entire image. This leads to patch like artifacts in image space.

Next let us switch the kernel and its Fourier transform:




This now is exactly the box filter that we considered in the last lecture. We can see that:

- It indeed acts as a low pass filter and lets through the central part of the Fourier space.
- However, it is not isotropic and the sinc like ringing lets certain high frequency regions pass that only contain noise.

## 6.6.2 Gaussian Filter

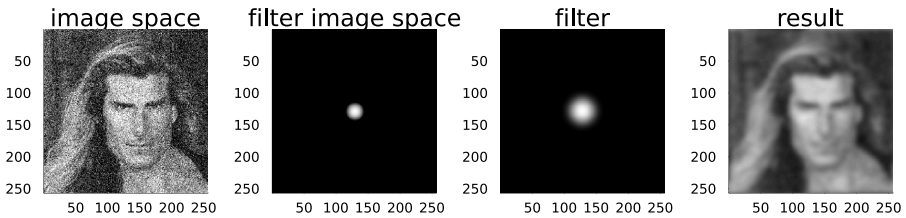
A much better alternative is, as you already know, the Gaussian filter. It was defined in image space as

$$h^\sigma_{\text{Gaussian}}(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

In frequency space on can show that the Gaussian is given by

$$\hat{h}^{\sigma}_{\text{Gaussian}}(u,v) = \exp\left(-2\pi^2\sigma^2(u^2+v^2)\right).$$

Let us see it in action:




### Observations

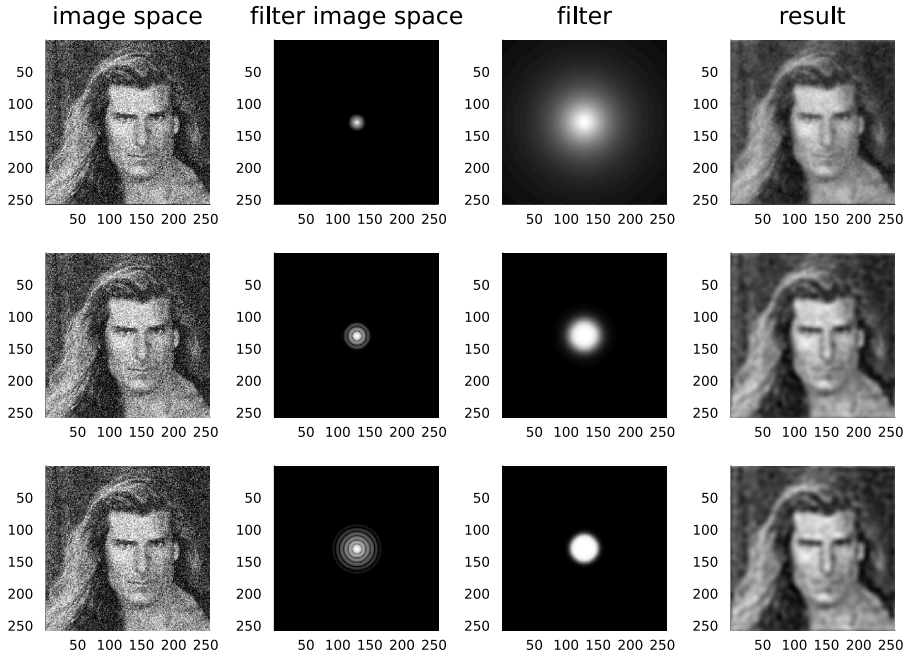
- Both the Fourier representation and the image space representation of the filter look much better.
- No sharp cut in Fourier space and no oscillations in both spaces.

## 6.6.3 Butterworth Filter

The Gauss filter has only one parameter for chosing the size of the kernel in both spaces. What cannot be adjusted is the *sharpness* when going from the pass band to the stop band. A filter that allows this is the butterworth filter:

$$\hat{h}^{D,n}_{\text{Butterworth}}(u,v) = \frac{1}{1+\left(\frac{\|(u,v)^T\|_2}{D}\right)^{2n}}$$

The following shows the filter in action for three different (increasing) $n$. The larger $n$ the sharper the filter. The size of the pass band is adjusted by $D$ (not shown).




### Observations

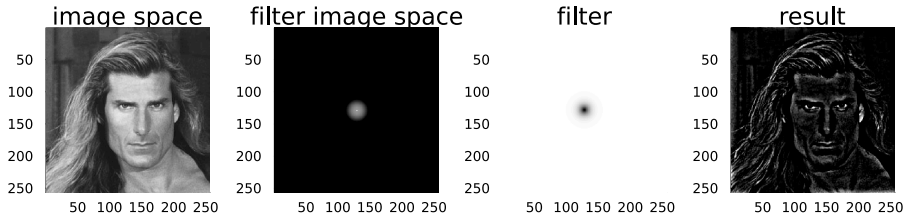
- The filter characteristic can be adjusted in a very fine granular fashion.
- Making the filter sharp in one domain always leads to ripple in the other domain. So there is no free lunch.

# 6.7 Sharpening Filters

We can design a sharpening filter using the general approach

$$\hat{h}_{\text{HP}}(u,v) = 1 - \hat{h}_{\text{LP}}(u,v)$$

Let us do this for the butterworth filter:




This filter is a high pass filter and in this way only keeps the edge information of the image. In case that we want instead a *sharpening* operation we can use that image and add it to the original image. The original image can be written as

$$f(x,y) = (f * \delta)(x,y).$$

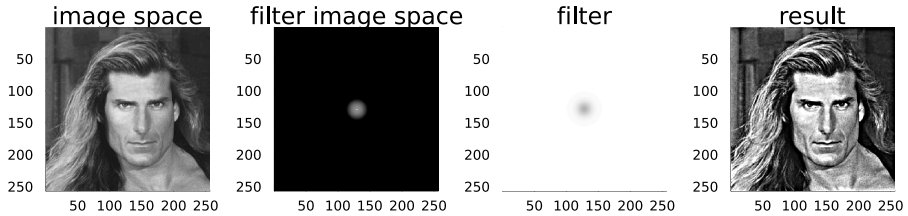
In frequency space this means

$$\hat{f}(u,v) = \hat{f}(u,v) \cdot 1(u,v).$$

Due to linearity we can thus, instead of adding the original image to the sharpened image, add $\mathbf{1}$ to the filter kernel yielding

$$\hat{h}_{\text{sharpening}}(u,v) = 1 + \alpha \hat{h}_{\text{HP}}(u,v) = 1 + \alpha(1 - \hat{h}_{\text{LP}}(u,v)).$$

The parameter $\alpha$ here is used to adjust the amount of the sharpened image that is added to the original one.



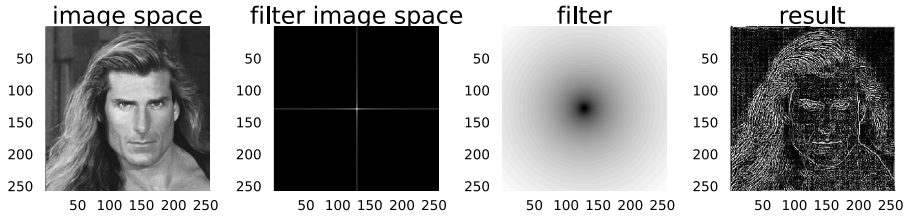

## 6.7.1 Laplacian

Recall that the Laplacian was defined as

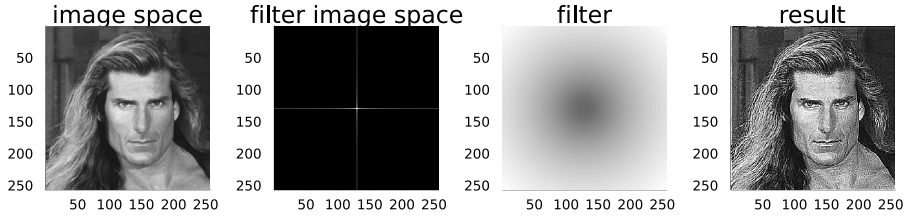
$$\Delta f = \frac{\partial^2 f}{\partial^2 x} + \frac{\partial^2 f}{\partial^2 y}$$

We know that a spatial derivative can be expressed in Fourier space as a linear weighting. Thus the Laplacian can be expressed in Fourier space by the filter

$$\hat{h}_{\text{Laplacian}}(u,v) = -4\pi^2(u^2 + v^2)$$




Again if we add 1 to the filter and appropriately weight the high pass part we can obtain a sharpening filter:




# 6.8 Implementation

We next discuss some implementation details when applying the discrete Fourier transform.

## 6.8.1 fftshift

When we compare the Fourier series

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{2\pi i \frac{nx}{P}}$$

with the (inverse) DFT

$$s(x) = \frac{1}{N} \sum_{\xi=0}^{N-1} \hat{s}(\xi) e^{2\pi i \frac{x\xi}{N}}$$

it gets clear is that the summation for the DFT is not centered around zero while it is for the Fourier series. The question is why?

**Answer:** Because it is the convention. In fact, we know that the discrete Fourier coefficients are periodic

$$s(\xi) = s(\xi + nN)$$

Thus, instead of the index range $[0, N-1]$ one can also take $[-\frac{N}{2}, \frac{N}{2} - 1]$. This is often more natural as the zero frequency is then located in the center. Switching between both representations is done by the `fftshift` operation, which is available in most FFT software libraries:

```
▸ [3, 4, 1, 2]
1 fftshift([1,2,3,4])
```

```
▸ [4, 5, 1, 2, 3]
1 fftshift([1,2,3,4,5])
```

This is how it works in 2D. There the two quarters are exchanged:

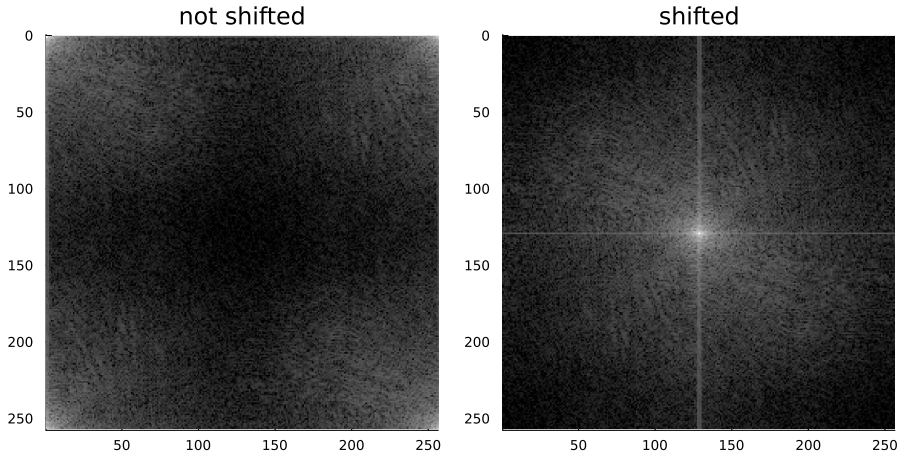
```
2×2 Matrix{Int64}:
 1  2
 3  4
```

```
1 [1 2; 3 4]
```

```
2×2 Matrix{Int64}:
 4  3
 2  1
```

```
1 fftshift([1 2; 3 4])
```

And we can also do this on images:




> **Note**
>
> One can apply a filtering either in the shifted domain or in the regular domain. Important is just that the image and the filter are treated the same way. Also it is important to account for the shift before doing an inverse Fourier transformation.

## 6.8.2 Padding

The convolution theorem in the continuous domain in principle has an analog formulation in the discrete setting. It is, however, only valid for the cyclic convolution:

$$(s * h)(x) = \sum_{y=0}^{N-1} s(y)h((x-y) \bmod N)$$

In fact, this linear transformation can be written as a matrix-vector multiplication with a circulant matrix.

If you compare this with our discussions on *boundary conditions* in the last lecture this means that we actually can only apply the convolution if we consider periodic boundary conditions.

But there is a way to use arbitary boundary conditions and still use the discrete convolution theorem. The idea is to appropriately extend the original image (so-called *padding*) that the periodic convolution is the same as the non-periodic one. This is done by adding zeros outside. Depending on the dimensions of $s$ and $h$ one needs to take more or less zeros.

## 6.8.3 Fast Fourier Transform

Until now we have not discussed in any way the computational cost of the convolution and the (discrete) Fourier transform. For a signal/image with $N$ entries and a kernel with $N$ entries, the computational cost for both convolutions and DFTs is $\mathcal{O}(N^2)$ because a sum over $N$ elements needs to be carried out for each of the $N$ output elements.

Quadratic cost in these operations is much too high in practice. This is both for *computational* but also for *memory* reasons. Lets consider that we have an $6,000 \times 6,000$ sized image where each pixel encodes a 32bit floating point number. Then the image is requires 137.3291015625 MB of memory. How large would the convolution or Fourier matrix then be? It would be 4.604316927725449 PB. So even if you have a PC with 1 TB of main memory, this is 4600 times more.

So the intermediate conclusion is that we cannot explicitely arange the transformation matrix but need to perform the transformations in a matrix-free fashion. But also the computational cost is much too high when evaluating the sums directly.

Fortunately, there is an alternative, which is called the Fast Fourier transform. Basically the FFT is an algorithm that allows to carry out the DFT with only $\mathcal{O}(N \log N)$ operations. To this end it exploits that a DFT of length $N$ can be expressed using two DFTs of length $\frac{N}{2}$. Using a recursion scheme one then ends up at $\mathcal{O}(N \log N)$.

> **Note**
>
> In this lecture we do not explain the FFT itself. If you have not seen a derivation of the FFT we strongly encourage you to investigate the topic. For the remaining of this lecture we use the FFT as a black-box algorithm though.

## 6.8.4 FFT based Convolution

With the knowledge of the arithmetic complexity of the FFT we can now think about the cost of an FFT based convolution. It requires $\mathcal{O}(N \log N)$ for going into the Fourier space, $\mathcal{O}(N)$ for applying the filter (scalar-wise multiplication) and $\mathcal{O}(N \log N)$ for going back into image domain. In sum this is still only $\mathcal{O}(N \log N)$. Thus, an FFT based convolution can be much faster than a directly evaluated convolution, which costs $\mathcal{O}(N^2)$.

However, in practice the image and the kernel are not always of the same size but the kernel is much smaller. In these cases an image-domain implementation can actually be faster than a Fourier-based implementation. Sophisticated libraries like ImageFiltering.jl implement both and automatically chose based on the kernel size, which strategy to chose (see documentation).

# 6.9 Wrapup

- Fourier representation of signals allows for a better understanding of filtering.
- Sampling artifacts can be explained and resolved using Fourier analysis.
- Filtering can be more efficiently applied in Fourier space.
- Understanding the the difference between continuous and discrete Fourier transformations is crutial for dealing with them in practice.