Fourier Series and Fourier Transform
An Introduction

Michael Figl

Center for Medical Physics and Biomedical Engineering
Medical University of Vienna
Introduction
We want to recall some definitions and theorems from linear algebra, mainly to clearly define the assumptions needed.
Some remarks on $\mathbb{R}^n$

We recall $\mathbb{R}^n := \{(a_1, a_2, \ldots) : x_i \in \mathbb{R}\}$

the standard basis consists of the vectors $e_i$, with a 1 at position $i$, $e_i = (0, \ldots, 0, 1, 0, \ldots, 0)$

the inner product is defined by $(x, y) := \sum x_i y_i$

if we take a vector $a = (a_1, a_2, \ldots)$ we get $(a, e_i) = a_i$

therefore $a = \sum (a, e_i) e_i$

What if we have many basis elements and don't want to sum everything up?
Some remarks on $\mathbb{R}^n$

- We recall $\mathbb{R}^n := \{(a_1, a_2, \ldots) : x_i \in \mathbb{R}\}$
Some remarks on $\mathbb{R}^n$

- We recall $\mathbb{R}^n := \{(a_1, a_2, \ldots) : x_i \in \mathbb{R}\}$
- the standard basis consists of the vectors $e_i$, with a 1 at position $i$, $e_i = (0, \ldots, 0, 1, 0, \ldots, 0)$
Some remarks on $\mathbb{R}^n$

- We recall $\mathbb{R}^n := \{(a_1, a_2, \ldots) : x_i \in \mathbb{R}\}$
- the standard basis consists of the vectors $e_i$, with a 1 at position $i$, $e_i = (0, \ldots, 0, 1, 0, \ldots, 0)$
- the inner product is defined by $(x, y) := \sum x_i y_i$
Some remarks on $\mathbb{R}^n$

- We recall $\mathbb{R}^n := \{(a_1, a_2, \ldots) : x_i \in \mathbb{R}\}$
- the standard basis consists of the vectors $e_i$, with a 1 at position $i$, $e_i = (0, \ldots, 0, 1, 0, \ldots, 0)$
- the inner product is defined by $(x, y) := \sum x_i y_i$
- if we take a vector $a = (a_1, a_2, \ldots)$ we get $(a, e_i) = a_i$
Some remarks on \( \mathbb{R}^n \)

- We recall \( \mathbb{R}^n := \{(a_1, a_2, \ldots) : x_i \in \mathbb{R}\} \)
- the standard basis consists of the vectors \( e_i \), with a 1 at position \( i \), \( e_i = (0, \ldots, 0, 1, 0, \ldots, 0) \)
- the inner product is defined by \( (x, y) := \sum x_i y_i \)
- if we take a vector \( a = (a_1, a_2, \ldots) \) we get \( (a, e_i) = a_i \)
- therefore \( a = \sum (a, e_i)e_i \)
Some remarks on $\mathbb{R}^n$

- We recall $\mathbb{R}^n := \{(a_1, a_2, \ldots) : x_i \in \mathbb{R}\}$
- the standard basis consists of the vectors $e_i$, with a 1 at position $i$, $e_i = (0, \ldots, 0, 1, 0, \ldots, 0)$
- the inner product is defined by $(x, y) := \sum x_i y_i$
- if we take a vector $a = (a_1, a_2, \ldots)$ we get $(a, e_i) = a_i$
- therefore $a = \sum (a, e_i) e_i$
Some remarks on $\mathbb{R}^n$

- We recall $\mathbb{R}^n := \{(a_1, a_2, \ldots) : x_i \in \mathbb{R}\}$
- the standard basis consists of the vectors $e_i$, with a 1 at position $i$, $e_i = (0, \ldots, 0, 1, 0, \ldots, 0)$
- the inner product is defined by $(x, y) := \sum x_i y_i$
- if we take a vector $a = (a_1, a_2, \ldots)$ we get $(a, e_i) = a_i$
- therefore $a = \sum (a, e_i) e_i$

What if we have many basis elements and don’t want to sum everything up?
Best Approximation in $\mathbb{R}^n$, I

Assume we have a vector $r = \sum_{i=1}^{N} (r, e_i) e_i$ but as $N$ is very big, we want just want to compute $y = \sum_{i=1}^{n} (r, e_i) e_i$, with $n \ll N$.

What can generally be said about $y$?
Best Approximation in $\mathbb{R}^n$, I

Assume we have a vector

$$r = \sum_{i=1}^{N} (r, e_i) e_i$$

but as $N$ is very big, we want just want to compute

$$y = \sum_{i=1}^{n} (r, e_i) e_i,$$

with $n \ll N$.

What can generally be said about $y$?
Best Approximation in $\mathbb{R}^n$, I

Assume we have a vector

$$r = \sum_{i=1}^{N} (r, e_i) e_i$$

but as $N$ is very big, we want just want to compute

$$y = \sum_{i=1}^{n} (r, e_i) e_i, \text{ with } n << N$$
Assume we have a vector

\[ r = \sum_{i=1}^{N} (r, e_i) e_i \]

but as \( N \) is very big, we want just want to compute

\[ y = \sum_{i=1}^{n} (r, e_i) e_i, \text{ with } n \ll N \]

What can generally be said about \( y \)?
Best Approximation in $\mathbb{R}^n$, II

$$r - y = \sum_{i=n+1}^{N} (r, e_i)e_i \perp V$$

especially $r - y \perp y - y' \in V$

for every $y' \in V$
Best Approximation in $\mathbb{R}^n$, II

$$r - y = \sum_{i=n+1}^{N} (r, e_i) e_i \perp V$$
especially $$r - y \perp y - y' \in V$$
for every $$y' \in V$$

Pythagoras’ theorem givens

$$\|y' - r\|^2 = \|y - r\|^2 + \|y - y'\|^2$$
Best Approximation in $\mathbb{R}^n$, II

$r - y = \sum_{i=n+1}^{N}(r, e_i)e_i \perp V$

especially $r - y \perp y - y' \in V$

for every $y' \in V$

Pythagoras’ theorem gives

$\|y' - r\|^2 = \|y - r\|^2 + \|y - y'\|^2$

Theorem

If we define $V = \langle e_i, i = 1, \ldots n \rangle$, then $y = \sum_{i=1}^{n}(r, e_i)e_i$ is the best approximation of $r$ in $V$. 

Generalisation

What did we need to prove the approximation property?

▶ an inner product space
▶ an orthogonal basis
Generalisation

What did we need to prove the approximation property?
Generalisation

What did we need to prove the approximation property?

- an inner product space
Generalisation

What did we need to prove the approximation property?

- an inner product space
- an orthogonal basis
Inner Product Space, I

Definition (Inner Product Space)
A vector space $V$ (over the field $F$), with an inner product $(x, y) : V \times V \to F$ is called an inner product space.

The inner product is symmetric

\[(x, y) = (y, x)\]

bilinear

\[(x + \lambda y, z) = (x, z) + \lambda (y, z)\]

positive-definite

\[\forall x \in V \setminus \{0\}, \quad (x, x) > 0\]
Inner Product Space, I

A not very surprising definition:
A not very surprising definition:

**Definition (Inner Product Space)**

A vector space $V$ (over the field $F$), with an inner product $(x, y) : V \times V \rightarrow F$ is called an *inner product space*. 

The inner product is

- symmetric: $(x, y) = (y, x)$
- bilinear: $(x + \lambda y, z) = (x, z) + \lambda (y, z)$
- positive-definite: $\forall x \in V \setminus \{0\}, (x, x) > 0$
A not very surprising definition:

**Definition (Inner Product Space)**

A vector space $V$ (over the field $F$), with an inner product $(x, y) : V \times V \rightarrow F$ is called an *inner product space*. The inner product is
A not very surprising definition:

**Definition (Inner Product Space)**

A vector space $V$ (over the field $F$), with an inner product $(x, y) : V \times V \to F$ is called an *inner product space*. The inner product is

- **symmetric** $(x, y) = (y, x)$
A not very surprising definition:

**Definition (Inner Product Space)**

A vector space $V$ (over the field $F$), with an inner product $(x, y) \in V \times V \rightarrow F$ is called an *inner product space*. The inner product is

- **symmetric** $(x, y) = (y, x)$
- **bilinear** $(x + \lambda y, z) = (x, z) + \lambda (y, z)$
A not very surprising definition:

**Definition (Inner Product Space)**

A vector space $V$ (over the field $F$), with an inner product $(x, y) : V \times V \rightarrow F$ is called an *inner product space*. The inner product is

- **symmetric** $(x, y) = (y, x)$
- **bilinear** $(x + \lambda y, z) = (x, z) + \lambda (y, z)$
- **positive-definite** $\forall x \in V \setminus \{0\} \ (x, x) > 0$
Inner Product Space, II

Definition

\[ \| x \| : = \sqrt{(x, x)} \]

Orthogonality

\[ x \perp y : \iff (x, y) = 0 \]

Angle

\[ x \angle y : = \arccos \frac{(x, y)}{\| x \| \| y \|} \]
We can now define
We can now define

**Definition**

$\text{Norm } \| x \| := \sqrt{(x, x)}$
We can now define

**Definition**

Norm \[ \|x\| := \sqrt{(x, x)} \]

Orthogonality \[ x \perp y :\iff (x, y) = 0 \]
We can now define

**Definition**

**Norm** \( \| x \| := \sqrt{(x, x)} \)

**Orthogonality** \( x \perp y :\Leftrightarrow (x, y) = 0 \)

**Angle** \( x \angle y := \arccos \left( \frac{(x, y)}{\|x\| \|y\|} \right) \)
Inner Product Space, Examples

Example $\mathbb{R}^n$ using $(a, b) := \sum a_i b_i$

Example $C[0,1]$ using $(f, g) := \int_0^1 fg$

Example $\mathbb{C}^n$ using $(a, b) := \sum a_i b_i$, sesquilinear!

$L^2$ := \{ $(x_i)_{i \in \mathbb{N}} : \sum_{i=0}^{\infty} x_i^2 < \infty$ \}, with $(a, b) := \sum_{i=0}^{\infty} a_i b_i$

$L^2$ := \{ measurable $f$ : $\int f^2 < \infty$ \}, with $(f, g) := \int fg$
Inner Product Space, Examples

Example

\[ \mathbb{R}^n \text{ using } (a, b) := \sum a_i b_i \]
Example

\( \mathbb{R}^n \) using \((a, b) := \sum a_i b_i\)

\( C[0, 1] \) using \((f, g) := \int_0^1 fg\)
Inner Product Space, Examples

Example

\( \mathbb{R}^n \) using \((a, b) := \sum a_i b_i\)

\( C[0, 1] \) using \((f, g) := \int_0^1 fg\)

\( \mathbb{C}^n \) using \((a, b) := \sum a_i \overline{b_i} \), sesquilinear!
Example

\( \mathbb{R}^n \) using \((a, b) := \sum a_i b_i\)

\( C[0, 1] \) using \((f, g) := \int_0^1 fg\)

\( \mathbb{C}^n \) using \((a, b) := \sum a_i \overline{b_i} \), sesquilinear!

\( l_2 := \{(x_i)_{i \in \mathbb{N}} : \sum_{i=0}^{\infty} x_i^2 < \infty\} \), with \((a, b) := \sum_{i=0}^{\infty} a_i \overline{b_i}\)
Inner Product Space, Examples

Example

\( \mathbb{R}^n \) using \((a, b) := \sum a_i b_i\)

\( C[0, 1] \) using \((f, g) := \int_0^1 fg\)

\( \mathbb{C}^n \) using \((a, b) := \sum a_i \overline{b_i}, \text{ sesquilinear!}\)

\( l_2 := \{(x_i)_{i \in \mathbb{N}} : \sum_{i=0}^{\infty} x_i^2 < \infty\}, \text{ with } (a, b) := \sum_{i=0}^{\infty} a_i \overline{b_i}\)

\( L_2 := \{f \text{ measurable} : \int f^2 < \infty\}, \text{ with } (f, g) := \int f \overline{g}\)
Bessel’s Inequality
Bessel’s Inequality

Spaces like $L_2$ or $l_2$ have infinite dimensions, therefore the orthogonal system can also be infinite: $e_i, \ i = 1, 2, \ldots$
Bessel’s Inequality

Spaces like $L_2$ or $l_2$ have infinite dimensions, therefore the orthogonal system can also be infinite: $e_i, \ i = 1, 2, \ldots$

Theorem (Bessel’s inequality)

$$\|f\|^2 \geq \sum |(f, e_i)|^2,$$

i.e. the series $((f, e_i))_i \in l_2$
Bessel’s Inequality

Spaces like $L_2$ or $l_2$ have infinite dimensions, therefore the orthogonal system can also be infinite: $e_i, \ i = 1, 2, \ldots$

Theorem (Bessel’s inequality)

$$\|f\|^2 \geq \sum |(f, e_i)|^2, \ i.e. \ the \ series \ ((f, e_i))_i \in l_2$$

$$0 \leq (f - \sum_{i=1}^{n} (f, e_i)e_i, f - \sum_{k=1}^{n} (f, e_k)e_k)$$

$$= \|f\|^2 - 2(f, \sum_{i=1}^{n} (f, e_i)e_i) + (\sum_{i=1}^{n} (f, e_i)e_i, \sum_{k=1}^{n} (f, e_k)e_k)$$

$$= \|f\|^2 - 2 \sum_{i=1}^{n} |(f, e_i)|^2 + \sum_{i,k} (f, a_i)(f, a_k) (e_i, e_k)$$

$$= \|f\|^2 - \sum_{i=1}^{n} |(f, e_i)|^2$$
$L_2[-\pi, \pi]$, Basis
$L_2[-\pi, \pi]$, Basis

We know want to find an orthogonal basis for the $L_2[-\pi, \pi]$. It will be based on trigonometric functions.
\(L_2[-\pi, \pi], \text{ Basis}\)

We know want to find an orthogonal basis for the \(L_2[-\pi, \pi]\). It will be based on trigonometric functions.

**Theorem (orthogonality relations)**

*for \(a, b \in \mathbb{N}\)*

\[
\frac{1}{\pi} \int_{-\pi}^{\pi} \cos ax \cos bx = \begin{cases} 2 & \text{if } a = b = 0 \\ \delta_{ab} & \text{else} \end{cases}
\]

\[
\frac{1}{\pi} \int_{-\pi}^{\pi} \sin ax \sin bx = \delta_{ab}
\]

\[
\frac{1}{\pi} \int_{-\pi}^{\pi} \sin ax \cos bx = 0
\]
$L_2[-\pi, \pi], \text{ Basis}$

We know want to find an orthogonal basis for the $L_2[-\pi, \pi]$. It will be based on trigonometric functions.

**Theorem (orthogonality relations)**

For $a, b \in \mathbb{N}$

\[
\frac{1}{\pi} \int_{-\pi}^{\pi} \cos ax \cos bx = \begin{cases} 
2 & \text{if } a = b = 0 \\
\delta_{ab} & \text{else}
\end{cases}
\]

\[
\frac{1}{\pi} \int_{-\pi}^{\pi} \sin ax \sin bx = \delta_{ab}
\]

\[
\frac{1}{\pi} \int_{-\pi}^{\pi} \sin ax \cos bx = 0
\]

We search for the coefficients of

\[
f(x) \sim \frac{a_0}{2} + \sum_{k} a_k \cos kx + \sum_{k} b_k \sin kx
\]
\( L_2[-\pi, \pi], \text{ Basis} \)
We search for the coefficients of

\[ f(x) \sim \frac{a_0}{2} + \sum_k a_k \cos kx + \sum_k b_k \sin kx \]
We search for the coefficients of
\[ f(x) \sim \frac{a_0}{2} + \sum_k a_k \cos kx + \sum_k b_k \sin kx \]

**Theorem (Fourier Coefficients)**

for \( a, b \in \mathbb{N} \)

\[
\begin{align*}
    a_k &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos ax \\
    b_k &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin ax
\end{align*}
\]
Approximation of an Rectangle, I
Approximation of an Rectangle, I

The simple rectangle function

\[ f(n) = \begin{cases} 
1, & \text{if } -\frac{\pi}{2} \leq x \leq \frac{\pi}{2} \\
0, & \text{else} 
\end{cases} \]

has the Fourier coefficients:

\[ a_k = \frac{1}{\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} f(x) \cos(kx) \, dx = \begin{cases} 
1, & k = 0 \\
\frac{2}{k\pi} \sin(k\frac{\pi}{2}), & \text{if } k \text{ is even} \\
0, & \text{else} 
\end{cases} \]

\[ b_k = \frac{1}{\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} f(x) \sin(kx) \, dx = 0 \]

\[ n=1 \]
Approximation of an Rectangle, I

The simple rectangle function

\[ f(n) = \begin{cases} 1, & \text{if } -\frac{\pi}{2} \leq x \leq \frac{\pi}{2} \\ 0, & \text{else} \end{cases} \]

has the Fourier coefficients:
Approximation of an Rectangle, I

The simple rectangle function

\[ f(n) = \begin{cases} 1, & \text{if } -\frac{\pi}{2} \leq x \leq \frac{\pi}{2} \\ 0, & \text{else} \end{cases} \]

has the Fourier coefficients:

\[ a_k = \frac{1}{\pi} \int_{-\pi/2}^{\pi/2} f(x) \cos(kx) \, dx = \frac{1}{\pi} \left[ \frac{\sin(kx)}{k} \right]_{-\pi/2}^{\pi/2} = \begin{cases} 0, & \text{if } k \text{ is even} \\ \frac{2}{k\pi}, & \text{else} \end{cases} \]

\[ b_k = \frac{1}{\pi} \int_{-\pi/2}^{\pi/2} f(x) \sin(kx) \, dx = 0 \]

For \( n=3 \):

\[ n=3 \]
Approximation of an Rectangle, I

The simple rectangle function

\[ f(n) = \begin{cases} 
1, & \text{if } -\frac{\pi}{2} \leq x \leq \frac{\pi}{2} \\
0, & \text{else}
\end{cases} \]

has the Fourier coefficients:
Approximation of an Rectangle, I

The simple rectangle function

\[ f(n) = \begin{cases} 
1, & \text{if } -\frac{\pi}{2} \leq x \leq \frac{\pi}{2} \\
0, & \text{else}
\end{cases} \]

has the Fourier coefficients:

for \( k \neq 0 \): 
\[
a_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos kx = \frac{1}{\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \cos kx \]

\[
= \frac{\sin kx}{k\pi} \bigg|_{-\frac{\pi}{2}}^{\frac{\pi}{2}} = \frac{2}{k\pi} \sin k\frac{\pi}{2} = \begin{cases} 
0, & \text{if } k \text{ is even} \\
\frac{2}{k\pi} (-1)^{\frac{k-1}{2}}, & \text{else}
\end{cases}
\]

\[ a_0 = 1, \ b_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin kx = 0 \]
Approximation of an Rectangle, 1

The simple rectangle function

\[ f(n) = \begin{cases} 
1, & \text{if } -\frac{\pi}{2} \leq x \leq \frac{\pi}{2} \\
0, & \text{else} 
\end{cases} \]

has the Fourier coefficients:

for \( k \neq 0 \): \( a_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos kx = \frac{1}{\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \cos kx \)

\[ = \left. \frac{\sin kx}{k\pi} \right|_{-\frac{\pi}{2}}^{\frac{\pi}{2}} = \frac{2}{k\pi} \sin k\frac{\pi}{2} = \begin{cases} 
0, & \text{if } k \text{ is even} \\
\frac{2}{k\pi} (-1)^{\frac{k-1}{2}}, & \text{else} 
\end{cases} \]

\[ a_0 = 1, \ b_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin kx = 0 \]
The simple rectangle function

\[ f(n) = \begin{cases} 
1, & \text{if } -\frac{\pi}{2} \leq x \leq \frac{\pi}{2} \\
0, & \text{else} 
\end{cases} \]

has the Fourier coefficients:

for \( k \neq 0 \):

\[ a_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos kx = \frac{1}{\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \cos kx \]

\[ = \frac{\sin kx}{k\pi} \bigg|_{-\frac{\pi}{2}}^{\frac{\pi}{2}} = \frac{2}{k\pi} \sin k\frac{\pi}{2} = \begin{cases} 
0, & \text{if } k \text{ is even} \\
\frac{2}{k\pi}(-1)^{\frac{k-1}{2}}, & \text{else} 
\end{cases} \]

\[ a_0 = 1, \quad b_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin kx = 0 \]
Approximation of an Rectangle, II

Therefore the Fourier series for the rectangle function is

\[ f(x) = \frac{1}{2} + \sum_{k=1}^{\infty} \frac{(-1)^{k-1}}{2k-1} \cos(2(k-1)x) \]

If we set \( x = 0 \) we get

\[ \frac{\pi}{4} = \sum_{k=1}^{\infty} \frac{(-1)^{k-1}}{2k-1} = \frac{1}{3} - \frac{1}{5} + \frac{1}{7} \ldots \]
therefore the Fourier series for the rectangle function is

\[ f(x) = \frac{1}{2} + \sum_{k=1}^{\infty} \frac{2(-1)^{k-1}}{\pi} \frac{\cos(2k - 1)x}{2k - 1} \]
therefore the Fourier series for the rectangle function is

\[ f(x) = \frac{1}{2} + \sum_{k=1}^{\infty} \frac{2(-1)^{k-1}}{\pi} \frac{\cos(2k-1)x}{2k-1} \]

if we set \( x = 0 \) we get

\[ \frac{\pi}{4} = \sum_{k=1}^{\infty} \frac{(-1)^{k-1}}{2k-1} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} \ldots \]
Convergence of Fourier Series

As seen with the rectangle function pointwise convergence cannot be expected everywhere (Gibbs phenomena), nevertheless

Theorem

We denote by $S_n$ the Fourier series with $n$ terms

The $L^2$ norm

$\forall f \in L^2; \| f - S_n \| = \int \| f - S_n \|^2 \to 0$ (Riesz-Fischer)

uniform

$\forall f \in C^1; S_n \to f$ uniformly (Dirichlet)

pointwise

$\forall f \in L^2; S_n \to f$ pointwise a.e. (Carleson)

with Riesz-Fischer we have

Theorem (Parseval equality)

$\| f \|^2 = \sum |(f, e^{i \omega})|^2$, i.e. $F: L^2 \to l^2$ defined by $f \mapsto (f, e^{i \omega})$ is isometric.
Convergence of Fourier Series

As seen with the rectangle function pointwise convergence cannot be expected everywhere (Gibbs phenomena), nevertheless...

With Riesz-Fischer we have

Theorem (Parseval equality)

\[ \| f \|_2^2 = \| S_n \|_2^2 \]

i.e. \( F : L^2 \rightarrow l^2 \) defined by \( f \mapsto (f, e^{i\theta}) \) is isometric.
Convergence of Fourier Series

As seen with the rectangle function pointwise convergence cannot be expected everywhere (Gibbs phenomena), nevertheless

**Theorem**

*We denote by $S_n$ the Fourier series with $n$ terms*
Convergence of Fourier Series

As seen with the rectangle function pointwise convergence cannot be expected everywhere (Gibbs phenomena), nevertheless

Theorem

We denote by $S_n$ the Fourier series with $n$ terms

$L_2$ norm $\forall f \in L_2; \| f - S_n \| = \int \| f - S_n \|^2 \to 0$ (Riesz-Fischer)
Convergence of Fourier Series

As seen with the rectangle function pointwise convergence cannot be expected everywhere (Gibbs phenomena), nevertheless

Theorem
We denote by $S_n$ the Fourier series with $n$ terms

$L_2$ norm $\forall f \in L_2; \|f - S_n\| = \int \|f - S_n\|^2 \to 0$ (Riesz-Fischer)

uniform $\forall f \in C^1; S_n \to f$ uniformly (Dirichlet)
Convergence of Fourier Series

As seen with the rectangle function pointwise convergence cannot be expected everywhere (Gibbs phenomena), nevertheless

Theorem
We denote by $S_n$ the Fourier series with $n$ terms

$L_2$ norm $\forall f \in L_2; \| f - S_n \| = \int \| f - S_n \|^2 \to 0$ (Riesz-Fischer)

uniform $\forall f \in C^1; S_n \to f$ uniformly (Dirichlet)

pointwise $\forall f \in L_2; S_n \to f$ pointwise a.e. (Carleson)
Convergence of Fourier Series

As seen with the rectangle function pointwise convergence cannot be expected everywhere (Gibbs phenomena), nevertheless

**Theorem**

*We denote by $S_n$ the Fourier series with $n$ terms*

$L_2$ norm  $\forall f \in L_2; \|f - S_n\| = \int \|f - S_n\|^2 \to 0$ (Riesz-Fischer)

uniform  $\forall f \in C^1; S_n \to f$ uniformly (Dirichlet)

pointwise  $\forall f \in L_2; S_n \to f$ pointwise a.e. (Carleson)
Convergence of Fourier Series

As seen with the rectangle function pointwise convergence cannot be expected everywhere (Gibbs phenomena), nevertheless

**Theorem**

We denote by $S_n$ the Fourier series with $n$ terms

- $L_2$ norm $\forall f \in L_2; \|f - S_n\| = \int \|f - S_n\|^2 \to 0$ (Riesz-Fischer)
- uniform $\forall f \in C^1; S_n \to f$ uniformly (Dirichlet)
- pointwise $\forall f \in L_2; S_n \to f$ pointwise a.e. (Carleson)

with Riesz-Fischer we have

**Theorem (Parseval equality)**

$$\|f\|^2 = \sum |(f, e_i)|^2, \text{ i.e. } F : L_2 \to l_2 \text{ defined by } f \mapsto (f, e_i); \text{ is isometric.}$$
Complex Fourier Series

\[ e^{ix} = \cos x + i\sin x \]

We can rewrite the Fourier series

\[ f(x) = \sum_{n=-\infty}^{\infty} c_n e^{inx} \]

with

\[ c_0 = a_0^2 \]

\[ c_n = a_n - ib_n \]

\[ c_{-n} = a_n + ib_n \]

for \( n > 0 \)

\[ c_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) e^{-inx} \, dx \]
Complex Fourier Series

using Euler's formula

\[ e^{ix} = \cos x + \sin x \]
Complex Fourier Series

using Eulers formula

\[ e^{ix} = \cos x + \sin x \]

we can rewrite the Fourier series

\[ f = \sum_{n=-\infty}^{\infty} c_n e^{inx} \]

with

\[ c_0 = \frac{a_0}{2} \]

\[ c_n = \frac{a_n - ib_n}{2} \quad \text{for } n > 0 \]

\[ c_{-n} = \frac{a_n + ib_n}{2} \quad \text{for } n > 0 \]
Complex Fourier Series
using Euler’s formula

\[ e^{ix} = \cos x + \sin x \]

we can rewrite the Fourier series

\[ f = \sum_{n=-\infty}^{\infty} c_n e^{inx} \]

with

\[ c_0 = \frac{a_0}{2} \]

\[ c_n = \frac{a_n - ib_n}{2} \text{ for } n > 0 \]

\[ c_{-n} = \frac{a_n + ib_n}{2} \text{ for } n > 0 \]

respectively

\[ c_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) e^{-inx} \]
Complex Fourier Series $\Rightarrow$ Fourier Transform

\[ c_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) e^{-ikx} \, dx, \]

\[ f(x) = \sum_{k=-\infty}^{\infty} c_k e^{ikx}, \]

with a limiting process (extending the interval $[-\pi, \pi]$ to $[-\infty, \infty]$)

Definition (Fourier Transform)

\[ c(n) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(x) e^{-inx} \, dx, \]

\[ f(x) = \int_{-\infty}^{\infty} c(n) e^{inx} \, dn, \]
Complex Fourier Series $\Rightarrow$ Fourier Transform

we recall

$$c_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) e^{-ikx} , \quad f = \sum_{k=-\infty}^{\infty} c_k e^{ikx}$$

with a limiting process (extending the interval $[-\pi, \pi]$ to $[-\infty, \infty]$) we get

Definition (Fourier Transform)

$$c(n) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(x) e^{-inx} \, dx , \quad f(x) = \int_{n=-\infty}^{\infty} c(n) e^{inx} \, dn$$
Fourier Transform

Fourier transform for one and two variables (slightly changed notation):

\[ \hat{f}(k) := \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-ikx} \, dx \]

\[ \hat{f}(k, s) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-i(xy + ks)} \, dx \, dy \]

The formulae for the inverse Fourier transform look similar, we have just to interchange \( f \) and \( \hat{f} \) and to cancel the "−" from the exponent of the exponential function.
Fourier Transform

Fourier transform for one and two variables (slightly changed notation):

\[ \hat{f}(k) := \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-ikx} \, dx \]

\[ \hat{f}(k, s) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-i \left( x \cdot (k_s) \right)} \, dx \, dy \]
Fourier Transform

Fourier transform for one and two variables (slightly changed notation):

\[ \hat{f}(k) := \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-ikx} \, dx \]

\[ \hat{f}(k, s) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-i\left(\frac{x}{y}\right) \cdot \left(\frac{k}{s}\right)} \, dx \, dy \]

The formulae for the inverse Fourier transform look similar, we have just to interchange \( f \) and \( \hat{f} \) and to cancel the "−" from the exponent of the exponential function.
Fourier Transform, Basic Properties
Fourier Transform, Basic Properties

Theorem

linear \( \lambda f + g \mapsto \lambda \hat{f} + \hat{g} \)
Fourier Transform, Basic Properties

Theorem

- **Linear**: \( \lambda f + g \mapsto \lambda \hat{f} + \hat{g} \)
- **Translation**: \( f(x + \alpha) \mapsto \hat{f}(k)e^{ik\alpha} \)
Fourier Transform, Basic Properties

Theorem

linear $\lambda f + g \mapsto \lambda \hat{f} + \hat{g}$

translation $f(x + \alpha) \mapsto \hat{f}(k)e^{ik\alpha}$

convolution $f \ast g \mapsto \hat{f} \hat{g}$
Fourier Transform, Basic Properties

Theorem

- **linear** \( \lambda f + g \mapsto \lambda \hat{f} + \hat{g} \)
- **translation** \( f(x + \alpha) \mapsto \hat{f}(k)e^{ik\alpha} \)
- **convolution** \( f \ast g \mapsto \hat{f} \hat{g} \)
- **differentiation** \( \frac{d^n}{(dx)^n} f \mapsto (ik)^n \hat{f}(k) \)
Fourier Transform, Basic Properties

**Theorem**

- **Linear** \( \lambda f + g \mapsto \lambda \hat{f} + \hat{g} \)
- **Translation** \( f(x + \alpha) \mapsto \hat{f}(k) e^{ik\alpha} \)
- **Convolution** \( f \ast g \mapsto \hat{f} \hat{g} \)
- **Differentiation** \( \frac{d^n}{(dx)^n} f \mapsto (ik)^n \hat{f}(k) \)
- **Eigenfunctions** \( e^{-ax^2} \mapsto \frac{1}{\sqrt{2a}} e^{-k^2/(4a)} \)
Theorem

- **linear** $\lambda f + g \mapsto \lambda \hat{f} + \hat{g}$
- **translation** $f(x + \alpha) \mapsto \hat{f}(k)e^{ik\alpha}$
- **convolution** $f \ast g \mapsto \hat{f} \hat{g}$
- **differentiation** $\frac{d^n}{(dx)^n} f \mapsto (ik)^n \hat{f}(k)$
- **eigenfunctions** $e^{-ax^2} \mapsto \frac{1}{\sqrt{2a}} e^{-k^2/(4a)}$
- **isometry** $(f, g) = (\hat{f}, \hat{g})$
Discrete Fourier Transform (DFT)

The MATLAB implementation of the Fourier transform for one variable looks more like a complex Fourier series:

$$X(k) := \sum_{j=1}^{N} x(j) e^{-2\pi i N (j-1)(k-1)}$$

The first element of the $X$ vector is the sum of all elements $X(1) = \sum_{j=1}^{N} x(j) e^{-2\pi i N (j-1)(1-1)} = e^{0} = 1 = \sum_{j=1}^{N} x(j)$.
Discrete Fourier Transform (DFT)

The MATLAB implementation of the Fourier transform for one variable looks more like a complex Fourier series:

\[ X(k) := \sum_{j=1}^{N} x(j) e^{-\frac{2\pi i}{N} (j-1)(k-1)} \]
Discrete Fourier Transform (DFT)

The MATLAB implementation of the Fourier transform for one variable looks more like a complex Fourier series:

$$X(k) := \sum_{j=1}^{N} x(j) e^{-\frac{2\pi i}{N} (j-1)(k-1)}$$

The first element of the $X$ vector is the sum of all elements

$$X(1) = \sum_{j=1}^{N} x(j) e^{-\frac{2\pi i}{N} (j-1)(1-1)} = \sum_{j=1}^{N} x(j)$$
DFT of $\sin(k \cdot x)$ in MATLAB

To find the DFT of a sine wave in MATLAB, we have

```matlab
>> x=(0:127)/128*2*pi;
>> fft(sin(x))
   0  -64i  0 ...  0  64i
>> fft(sin(5*x))
   0  0  0  0  0  -64i  0 ... 
   0  64i  0  0  0  0
```
DFT of \( \sin(k \times x) \) in MATLAB

To find the DFT of a sine wave in MATLAB, we have

\[
\begin{align*}
&\text{>> } x = (0:127)/128*2*\pi; \\
&\text{>> } \text{fft} (\sin(x)) \\
&\quad \quad \quad \quad 0 \quad -64i \quad 0 \ldots \quad 0 \quad 64i \\
&\text{>> } \text{fft} (\sin(5 \times x)) \\
&\quad \quad \quad \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad -64i \quad 0 \ldots \\
&\quad \quad \quad \quad 0 \quad 64i \quad 0 \quad 0 \quad 0 \quad 0
\end{align*}
\]

With the time difference \( \tau \) between \( i \) and \( i + 1 \), the periodic time (time from 0 to \( N \)) is \( N \tau \), therefore the frequency of the \( k^{th} \) peak in the DFT is

\[
f_k = \frac{(k - 1)}{N \tau}
\]
DFT of $\sin(kx)$ in MATLAB

If we use

$$\sin(x) = \frac{e^{ix} - e^{-ix}}{2i}$$

for $x = \frac{2\pi}{N}(j - 1)$, $j = 1, \ldots N$ the DFT of $\sin(nx)$ is

$$X(k) = \sum_{j=1}^{N} x(j) e^{-\frac{2\pi i}{N}(j-1)(k-1)}$$

$$= \sum_{j=1}^{N} e^{n\frac{2\pi i}{N}(j-1)} - e^{-n\frac{2\pi i}{N}(j-1)} e^{-\frac{2\pi i}{N}(j-1)(k-1)}$$

$$= \frac{1}{2i} \left( \sum_{j=1}^{N} e^{\frac{2\pi i}{N}(j-1)(n-k+1)} - \sum_{j=1}^{N} e^{\frac{2\pi i}{N}(j-1)(-n-k+1)} \right)$$

$$= \frac{1}{2i} \left( \sum_{j=0}^{N-1} e^{\frac{2\pi i}{N}j(n-k+1)} - \sum_{j=0}^{N-1} e^{\frac{2\pi i}{N}j(-n-k+1)} \right)$$
DFT of $\sin(kx)$ in MATLAB

With the formula for the geometric series, we have

$$\sum_{j=0}^{N-1} e^{\frac{2\pi i}{N}(n-k+1)j} = \begin{cases} \frac{(e^{\frac{2\pi i}{N}(n-k+1)})^N - 1}{e^{\frac{2\pi i}{N}(n-k+1)} - 1} = 0, & \text{if } n - k + 1 \neq 0 \\ N, & \text{else, i.e. if } k = n + 1 \end{cases}$$

and

$$\sum_{j=0}^{N-1} e^{\frac{2\pi i}{N}(-n-k+1)j} = \begin{cases} \frac{(e^{\frac{2\pi i}{N}(-n-k+1)})^N - 1}{e^{\frac{2\pi i}{N}(-n-k+1)} - 1} = 0, & \text{if } N - n - k + 1 \neq 0 \\ N, & \text{if } k = N - n + 1 \end{cases}$$

E.g. for $\sin(7x)$ and $N = 128$, the DFT has $X(8) = -64i$, $X(122) = 64i$ and all the other $X(k)$ are zero. For $\cos(7x)$ we have $X(8) = X(122) = 64$ and all the other $X(k)$ are zero.
Applications

We present two applications

- Filtering of a noisy signal
- Determination of the main frequencies contained in a signal

The examples were implemented in MATLAB.
Applications

We present two applications

- filtering of a noisy signal
Applications

We present two applications

- filtering of a noisy signal
- determination of the main frequencies contained in a signal
Applications

We present two applications

- filtering of a noisy signal
- determination of the main frequencies contained in a signal
We present two applications

- filtering of a noisy signal
- determination of the main frequencies contained in a signal

The examples were implemented in MATLAB
We start with a mixture of two sinus signals, of different frequency and amplitude:

\[
\begin{align*}
\text{x} &= (0:127)/128*4\pi; \\
\text{s} &= \sin(2\times x) + 0.8\times \sin(15\times x);
\end{align*}
\]
Low Pass Filtering - Noise Suppression

We start with a mixture of two sinus signals, of different frequency and amplitude:

```matlab
>> x=(0:127)/128*4*pi;
>> s = sin(2*x)+0.8*sin(15*x);
>> plot(x,s)
```
Low Pass Filtering - Noise Suppression

Then we compute the Fourier Transform

\[
\text{fs} = \text{fft}(s);
\]

\[
\text{plot}(\text{abs}(s))
\]

The outer peaks are the low frequencies, the inner ones are the noise.
Low Pass Filtering - Noise Suppression

Then we compute the Fourier Transform

>> fs=fft(s);
>> plot(abs(s))
Low Pass Filtering - Noise Suppression

Then we compute the Fourier Transform

```matlab
>> fs=fft(s);
>> plot(abs(s))
```

The outer peaks are the low frequencies, the inner ones are the noise.
Low Pass Filtering - Noise Suppression

Now we delete the peaks in the Fourier transform that were caused by noise. Then we apply the inverse Fourier transform and plot both the original together with the cleaned signal.

\begin{verbatim}
>> fs_clean=fs;
>> fs_clean[20:110]=0;
>> s_clean=real(ifft(fs_clean));
>> plot(x,s,x,s_clean)
\end{verbatim}
Low Pass Filtering - Noise Suppression

Now we delete the peaks in the Fourier transform that were caused by noise. Then we apply the inverse Fourier transform and plot both the original together with the cleaned signal.

```matlab
>> fs_clean=fs;
>> fs_clean[20:110]=0;
>> s_clean=real(ifft(fs_clean));
>> plot(x,s,x,s_clean)
```
Heart- and Breathing Frequency using the DFT

During heart surgery (TECAB) a video sequence of the pericardium was made with an endoscope. Our idea is:
1. compare the first frame with the second, third, fourth, etc.
2. in-phase image frames are similar $\rightarrow$ find prominent frequencies using the Fourier transform.
Heart- and Breathing Frequency using the DFT

During heart surgery (TECAB) a video sequence of the pericardium was made with an endoscope.
Heart- and Breathing Frequency using the DFT

During heart surgery (TECAB) a video sequence of the pericardium was made with an endoscope.

Our idea is
Heart- and Breathing Frequency using the DFT

During heart surgery (TECAB) a video sequence of the pericardium was made with an endoscope.

Our idea is

1. compare the first frame with the second, third, fourth, etc using an image comparison measure
Heart- and Breathing Frequency using the DFT

During heart surgery (TECAB) a video sequence of the pericardium was made with an endoscope.

Our idea is

1. compare the first frame with the second, third, fourth, etc using an image comparison measure
2. in-phase image frames are similar → find prominent frequencies using the Fourier transform.
Loading the Video Frames

We load 750 frames of size 768x576 of the video sequence:

```matlab
frames = zeros(576,768,750);
for k=1:750
    filename = './slices/b-
    fn = sprintf('%s%05d.jpg',filename, k);
    im=double(imread(fn));
    frames(:,:,k)= 0.3*im(:,:,1)+0.59*im(:,:,2)+0.11*im(:,:,3);
end
```

because of limited memory we converted the colour images to greyscale (luminance) via

frames(:,:,k)=0.3*...
Loading the Video Frames

We load 750 frames of size 768x576 of the video sequence:

```matlab
>> frames = zeros(576,768,750);
>> for k=1:750
    filename = './slices/b-';
    fn = sprintf('%s%05d.jpg',filename, k);
    im=double(imread(fn));
    frames(:,:,k)= 0.3*im(:,:,1)+0.59*im(:,:,2)+0.11*im(:,:,3);
end
```

because of limited memory we converted the colour images to greyscale (luminance) via

```
frames(:,:,k)=0.3*...
```

(768*576*750
doubles are about 2.4 GB on a 64 bit machine)
Loading the Video Frames

We load 750 frames of size 768x576 of the video sequence:

```matlab
>> frames = zeros(576,768,750);
>> for k=1:750
filename = './slices/b-';
fn = sprintf('%s%05d.jpg',filename, k);
im=double(imread(fn));
frames(:,:,k)= 0.3*im(:,:,1)+0.59*im(:,:,2)+0.11*im(:,:,3);
end
```

because of limited memory we converted the colour images to greyscale (luminance) via

```
frames(:, :, k)= 0.3*im(:, :, 1)+0.59*im(:, :, 2)+0.11*im(:, :, 3);
```

(768*576*750 doubles are about 2.4 GB on a 64 bit machine)
Frame Comparison Function

To compare the frames we use summed squared differences:

```matlab
function res = ssd(i1, i2)
res = sum(sum((i1 - i2).^2));
end
```

```matlab
ssdall = [];
for k = 1:750
    ssdall = [ssdall, ssd(frames(:,:,1), frames(:,:,k))];
end
```
Frame Comparison Function

To compare the frames we use summed squared differences:

```matlab
function res = ssd( i1, i2 )
    res = sum(sum((i1 - i2).^2));
```

and compute the difference from frame one to all the others:

```matlab
>> ssdall = [];
>> for k = 1:750;
     ssdall = [ssdall ssd(frames(:,:,1),frames(:,:,k))];
>> end
```
Frame Comparison Function

To compare the frames we use summed squared differences:

function res = ssd( i1, i2 )
res = sum(sum((i1-i2).^2));

and compute the difference from frame one to all the others:
Frame Comparison Function

To compare the frames we use summed squared differences:

```matlab
function res = ssd( i1, i2 )
    res = sum(sum((i1-i2).^2));
```

and compute the difference from frame one to all the others:

```matlab
>> ssdall=[];
>> for k=1:750;
    ssdall = [ssdall ssd(frames(:,:,1),frames(:,:,k))];
end
```
Frame Comparison Function
Frame Comparison Function

The difference of frame one to the first frames is artificially small, we use only frames 13-750
Frame Comparison Function

The difference of frame one to the first frames is artificially small, we use only frames 13-750

>> ssdsmall=ssdall(13:750);
>> plot(sdssmall)
Derive the DFT

The first component of the FFT is the sum of all elements, to avoid this we subtract the mean value.

We have 738 frames and 25 frames per second, $\tau = \frac{1}{25} = 0.04$, the frequency of the $k$-th peak is $f_k = k - 1 \frac{N}{\tau} = k - 1.2952$ Hz $\approx 2 \ast (k - 1)$ beats per minute.

```matlab
>> ssdsmall = ssdall(13:750);
>> fssd = fft(ssdsmall - mean(ssdsmall));
>> x = (0:40)./(738*0.04)/60;
>> plot(x,abs(fssd(1:41)))
```

0 20 40 60 80
0 2 4 6 8 x 10
9
events per minute

32 / 36
Derive the DFT

The first component of the FFT is the sum of all elements, to avoid this we subtract the mean value.

\[ \tau = \frac{1}{25} = 0.04, \]

the frequency of the \( k \)-th peak is

\[ f_k = \frac{k - 1}{N \tau} = \frac{k - 1}{29.52} \text{ Hz} \approx 2 \times (k - 1) \text{ beats per minute} \]

\[
\text{ssdsmall} = \text{ssdall}(13:750);
\]
\[
\text{fssd} = \text{fft}(	ext{ssdsmall} - \text{mean}(	ext{ssdsmall}));
\]
\[
\text{x} = \left(0:40\right) / \left(738 \times 0.04\right) / 60;
\]
\[
\text{plot}(\text{x}, \text{abs}(\text{fssd}(1:41)))
\]

0 20 40 60 80
0 2 4 6 8 x 10^9
events per minute

\[
0 20 40 60 80
\]

32 / 36
Derive the DFT

The first component of the FFT is the sum of all elements, to avoid this we subtract the mean value.
We have 738 frames and 25 frames per second, \( \tau = 1/25 = 0.04 \), the frequency of the k-th peak is
\[
f_k = \frac{k-1}{N\tau} = \frac{k-1}{29.52} \text{ Hz} \approx 2 \times (k - 1) \text{ beats per minute}
\]
Derive the DFT

The first component of the FFT is the sum of all elements, to avoid this we subtract the mean value.
We have 738 frames and 25 frames per second, \( \tau = \frac{1}{25} = 0.04 \),
the frequency of the \( k \)-th peak is
\[
f_k = \frac{k-1}{N\tau} = \frac{k-1}{29.52} \text{ Hz} \approx 2 \times (k - 1) \text{ beats per minute}
\]

```matlab
>> ssdsmall=ssdall(13:750);
>> fssd=fft(ssdsmall-mean(ssdsmall));
>> x=(0:40)./((738*0.04)/60);
>> plot(x,abs(fssd(1:41)))
```
Comparison with ECG

The first peak of the FFT is the respiration, the second the heart motion. The peaks are at $k = 7$, which is 12 bpm and at $k = 27$, which is 52 bpm.
Comparison with ECG

The first peak of the FFT is the respiration, the second the heart motion.
Comparison with ECG

The first peak of the FFT is the respiration, the second the heart motion.
The peaks are at $k = 7$, which is 12 bpm and at $k = 27$, which is 52 bpm.
Unfortunately not even the sun is free from ugly little spots. A regular variation in the number of sunspots was described by Samuel Schwabe in 1843. Rudolf Wolf collected sunspot activity data and calculated a period of 11.1 years. To compare the data he defined the *Wolf'sche Relativzahl*, also called sunspot number.
Unfortunately not even the sun is free from ugly little spots.
Unfortunately not even the sun is free from ugly little spots. A regular variation in the number of sunspots was described by Samuel Schwabe in 1843.
Unfortunately not even the sun is free from ugly little spots. A regular variation in the number of sunspots was described by Samuel Schwabe in 1843.
Sunspots Period

Unfortunately not even the sun is free from ugly little spots. A regular variation in the number of sunspots was described by Samuel Schwabe in 1843.

Rudolf Wolf collected sunspot activity data and calculated a period of 11.1 years. To compare the data he defined the *Wolf’sche Relativzahl*, also called *sunspot number*. 
The sunspot number is calculated by first counting the number of sunspot groups $g$ and then the number of individual sunspots $s$. Then $SSN = 10g + s$.

Since most sunspot groups have, on average, about ten spots, this formula for counting sunspots gives reliable numbers even when the observing conditions are less than ideal and small spots are hard to see.

Source: solarscience.msfc.nasa.gov/SunspotCycle.shtml
Sunspot Number

The *sunspot number* is calculated by first counting the number of sunspot groups $g$ and then the number of individual sunspots $s$.
Sunspot Number

The *sunspot number* is calculated by first counting the number of sunspot groups $g$ and then the number of individual sunspots $s$. Then $SSN = 10 \cdot g + s$
The *sunspot number* is calculated by first counting the number of sunspot groups $g$ and then the number of individual sunspots $s$. Then $SSN = 10 \cdot g + s$
The *sunspot number* is calculated by first counting the number of sunspot groups \( g \) and then the number of individual sunspots \( s \). Then \( SSN = 10 \cdot g + s \)

Since most sunspot groups have, on average, about ten spots, this formula for counting sunspots gives reliable numbers even when the observing conditions are less than ideal and small spots are hard to see.
Sunspot Number

The *sunspot number* is calculated by first counting the number of sunspot groups $g$ and then the number of individual sunspots $s$. Then $SSN = 10 \cdot g + s$

Since most sunspot groups have, on average, about ten spots, this formula for counting sunspots gives reliable numbers even when the observing conditions are less than ideal and small spots are hard to see.

Source solarscience.msfc.nasa.gov/SunspotCycle.shtml
Sunspot Cycle Length

```matlab
fs = abs(fft(ssnn));
plot(fs(1:100));
[C I] = max(fs(1:30));
I = 25;
length(ssn) = 3181;

The maximum is at \( k = 25 \), remember \( f_k = \frac{k - 1}{N \tau} \), \( \tau = \frac{1}{12} \) years are the time unit.

\( f = \frac{25 - 1}{3181 \times 1/12} \)
\( f = 0.0905 \)
\( \frac{1}{f} = 11.0451 \)
```
Sunspot Cycle Length

>> fs=abs(fft(ssnn));
>> plot(fs(1:100))
>> [C I]=max(fs(1:30))

... 
I = 25
>> length(ssn)
 3181
Sunspot Cycle Length

```matlab
>> fs=abs(fft(ssnn));
>> plot(fs(1:100))
>> [C I]=max(fs(1:30))
... 
I = 25
>> length(ssn)
3181
```
Sunspot Cycle Length

>> fs=abs(fft(ssnn));
>> plot(fs(1:100))
>> [C I]=max(fs(1:30))
...
I = 25
>> length(ssn)
3181

The maximum is at \( k = 25 \),
Sunspot Cycle Length

>> fs=abs(fft(ssnn));
>> plot(fs(1:100))
>> [C I]=max(fs(1:30))
...
I = 25
>> length(ssn)
3181

The maximum is at \( k = 25 \), remember \( f_k = \frac{k-1}{N\tau} \), \( \tau = 1/12 \), years are the time unit
Sunspot Cycle Length

>> fs=abs(fft(ssnn));
>> plot(fs(1:100))
>> [C I]=max(fs(1:30))
...
I = 25
>> length(ssn)
3181

The maximum is at $k = 25$, remember $f_k = \frac{k-1}{N \tau}$, $\tau = 1/12$, years are the time unit

>> f=(25-1)/(3181*1/12)
f = 0.0905
>> 1/f
11.0451