

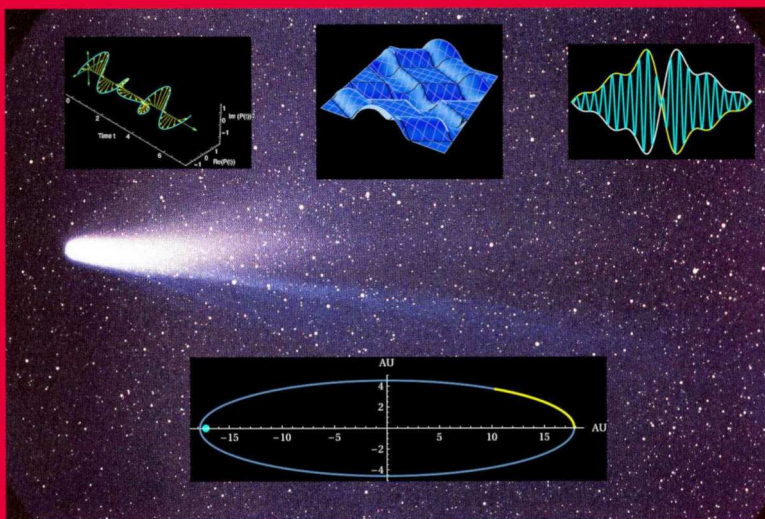
Rolf Brigola

# Fourier Analysis with Mathematica 1

Learning Fourier Series by Examples

2nd extended edition

**Fourier Series, Wave and Heat Equations  
DFT, DCT, Aliasing, Leakage  
Interpolation, JPEG, Huffman Code  
Windowed Fourier Transforms  
Spectrograms, OFDM**



Rolf Brigola

# **Fourier Analysis with Mathematica 1**

Learning Fourier Series by Examples

(2nd extended edition)

Prof. Dr. Rolf Brigola  
Technische Hochschule Nürnberg Georg-Simon-Ohm  
Fakultät Angewandte Mathematik, Physik und Allgemeinwissenschaften  
Keßlerplatz 12, 90489 Nürnberg

© Rolf Brigola, 2025  
All rights reserved

Imprint: Independently published

Mathematics Subject Classification (2010): 42-01

The graphics on the cover shows Halley's Comet, its orbit as calculated in the text, and in yellow the distance traveled within 2 years starting from aphelion. The picture is from NASA ID LSPN-1725.

Top left is a trigonometric polynomial illustrated as a circular wave, in the center a solution of the 1D wave equation for a finite string with a smooth initial condition and homogeneous boundary values, and top right an amplitude modulation of a trigonometric polynomial.

## Preface

Fourier series are fundamental mathematical tools for describing and solving a wide array of technical and scientific problems. These include areas of physics, mechanical engineering, electrical engineering, and signal and control theory. This text is intended for readers seeking to understand the theoretical foundation of Fourier analysis methods, and their practical applications.

Students of mathematics or engineering learn the basics on Fourier series representations of periodic functions in their early math courses. Nowadays, Fourier series build the mathematical foundation of a myriad of applications that largely permeate our everyday lives, from basic electrical engineering, medicine and chemistry to modern signal processing in communication devices and computational mathematics in any scientific application areas.

The text incorporates insights from many years of lectures delivered to students of applied mathematics, physics, electrical engineering, and communications engineering at the Technische Hochschule Nürnberg Georg Simon Ohm, starting in their second semester.

This booklet has its focus on showing how theoretical results on Fourier series can be used in applications and examples with the help of the computer algebra system *Mathematica*. Thus, no proofs of used theorems and properties of Fourier series are given in the text. Instead, it is demonstrated how to comprehend properties and applications of Fourier series with the help of *Mathematica*, which frees us from time-consuming own calculations and gives us the opportunity to understand the subject matter with clear illustrations.

In the present 2nd edition of this text, I have corrected a few typos, have (hopefully) improved some explanations and have added a new section on a Ritz-Galerkin solution for a Dirichlet boundary value problem on a rectangle. Due to the simple shape of the region, a Ritz-Galerkin solution with Fourier series is possible. The principle is already a preparation for solutions on more complicated regions by the

Finite Element method (FEM). This shall be a topic with regard to distribution theory in a subsequent booklet, which is in progress. The extended theory of Fourier transforms has its main success in connection with distribution theory. Typical applications with the help of *Mathematica* will be treated in a future booklet on Fourier Analysis with Mathematica 2.

For the theory with proofs of the theorems and detailed examples, exercises and their solutions it is referred to the authors textbook

### [Fourier Analysis with Distributions](#)

A First Course with Applications

Springer, Text in Applied Mathematics 79, 2025

Rolf Brigola

Nürnberg, Germany, 2025

# Contents

<b>1</b>	<b>Basics on Fourier Series with Mathematica</b>	<b>1</b>
1.1	Dirichlet and Fejér Kernels	3
	Dirichlet Kernels	
	The Fourier Series of the Sawtooth	
	Fejér kernels, Smoothing, Gibbs Phenomenon	
1.2	Properties of Fourier Series	9
	Translations of a Signal	
	Amplitude Modulation, Translation of the Spectrum	
	Smoothness and Magnitude of the Spectrum	
	Uniform Convergence of Fourier Series	
	Fourier Series of Derivatives and Integrals	
1.3	Orthogonal Projections into Finite-Dimensional Subspaces of $L^2([a, b])$	19
	Approximation by Legendre polynomials	
1.4	Convergent Trigonometric Series, which are not Classical Fourier Series	24
1.5	Graphical Illustrations of Trigonometric Polynomials	25
<b>2</b>	<b>Application of Fourier Series to Linear Differential Equations</b>	<b>29</b>
2.1	Ordinary Linear Differential Equations with Constant Coefficients	29
	Forced periodic oscillations of a mass on a spring	
2.2	Fourier series in homogeneous 1D Heat Equations	34
2.3	Fourier Series in inhomogeneous 1D Heat Equations	42
2.4	Fourier Series Solution for the Potential Equation on a Circular Disk	45
2.5	Initial Boundary Value Problem for a Force-Free Vibrating String	47
2.6	Solution of a Kepler Equation by Fourier Series Expansion	50
2.7	Solving a 2D Poisson Equation for a Rectangular Membrane by a Ritz-Galerkin Solution	53

<b>3</b>	<b>Discrete Fourier Transforms</b>	<b>61</b>
3.1	Fundamentals on the DFT	61
	DFT and Frequency Assignment, Handling of Alias Effects	
	Subsampling in digital transmission systems	
	Delayed Sampling	
	Correction in the Spectrum of Trigonometric Polynomials	
3.2	Estimation of Signal Spectra, Aliasing	70
3.3	Leakage, Time Windows	74
3.4	Inverse Discrete Fourier Transform IDFT, Interpolation	78
3.5	DFT, IDFT and Time Windows in Digital Signal Processing	83
	WLAN Transmission, Spectrograms	
3.6	The Discrete Cosine Transforms DCT I and DCT II	94
3.7	Interpolation with the DCT I and DCT II	96
3.8	Numerical Integration, Clenshaw-Curtis Quadrature	99
3.9	The DCT in Interpolation with Chebyshev Polynomials	102
	Chebyshev Polynomials of the first kind	
	Chebyshev Polynomials as an orthogonal system	
	Comparison with interpolation by Legendre polynomials	
	Connection with DCT I and DCT II, Example of C. Runge	
	The Alias Effect with Chebyshev Polynomials	
	An Extremal Property of the Chebyshev Polynomials	
<b>4</b>	<b>The DCT 2D, JPEG, Huffman Code</b>	<b>117</b>
4.1	DCT 2D, JPEG	117
4.2	Huffman Coding	125

Rolf Brigola

## **Fourier Analysis with Mathematica 1** **Learning Fourier Series by Examples**

### **1 Basics on Fourier Series with Mathematica**

Fourier analysis dates back to the ideas of Daniel Bernoulli (1700-1782) and French mathematician Jean-Baptiste Joseph Fourier (1768-1830).

Historically, it started with the representation of solutions for heat and wave equations as superpositions of trigonometric functions. Students of mathematics or engineering learn the basics on Fourier series representations of periodic functions in their early math courses. In more advanced studies, Fourier analysis is developed to describe and solve a wide variety of problems in mathematics, natural sciences and engineering. Nowadays, it builds the mathematical foundation of a myriad of applications that largely permeate our everyday lives, from basic electrical engineering, medicine and chemistry to modern signal processing in communication devices and computational mathematics in any scientific application areas.

The presented text here and in subsequent parts is tailored to show how certain types of linear problems can be solved with the help of Fourier Analysis and its application with Mathematica. For the theoretical background, the definitions, the theorems and their proofs I refer to my textbook as [\[1\]](#)

[Rolf Brigola \(2025\) Fourier Analysis and Distributions. A First Course with Applications. Springer's TAM series, Vol. 79.](#)

Thus, in a certain sense, this book can be understood as a supplement for the above, showing how one possibly can avoid some boring own calculations, how to solve treated problems and how to generate illustrative graphics with the help of Mathematica or an equivalent computer algebra system.

We start with Fourier series and some of their applications. In subsequent booklets we treat distributions, the Fourier transform and applications like linear filter design, sampling and more. This volume is related to the chapters 1-7 of my above indicated textbook. The aim is to first introduce and remind users of some basics about Fourier series with the help of Mathematica. It will show Mathematica commands that can be used to calculate Fourier series and to generate graphics to illustrate the facts. Typical application examples of Fourier Series are shown, which demonstrate the benefits of Fourier analysis for approximation tasks and signal processing. Subsequently, we will look at the DFT, DCT and Chebyshev polynomials and some of their typical properties that are relevant for applications.



**Author's note:** I am not really an expert on the almost inexhaustible possibilities offered by a really practiced, in-depth use of Mathematica. I have therefore essentially tried to illustrate the mathematical material with this offer and to show how the content covered can be accessed with Mathematica instructions (some of which are probably often too complicated, but also transparent for Mathematica beginners). I trust that readers will become familiar with Mathematica commands and options (especially with the numerous possible graphics options) by inspecting the given examples and using analog versions in own examples.

With regard to many application examples in the text, I often call the variable of Fourier series a “time” parameter. Of course it changes its meaning depending on the specific application examples from physics or other fields. For the mathematics, this makes no difference.

Details on all Mathematica commands and options can easily be found by the online help system of Mathematica. For certain aspects of Mathematica such as programming one can also easily find tutorials and other sources by searching the internet. The contents were all written with Mathematica. The examples can also be treated with corresponding commands in other computer algebra or numerical systems like Maple, Matlab et al.

## 1.1 Dirichlet and Fejér Kernels

Dirichlet kernels and Fejér kernels play a fundamental role in the study of Fourier series. Partial sums of Fourier series of periodic functions are convolutions with Dirichlet kernels (cf. the above mentioned book, chapter 3, 4 and chapter 7).

At first I define some functions. Here I use two functions for calculating Fourier coefficients and Fourier expansions for periodic functions defined on an interval  $[a,b]$  with period  $(b-a)$ . The reason is that Mathematica offers only the respective commands for periodic functions defined on a symmetric interval around zero (see below). One can save own functions in an m-file and load this at the beginning of a notebook to have them available.

Defining of own functions for Fourier coefficients and expansion:

```
In[ ]:= fourcoeff [f_, A_, B_, k_] :=
  1 / (B - A) Integrate [f[t] Exp[-I k t 2 Pi / (B - A)], {t, A, B}]
  (* Calculate the k-th complex Fourier coeff. *)
fourpolynomial [f_, A_, B_, n_] :=
  FullSimplify [Refine [Sum [fourcoeff [f, A, B, k] Exp[I k t 2 Pi / (B - A)], {k, -n, n}],
    {Element [k, Integers]}]]
  (* Calculate the fourier expansion as trig. polynomials up to degree n *)
b[k_, A_, B_] := FullSimplify [I (fourcoeff [f, A, B, k] - fourcoeff [f, A, B, -k])]
  (* Sin-coeff b_k *)
a[k_, A_, B_] := FullSimplify [(fourcoeff [f, A, B, k] + fourcoeff [f, A, B, -k])]
  (* Cos-coeff a_k *)
```

### 1. Dirichlet Kernels

We consider the  $2\pi$  - periodic Dirichlet kernel  $\text{Dir}[t, N]$  of degree  $N \geq 1$  :

```
In[ ]:= Dir[t_, N_] = Simplify [1 + 2 Sum [Cos[k t], {k, 1, N}]]
```

$$\text{Out[ ]} = 1 + 2 \cos \left[ \frac{1}{2} \times (1 + N) t \right] \csc \left[ \frac{t}{2} \right] \sin \left[ \frac{N t}{2} \right]$$

In a trigonometric form

```
In[ ]:= TrigToExp [%]
```

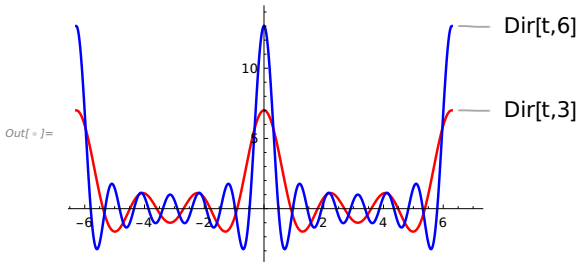
$$\text{Out[ ]} = 1 + \frac{\left( e^{-\frac{1}{2} i N t} - e^{\frac{i N t}{2}} \right) \left( e^{-\frac{1}{2} i (1+N) t} + e^{\frac{1}{2} i (1+N) t} \right)}{e^{-\frac{i t}{2}} - e^{\frac{i t}{2}}}$$

**Example 1.** As illustration the Dirichlet kernels of degrees 3 and 6 are shown. They oscillate more and more with increasing degrees  $N$ . **At no point exists a limit for  $N \rightarrow \infty$ .**

```

In[ ]:= g1 := Plot[Dir[t, 3], {t, -2 Pi, 2 Pi}, PlotStyle -> Directive [
      Red, Thickness [0.006]], PlotRange -> All, PlotLabels -> Style["Dir[t,3]", 14]]
g2 := Plot[Dir[t, 6], {t, -2 Pi, 2 Pi}, PlotStyle -> Directive [
      Blue, Thickness [0.006]], PlotRange -> All, PlotLabels -> Style["Dir[t,6]", 14]]
Show[g1, g2, ImageSize -> Medium ]

```



We observe that a Dirichlet kernel of degree  $N$  has the maximum possible number of  $2N$  zeros per period.

Dirichlet kernels of other periods also have  $2N$  zeros per period.

The mean values of the Dirichlet kernels over one period are one. Here as example:

```

In[ ]:= Integrate [Dir[t, 5], {t, 0, 2 Pi}] / (2 Pi)

```

```

Out[ ]:= 1

```

## 2. The Fourier Series of the Sawtooth

**Example 2.** As first periodic function we consider the  $2\pi$ -periodic sawtooth in the time interval from  $-4\pi$  to  $2\pi$ .

At first the section on  $[0, 2\pi]$  and then a plot over 3 periods. Afterwards the Fourier expansion.

Mathematica offers an implemented procedure for that, but uses as standard  $2\pi$ -periodic functions, defined in the interval  $[-\pi, \pi]$ . Thus, we define the sawtooth as function in that interval or use the above defined functions `fourcoeffc` and `fourserc`.

The command `FourierTrigSeries` gives the partial sum of the Fourier expansion with a number of upper harmonics.

**The Gibbs Phenomenon at the discontinuity points is clearly recognizable in the plot below. The overshoot near the discontinuity is about 9% of the jump height (see [1], 3.2).**

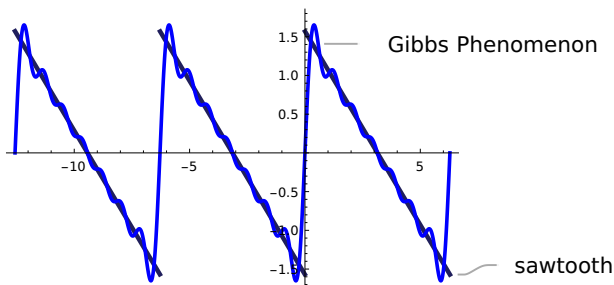
```

In[ ]:= sawtooth[t_] = (Pi - t)/2 (UnitStep[t] - UnitStep[t - 2 Pi])
p1 = Plot[Sum[sawtooth[t - k 2 Pi], {k, -4, 2}], {t, -4 Pi, 2 Pi},
  PlotStyle -> Directive[RGBColor[0.127, 0.121, 0.36], Thickness[0.01]],
  PlotLabels -> Style["sawtooth", 14]];
f[t_] := sawtooth[t + 2 Pi] + sawtooth[t];
FSsawtooth[t_] = FourierTrigSeries[f[t], t, 7]
p2 := Plot[FSsawtooth[t], {t, -4 Pi, 2 Pi},
  PlotStyle -> Directive[Blue, Thickness[0.008]], PlotRange -> All,
  PlotLabels -> Style["Gibbs Phenomenon", 14], ImageSize -> Medium];
p2a = Show[p1, p2, ImageSize -> Medium];
Show[p2a]

```

$$\text{Out}[ ] = \frac{1}{2} (\pi - t) (\text{UnitStep}[t] - \text{UnitStep}[-2\pi + t])$$

$$\text{Out}[ ] = \sin[t] + \frac{1}{2} \sin[2t] + \frac{1}{3} \sin[3t] + \frac{1}{4} \sin[4t] + \frac{1}{5} \sin[5t] + \frac{1}{6} \sin[6t] + \frac{1}{7} \sin[7t]$$



Now its Fourier expansion. Mathematica offers an implemented procedure for that, but uses as standard periodic functions defined in a symmetric interval around zero.

Thus, we define the sawtooth as function in the interval  $[-\pi, \pi]$  or use the above defined functions `fourcoeffc` and `fourserc`. The command `FourierTrigSeries` gives the partial sum of the Fourier expansion with a number of upper harmonics.

**The Gibbs Phenomenon at the discontinuity points is clearly recognizable in the plot.**

The same with the above defined functions integrating from  $-2\pi$  to 0.

```
In[*]:= fourcoeffc [f, -2 Pi, 0, 5]
```

```
b[5, -2 Pi, 0]
```

```
h[t_] = fourpolynomial [f, -2 Pi, 0, 7]
```

$$\text{Out}[*]= -\frac{i}{10}$$

$$\text{Out}[*]= \frac{1}{5}$$

$$\text{Out}[*]= (1 + \cos[t]) \sin[t] + \frac{1}{3} \sin[3t] + \frac{1}{4} \sin[4t] + \frac{1}{5} \sin[5t] + \frac{1}{6} \sin[6t] + \frac{1}{7} \sin[7t]$$

### Convolution is the key to understanding Fourier expansions:

Fourier series are obtained by convolutions of a function  $f$  with Dirichlet kernels, later with Fejér kernels and other summation kernels.

In the last example, you obtain the same partial sum by the  $2\pi$ -periodic convolution of the sawtooth with the  $2\pi$ -periodic Dirichlet kernel of degree 7 :

```
In[*]:= convolution [t_] =
```

```
FullSimplify [1 / (2 Pi) Integrate [sawtooth [s] * Dir[t - s, 7], {s, 0, 2 Pi}]]
```

$$\text{Out}[*]= (1 + \cos[t]) \sin[t] + \frac{1}{3} \sin[3t] + \frac{1}{4} \sin[4t] + \frac{1}{5} \sin[5t] + \frac{1}{6} \sin[6t] + \frac{1}{7} \sin[7t]$$

Since  $\cos[t] \sin[t] = \sin[2t]/2$ , we see that this convolution is again the above partial sum of the sawtooth.

### 3. Fejér kernels, Smoothing, and Vanishing of the Gibbs Phenomenon

The Fejér kernels are the arithmetic means of the Dirichlet kernels. We define the  $2\pi$ -periodic Fejér kernel  $\text{Fej}[t, N]$  of degree  $N-1 \geq 1$  and plot it for degree 9.

**Example 3.** The graphics shows a Dirichlet kernel of degree 6 and a Fejér kernel of degree 9.

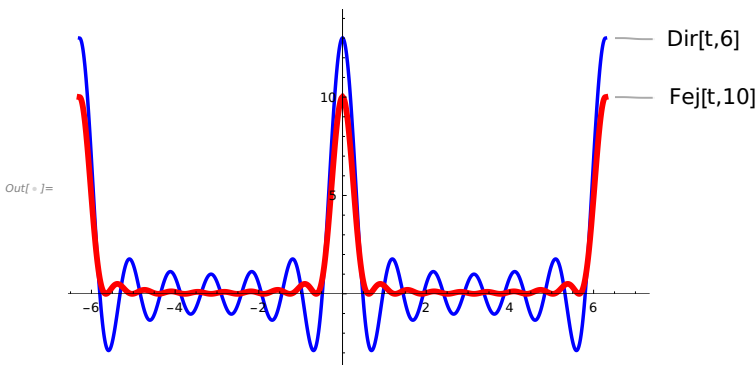
```
In[*]:= Fej[t_, N_] = 1 / N * (1 + Sum[Dir[t, k], {k, 1, N - 1}])
```

```
g3 = Plot[Fej[t, 10], {t, -2 Pi, 2 Pi}, PlotStyle -> Directive[
```

```
Red, Thickness[0.01]], PlotRange -> All, PlotLabels -> Style["Fej[t,10]", 14]];
```

$$\text{Out}[*]= \frac{1 + \frac{1}{2} \left( -\csc\left[\frac{t}{2}\right]^2 \sin\left[\frac{1}{2}(-\pi + 2t)\right] + \csc\left[\frac{t}{2}\right]^2 \sin\left[\frac{1}{2}(-\pi + 2Nt)\right] \right)}{N}$$

```
In[ ]:= plot3a = Show[g2, g3]
```



The Fejér kernels also have mean values one, but unlike the Dirichlet kernels are non-negative. In contrast to the Dirichlet kernels, they converge for growing  $N$  uniformly to zero in every closed interval that does not contain any points of the form  $2k\pi$ . The trigonometric polynomial with degree  $N$  for the Fejér kernel  $\text{Fej}[t, N+1]$  has  $N$  zeros per period. Here the mean value of such a kernel:

```
In[ ]:= Integrate [Fej[t, 5], {t, 0, 2 Pi}]/(2 Pi)
```

```
Out[ ]:= 1
```

Since there are continuous periodic functions whose Fourier series are divergent at infinitely many points, it was an important result of L. Fejér in 1904 that the arithmetic means of the partial sums of the Fourier series of a continuous periodic function  $f$  converge even uniformly towards  $f$ . (For the proof see [1], chapter 7.)

In addition, the Gibbs phenomenon no longer occurs when approximating functions with jump points. Due to the lower weighting of high-frequency components the approximation is smoothed and less wavy. The price for this is a larger error in the quadratic mean in comparison with an approximation using a partial sum of the Fourier series with the same degree.

Equivalent to the formation of such arithmetic means is the periodic convolution with Fejér kernels. We show this using the example of a partial sum of the sawtooth:

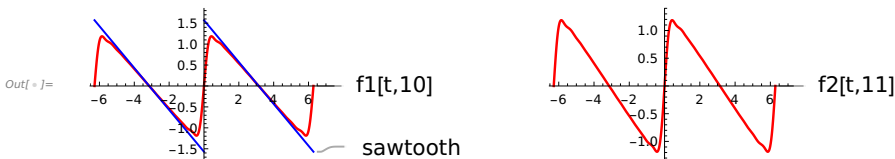
**You can see that there is no Gibbs phenomenon with the approximation by Fejér means.**

**Example 4.** Blue the sawtooth, red the approximation with the Fejér averaging. For comparison at the right the plot of the periodic convolution of the sawtooth with our Fejér kernel  $\text{Fej}[t,11]$  (The calculation takes some time, i.e., some patience is necessary. This calculation is of course too complicated from a practical point of view if, as above, the resulting trigonometric polynomial can be specified directly.)

```

In[ ]:= f1[t_, N_] := Sum[(1 - k / (N + 1)) Sin[k t] / k, {k, 1, N}]
g4 := Plot[f1[t, 10], {t, -2 Pi, 2 Pi}, PlotStyle -> Directive[
    Red, Thickness[0.01]], PlotRange -> All, PlotLabels -> Style["f1[t,10]", 14]]
g5 := Plot[sawtooth[t + 2 Pi] + sawtooth[t], {t, -2 Pi, 2 Pi},
    PlotStyle -> Directive[Blue, Thickness[0.008]],
    PlotRange -> All, PlotLabels -> Style["sawtooth ", 14]]
g5a = Show[g4, g5];
f2[t_, N_] := NIntegrate[sawtooth[s] × Fej[t - s, N], {s, 0, 2 Pi}] / (2 Pi)
g6 = Plot[f2[t, 11], {t, -2 Pi, 2 Pi}, PlotStyle -> Directive[
    Red, Thickness[0.01]], PlotRange -> All, PlotLabels -> Style["f2[t,11]", 14]];
GraphicsRow[{g5a, g6}]

```

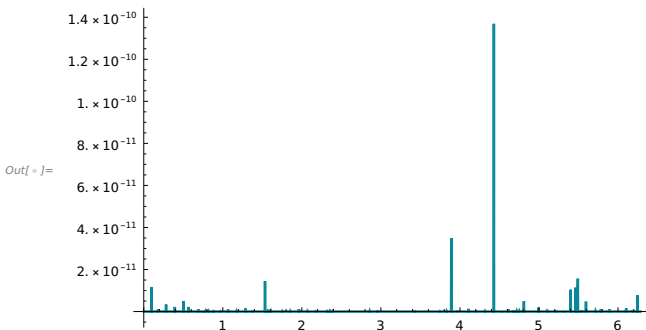


Let's check the differences between both approximations. The deviations are due to the numerical integration in the convolution integral.

```

In[ ]:= Plot[Abs[f1[t, 4] - f2[t, 5]], {t, 0, 2 Pi}, PlotRange -> All, ImageSize -> Small]

```



## 1.2 Properties of Fourier Series

The following are examples of how the Fourier coefficients change (or not), when a periodic function is shifted, mirrored or its amplitudes are modulated.

### Integration Interval, Time Reversal

For T-periodic functions, it is clear that the integration interval for calculating the Fourier coefficients can be shifted and only depends on the period duration. It is also clear that even periodic functions have a cos series, odd ones a sine series. With time reversal from  $f(t)$  to the function  $f(-t)$ , the Fourier coefficients  $c_k$  are transformed to  $c_{-k}$  (substitution rule for integrals) and with complex conjugation of  $f$  to  $c_{-k}$ .

### Similarity

The function  $f(at)$  similar to a periodic function  $f(t)$ ,  $a > 0$ , has the same Fourier coefficients. However, their frequency assignment changes. As an example, we consider the sawtooth( $t$ ) and the scaled similar function  $f(t)=\text{sawtooth}(2t)$ , their Fourier series expansions up to a certain degree and their Fourier coefficients.

**Example 5.** We see the same Fourier coefficients/amplitudes, but assigned to double the frequencies compared to the original sawtooth.

To put it graphically: The signal  $f$  runs twice as fast, but otherwise looks the same, mathematically "similar".

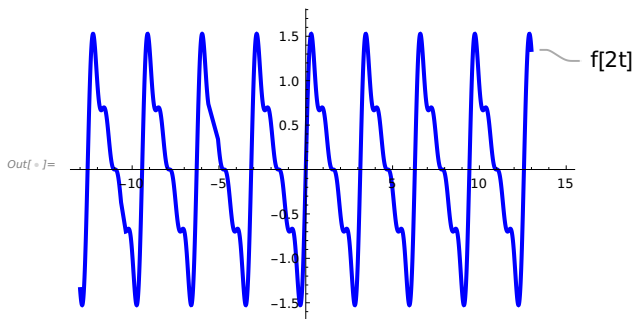
```
In[ ]:= FourierTrigSeries [f[2 t], t, 8]
```

```
Plot[ Sin[2 t] +  $\frac{1}{2}$  Sin[4 t] +  $\frac{1}{3}$  Sin[6 t] +  $\frac{1}{4}$  Sin[8 t], {t, -13, 13},
```

```
PlotStyle → Directive [Blue, Thickness [0.008]],
```

```
PlotRange → All, ImageSize → Small, PlotLabels → Style["f[2t]", 14]]
```

```
Out[ ]:= Sin[2 t] +  $\frac{1}{2}$  Sin[4 t] +  $\frac{1}{3}$  Sin[6 t] +  $\frac{1}{4}$  Sin[8 t]
```





## Translations of a Signal

A "time shift" of a  $T$ -periodic signal does not change its amplitudes, but its phases. For  $f(t+t_0)$  each Fourier coefficient is multiplied by  $\text{Exp}(ik\omega_0 t_0)$  for the phase shift ( $\omega_0 = 2\pi / T$ ).

**Example 6.** To illustrate this, we shift the  $2\pi$ -periodic sawtooth by  $t_0 = \pi$  and look at the Fourier coefficients: They are then each multiplied by  $(-1)^k$  in comparison with the unshifted function, resulting in an alternating series.

```

In[ ]:= ftransl[t_] = sawtooth[t + Pi]

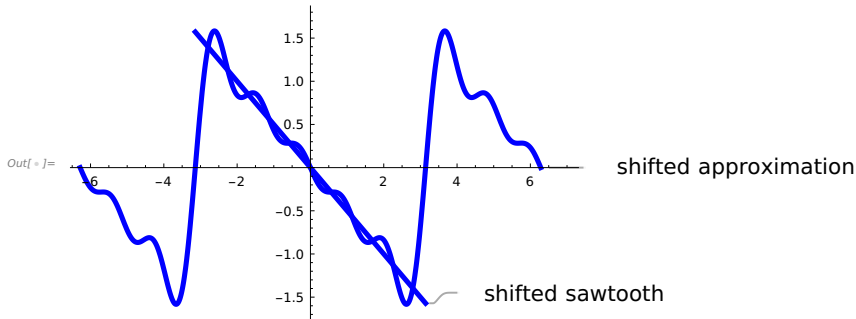
g7 = Plot[-1/2 t (-UnitStep[-Pi + t] + UnitStep[Pi + t]), {t, -Pi, Pi},
PlotStyle -> Directive[Blue, Thickness[0.01]],
PlotRange -> All, PlotLabels -> Style["shifted sawtooth", 14]];

trigpol5[t_] = FourierTrigSeries[ftransl[t], t, 5]
g8 = Plot[trigpol5[t], {t, -2 Pi, 2 Pi},
PlotStyle -> Directive[Blue, Thickness[0.01]],
PlotRange -> All, PlotLabels -> Style["shifted approximation", 14]];
Show[
  g7,
  g8]

```

$$\text{Out[ ]} = -\frac{1}{2} t (-\text{UnitStep}[-\pi + t] + \text{UnitStep}[\pi + t])$$

$$\text{Out[ ]} = -\sin[t] + \frac{1}{2} \sin[2t] - \frac{1}{3} \sin[3t] + \frac{1}{4} \sin[4t] - \frac{1}{5} \sin[5t]$$



## Amplitude Modulation, Translation of the Spectrum

Amplitude modulation causes a translation of the spectrum.

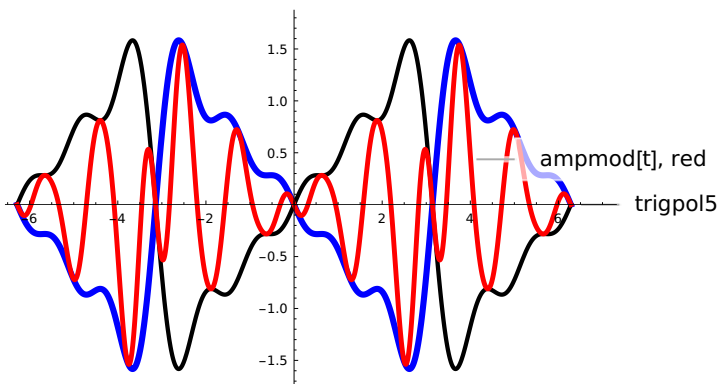
**This is one of the most important properties**, because in modern communication systems like mobile telephony, digital broadcasting or WLAN the information is transmitted in the complex amplitudes of trigonometric polynomials in high frequency bands. Amplitude modulation is therefore a method of transferring a signal with a limited bandwidth to a desired frequency band for transmission and returning it to the original frequency band for reception. Sidebands are suppressed in each case (once the left suppressed, once the right). Thus, by this method considerable power losses have to be accepted.

**Example 7.** We multiply the trigonometric polynomial generated last with  $\text{Cos}[5t]$ . The result is a spectral shift to the left and to the right, i.e. two sidebands and a halving of the spectral values. Everyone is probably familiar with an application example on radio receivers with transmissions that can be received via AM (amplitude modulation). The amplitude modulation  $\text{ampmod}[t]$  is shown in the subsequent plot:

```

In[ ]:= ampmod[t_] := trigpol5[t] Cos[5 t];
plota := Plot[{trigpol5[t]}, {t, -2 Pi, 2 Pi},
  PlotStyle -> Directive[{Blue}], Thickness[0.01]], PlotRange -> All,
  ImageSize -> Medium, PlotLabels -> Style[trigpol5, 14]];
plotb := Plot[{-trigpol5[t]}, {t, -2 Pi, 2 Pi},
  PlotStyle -> Directive[{Black}], Thickness[0.007]],
  PlotRange -> All, ImageSize -> Medium];
plotc := Plot[ampmod[t], {t, -2 Pi, 2 Pi},
  PlotStyle -> Directive[{Red}], Thickness[0.008]], PlotRange -> All,
  ImageSize -> Medium, PlotLabels -> Style["ampmod[t], red", 14]];
Show[plota, plotb, plotc]

```



Here the shifted spectrum and the corresponding trigonometric polynomial:

```
In[ ]:= FourierCoefficient [ampmod[t], t, k]
FourierTrigSeries [ampmod[t], t, 10]
```

$$\text{Out[ ]} = \begin{cases} -\frac{i}{20} & k = -10 \\ \frac{i}{20} & k = 10 \\ -\frac{i}{16} & k = -1 \parallel k = 9 \\ \frac{i}{16} & k = -9 \parallel k = 1 \\ -\frac{i}{12} & k = -8 \parallel k = 2 \\ \frac{i}{12} & k = -2 \parallel k = 8 \\ -\frac{i}{8} & k = -3 \parallel k = 7 \\ \frac{i}{8} & k = -7 \parallel k = 3 \\ -\frac{i}{4} & k = -6 \parallel k = 4 \\ \frac{i}{4} & k = -4 \parallel k = 6 \\ 0 & \text{True} \end{cases}$$

$$\text{Out[ ]} = -\frac{\sin[t]}{8} + \frac{1}{6} \sin[2t] - \frac{1}{4} \sin[3t] + \frac{1}{2} \sin[4t] - \frac{1}{2} \sin[6t] + \frac{1}{4} \sin[7t] - \frac{1}{6} \sin[8t] + \frac{1}{8} \sin[9t] - \frac{1}{10} \sin[10t]$$

To illustrate this, we will use only the “upper sideband signal” in the frequency band from 6 to 10 rad/s, subject it to renewed amplitude modulation with  $\cos[5t]$ , filter out (by hand) the lower sideband up to 5 rad/s in the result and plot the result. You can see the shape of the function `trigpol5[t]` again, but now with considerably reduced amplitudes.

```

In[ ]:= uppersidebandsignal [t_]:=
  - $\frac{1}{2}$  Sin[6 t] +  $\frac{1}{4}$  Sin[7 t] -  $\frac{1}{6}$  Sin[8 t] +  $\frac{1}{8}$  Sin[9 t] -  $\frac{1}{10}$  Sin[10 t];
pup = Plot[uppersidebandsignal [t], {t, -2 Pi, 2 Pi},
  PlotStyle → Directive [Blue, Thickness [0.006]],
  PlotRange → All, PlotLabel → Style["upper sideband signal", 12]];

```

```

TrigReduce [Cos[5 t] uppersidebandsignal [t]]

```

```

lowersidebandsignal [t_] =

```

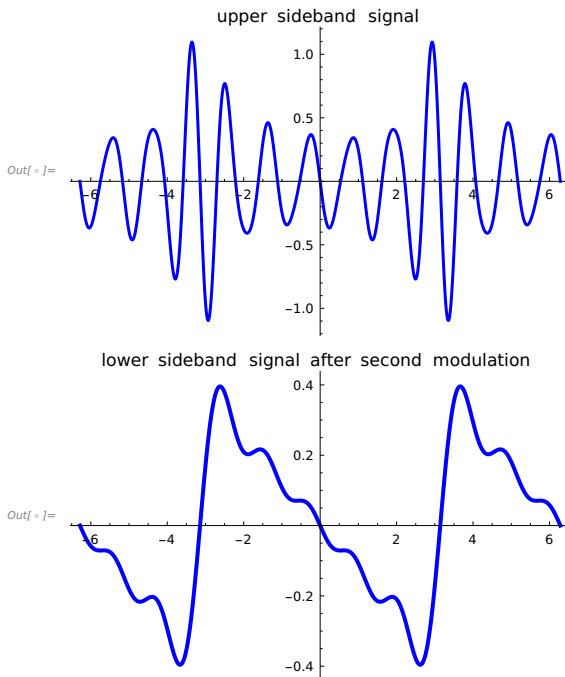
$$\frac{1}{240} \times (-60 \sin[t] + 30 \sin[2 t] - 20 \sin[3 t] + 15 \sin[4 t] - 12 \sin[5 t]);$$

```

plow = Plot[lowersidebandsignal [t], {t, -2 Pi, 2 Pi},
  PlotStyle → Directive [Blue, Thickness [0.008]], PlotRange → All,
  PlotLabel → Style["lower sideband signal after second modulation ", 12]];
Show[pup]
Show[plow]

```

$$\begin{aligned}
\text{Out[ ]} = & \frac{1}{240} \times (-60 \sin[t] + 30 \sin[2 t] - 20 \sin[3 t] + 15 \sin[4 t] - 12 \sin[5 t] - \\
& 60 \sin[11 t] + 30 \sin[12 t] - 20 \sin[13 t] + 15 \sin[14 t] - 12 \sin[15 t])
\end{aligned}$$



The result right is again the signal `trigpol5[t]`, **but with amplitudes decreased by a factor of 1/4.**

## Smoothness and Magnitude of the Spectrum

The smoother a periodic function is, the faster the spectral values approach zero for  $|k| \rightarrow \infty$ . We consider two examples.

**Example 8.** The first example is the function  $g(t)=(t^2)$ ,  $2\pi$ -periodically extended. It is continuous on  $\mathbb{R}$ , but not continuously differentiable.

Its spectrum decreases in magnitude like  $1/|k|^2$ . The second example is the function  $f(t)=t(\pi+t)$  on  $[-\pi,0]$  and  $f(t)=t(\pi-t)$  on  $[0,\pi]$ . Its  $2\pi$ -periodic extension is continuously differentiable with piecewise continuous second derivative.

Its spectrum decreases for  $|k| \rightarrow \infty$  like  $1/|k|^3$ .

```
In[ ]:= g[t_] := t^2
```

```
FourierTrigSeries [g[t], t, 6]
```

$$\text{Out}[ ] = \frac{\pi^2}{3} + 4 \left( -\cos[t] + \frac{1}{4} \cos[2t] - \frac{1}{9} \cos[3t] + \frac{1}{16} \cos[4t] - \frac{1}{25} \cos[5t] + \frac{1}{36} \cos[6t] \right)$$

```
In[ ]:= f[t_] := Piecewise[{{t (Pi + t), t <= 0}, {t (Pi - t), t >= 0}}]
```

```
FourierTrigSeries [f[t], t, 10]
```

$$\text{Out}[ ] = \frac{8 \sin[t]}{\pi} + \frac{8 \sin[3t]}{27\pi} + \frac{8 \sin[5t]}{125\pi} + \frac{8 \sin[7t]}{343\pi} + \frac{8 \sin[9t]}{729\pi}$$

It can be seen below that  $f$  has a continuously differentiable  $2\pi$ -periodic extension, which, however, only has a piecewise continuous second derivative.

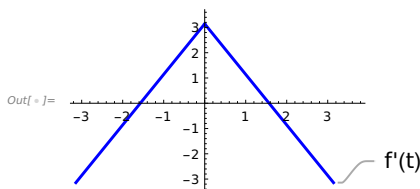
```
In[ ]:= f'[t]
```

```
Plot[f'[t], {t, -Pi, Pi},
```

```
PlotStyle -> Directive[Blue, Thickness[0.01]],
```

```
PlotRange -> All, PlotLabels -> Style["f'(t)", 14], ImageSize -> Small]
```

$$\text{Out}[ ] = \begin{cases} \pi + 2t & t < 0 \\ \pi & t = 0 \\ \pi - 2t & \text{True} \end{cases}$$



The correlations between (local) differentiability properties on the one hand and the (global) decay of the spectrum on the other hand show that, for example, small local perturbations with loss of differentiability properties can drastically change the entire spectrum of a function.

In the practice of signal processing such disturbances are often unavoidable and inevitably result in major challenges for problems in which estimates of the spectra of the signals shall be given from observed function data. The mathematical tools are preferably variants of low-pass filters for the so-called “denoising” before the spectral estimation. Conversely, in order to simulate steep signal slopes using a trigonometric polynomial requires high-frequency components in the approximation, i.e. in other words, a large bandwidth.

## Uniform Convergence of Fourier Series of Continuous Functions on a Closed Bounded Interval

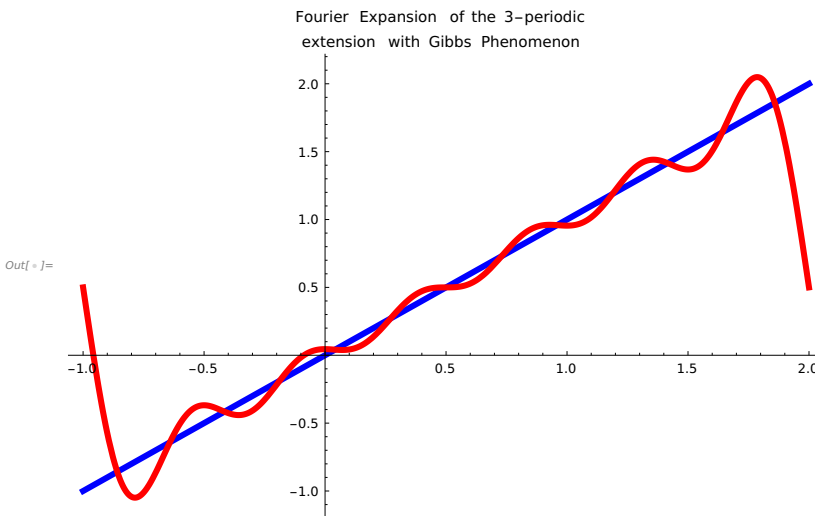
As we have seen, Fourier series of functions  $f$  on an interval  $[a,b]$  exhibit the Gibbs phenomenon. However, if you are interested in a trigonometric approximation of  $f$  in the interval  $[a,b]$  and  $f$  is continuous, you achieve uniform convergence on the closed interval  $[a,b]$  by extending the function to a continuous  $2(b-a)$ -periodic function on the entire real axis. You then get on  $[a,b]$  uniform convergent trigonometric approximations by periodic convolutions of  $f$  with the Fejér kernels. If  $f$  is continuous and additionally piecewise continuously differentiable, you achieve uniform convergent Fourier series approximations by  $2(b-a)$ -periodic convolutions of  $f$  with the according Dirichlet kernels. This is the basic fact for a proof of the **Weierstrass approximation theorem**, because the trigonometric approximations can again be uniformly approximated by their Taylor polynomials.

**Example 9.** Consider the function  $f(t)=t$  on  $[-1,2]$  and its 6-periodic continuous extension.

```

In[ ]:= f[t_] = t (HeavisideTheta[t + 1] - HeavisideTheta[t - 2]);
p1 = Plot[f[t], {t, -2, 4}, PlotLabel -> "f(t)"];
fextended1[t_] = f[t] + f[t - 3] + f[t - 6];
In[ ]:= p2 = Plot[fextended1[t], {t, -1, 2},
    PlotLabel -> "Fourier Expansion of the 3-periodic
    extension with Gibbs Phenomenon", PlotRange -> All, PlotStyle -> Directive[
        Blue, Thickness[0.008]]];
fser1[t_] = fourpolynomial[f, -1, 2, 6];
In[ ]:= p3 = Plot[fser1[t], {t, -1, 2}, PlotStyle -> Directive[
    Red, Thickness[0.008]]];
Show[p2, p3]

```

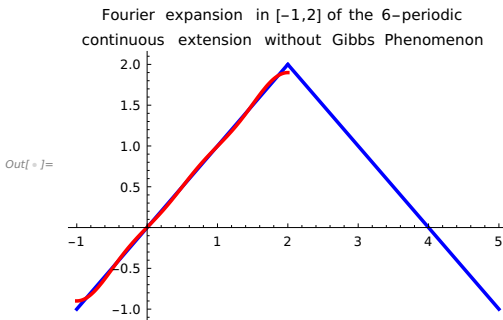


In the red curve we clearly see the Gibbs phenomenon, which prevents uniform convergence of the Fourier expansions in the interval  $[-1, 2]$ .

**Example 10.** Now, we extend the function  $f$  to a 6 - periodic continuous and piecewise continuously differentiable function on the entire real axis and build the Fourier series of that extension. Then, this series converges uniformly to  $f$  on the entire closed interval  $[-1, 2]$  (see the red approximation in the illustration below).

```
In[ ]:= fextended2[t_] = f[t] - (f[t - 3] - 1) (HeavisideTheta[t - 2] - HeavisideTheta[t - 5]);
p4 = Plot[fextended2[t], {t, -1, 5},
  PlotLabel -> "Fourier expansion in [-1,2] of the 6-periodic
  continuous extension without Gibbs Phenomenon ",
  PlotRange -> All, PlotStyle -> Directive[
    Blue, Thickness[0.008]]];
fser2[t_] = fourpolynomial[fextended2, -1, 5, 6]
75 π² - 900 Cos[π/3 t] + 200 Cos[π t] - 36 Cos[5π/3 t] - 36 √3 (-25 Sin[π/3 t] + Sin[5π/3 t])
Out[ ]:= 150 π²
```

```
In[ ]:= p5 = Plot[fser2[t], {t, -1, 2},
  PlotLabel -> "Fourier expansion in [-1,2] of the 6-periodic
  continuous extension without Gibbs Phenomenon ",
  PlotRange -> All, PlotStyle -> Directive[
    Red, Thickness[0.008]]];
Show[p4, p5]
```



### Fourier Series of Derivatives and Integrals

The Fourier coefficients of the derivative  $f'$  of a  $T$ -periodic function  $f$  with Fourier coefficients  $c_k$  are the coefficients  $ik\omega_0 c_k$  with  $\omega_0 = 2\pi/T$ . Of course, it is possible that the resulting series no longer converges at any point, as the example of the sawtooth series immediately shows.

If the derivative  $f'$  is piecewise continuous and its Fourier series converges at a point  $t_0$ , then its limit is

$(f'(t_0 +) + f'(t_0 -))/2$  by the theorem of Fejér with right and left sided limits.

Much simpler is the integration of Fourier series  $f$ : They can be integrated term by term.

As integral function  $\int_0^t f(x) dx$  we obtain a periodic function oscillating on the ramp  $c_0 t + F_0$ , where  $c_0$  is the mean value of  $f$  and  $F_0$  the mean value of the function  $\int_0^t (f(x) - c_0) dx$ .

**Example 11.** We consider as an example a partial sum of the sawtooth, shifted upwards by  $1/2$ : You see the trigonometric polynomial, afterwards its integral function, and both plotted

```
In[ ]:= f[t_] = 1/2 + Sum[Sin[k t]/k, {k, 1, 4}]
```

```
p1 = Plot[f[t], {t, -2 Pi, 2 Pi},
```

```
PlotStyle -> Directive[Blue, Thickness[0.008]],
```

```
PlotRange -> All, PlotLabels -> Style["f(t)", 14]];
```

```
intf[t_] = TrigReduce[Integrate[f[x], {x, 0, t}]]
```

```
Out[ ]:=  $\frac{1}{2} + \sin[t] + \frac{1}{2} \sin[2 t] + \frac{1}{3} \sin[3 t] + \frac{1}{4} \sin[4 t]$ 
```

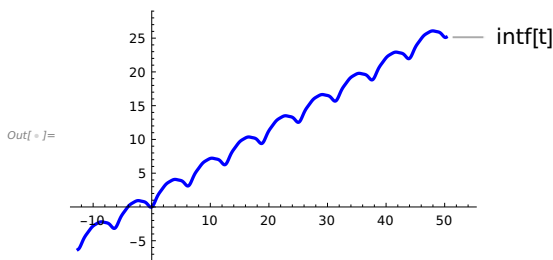
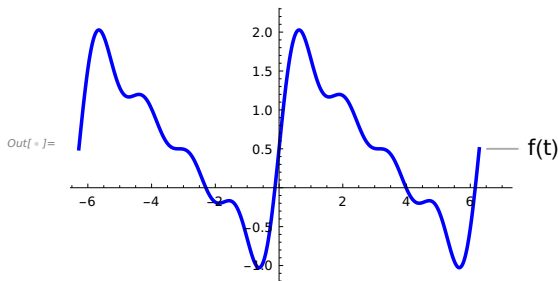
```
Out[ ]:=  $\frac{1}{144} \times (205 + 72 t - 144 \cos[t] - 36 \cos[2 t] - 16 \cos[3 t] - 9 \cos[4 t])$ 
```

```
In[ ]:= p2 = Plot[intf[t], {t, -4 Pi, 16 Pi}, PlotStyle -> Directive[Blue, Thickness[0.008]],
```

```
PlotRange -> All, PlotLabels -> Style["intf[t]", 14]];
```

```
Show[p1]
```

```
Show[p2]
```



The result is a periodic function oscillating on the ramp  $c_0 t + F_0$ ,  $c_0$  and  $F_0$  as above. The function is zero at the origin.

The complete Fourier series of the sawtooth has a spectral decay like  $1/k$  (the sawtooth is not continuous), the integral function has a spectrum, which decays like  $1/k^2$  for increasing  $|k|$  (it is continuous, but is not continuously differentiable).

Check  $F_0$

```
In[ ]:= F0 = Integrate[Integrate[f[x] - 1/2, {x, 0, t}], {t, 0, 2 Pi}]/(2 Pi)
```

```
Out[ ]:=  $\frac{205}{144}$ 
```



We come to the **important aspect** that Fourier expansions of T-periodic functions  $f$  can be seen as orthogonal projections of  $f$  into finite-dimensional subspaces of the space  $L^2([0, T])$ , the space of square integrable functions  $g: [0, T] \rightarrow \mathbb{C}$ .

A partial sum of degree  $\leq N$  of the Fourier series of a T-periodic square integrable function  $f$  is the best approximation for  $f$  in the finite-dimensional subspace of  $L^2([0, T])$ , which is generated by the trigonometric functions  $1, \cos[2\pi kt/T]$  and  $\sin[2\pi kt/T]$ ,  $k=1, \dots, N$ . It minimizes the norm of the error (and thus the power losses in the approximation). It is therefore the orthogonal projection of the function  $f$  into the subspace generated by trigonometric functions according to the degree of  $f$ .

The inner product to "measure" orthogonality is the one usually considered in  $L^2([0, T])$ . This optimization with respect to the error in the quadratic mean (resp. in the RMS mean, Root Mean Square) is one of the main reasons for the use of Fourier expansions in engineering, where good pointwise approximations are often less important than mean values as defined by the concept of power. The series of magnitude squares of the (complex) Fourier coefficients converges to the total power of the "signal"  $f$ . This is the meaning of the Parseval equation.

As an example again the sawtooth: First the series of the squared absolute values of the complex Fourier coefficients, afterwards the power calculated as mean square of the function. Their difference is zero, i.e., they are equal.

$$\text{Sum}[1/(2 k^2), \{k, 1, \text{Infinity}\}] - \text{Integrate}[(\text{Pi} - t)^2/4, \{t, 0, 2 \text{Pi}\}]/(2 \text{Pi})$$

Out[ ]= 0

$$\text{Sum}[1/(2 k^2), \{k, 1, \text{Infinity}\}]$$

Out[ ]=  $\frac{\pi^2}{12}$

### 1.3 Orthogonal Projections into Finite-Dimensional Subspaces of $L^2([a, b])$

Fourier series expansions of functions in  $L^2([0, T])$  with the orthogonal system of the trigonometric functions  $1, \cos[2\pi kt/T], \sin[2\pi kt/T]$ ,  $k=1, \dots, N$ , are only a first example of the general concept of calculating with functions in a space  $L^2([0, T])$  as function series with the help of a complete orthogonal system.

This corresponds to representations of vectors in finite dimensional spaces by different bases. The partial sums of such series are then again orthogonal projections into the finite-dimensional subspaces of  $L^2([0, T])$  generated by the participating basis functions, and thus the best approximations to the function in the respective subspaces in the sense of the  $L^2([0, T])$  norm.

There are many such orthogonal systems of functions in  $L^2([0, T])$ , analogously in  $L^2([a, b])$ , which are used in mathematics and technology. We consider here only one example, others in subsequent chapters. As an example, consider the task of approximating the function  $f(t) = \sin[3t]$  in the interval  $[-1, 1]$  by a polynomial. Polynomials are often preferred approximation functions for many reasons. They allow for particularly simple processing such as differentiating, integrating, calculating values.

As the first method of approximation, you usually learn Taylor expansion in the first semester. You also learn that a Taylor polynomial provides the exact function value at the development point, but the error often increases with the distance from the development point.

A polynomial approximation in mean square has the property that the approximating function oscillates around the given function, but remains near to it on the entire interval. We compare two approximations of a sine function, at first with the Taylor polynomial of degree 5 with expansion at zero, secondly an approximation in mean square with the orthogonal system of the Legendre polynomials in the real vector space  $L^2([-1, 1])$ .

As inner product we choose  $\langle f, g \rangle = \int_{-1}^1 f(x) g(x) dx$ .

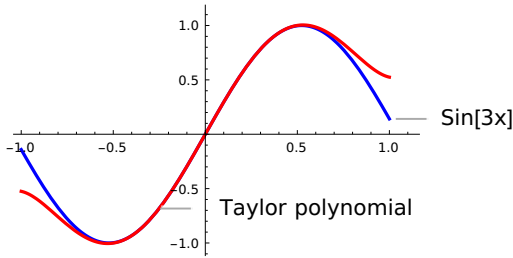
**Example 12.** Plot of the Taylor polynomial  $n1[t]$  of degree 5 for the sine and a plot of the sine function:

```

In[ ]:= n1[x_] = Normal[Series[Sin[3 x], {x, 0, 5}]]
p1 := Plot[Sin[3 x], {x, -1, 1}, PlotStyle -> Directive[Blue, Thickness[0.008]],
PlotRange -> All, PlotLabels -> Style["Sin[3x]", 14]]
p2 := Plot[n1[x], {x, -1, 1}, PlotStyle -> Directive[Red, Thickness[0.008]],
PlotRange -> All, PlotLabels -> Style["Taylor polynomial ", 14]]
Show[p1, p2, ImageSize -> Medium]

```

$$\text{Out[ ]:= } 3x - \frac{9x^3}{2} + \frac{81x^5}{40}$$



You see the typically increasing error with the distance from zero.

Now the approximation with the orthogonal system of the Legendre polynomials.

Mathematica knows these polynomials as LegendreP. In normed form they are defined by

$$\text{In}[*]:= \text{LPol}[k\_ , x\_ ] = 1 / (2^k k!) D[(x^2 - 1)^k, \{x, k\}] / \text{Sqrt}[2 / (2k + 1)]$$

$$\text{Out}[*]:= \frac{2^{-\frac{1}{2}-k} \text{y}[2+n][k]}{\sqrt{\frac{1}{1+2k}}}$$

As an example the normed Legendre polynomial of degree 3 and its  $L^2([-1, 1])$  norm with the inner product from above:

$$\text{In}[*]:= \text{Expand}[\text{LPol}[3, x]]$$

$$\text{Integrate}[\text{LPol}[3, x]^2, \{x, -1, 1\}]$$

$$\text{Out}[*]= -\frac{3}{2} \sqrt{\frac{7}{2}} x + \frac{5}{2} \sqrt{\frac{7}{2}} x^3$$

$$\text{Out}[*]= 1$$

The Legendre polynomials in *Mathematica* implemented as LegendreP[k,x] are differently normed.

For comparison LegendreP[3,x] in Mathematica and its norm with our inner product.

$$\text{In}[*]:= \text{Expand}[\text{LegendreP}[3, x]]$$

$$\text{Sqrt}[\text{Integrate}[\text{LegendreP}[3, x]^2, \{x, -1, 1\}]]$$

$$\text{Out}[*]= -\frac{3x}{2} + \frac{5x^3}{2}$$

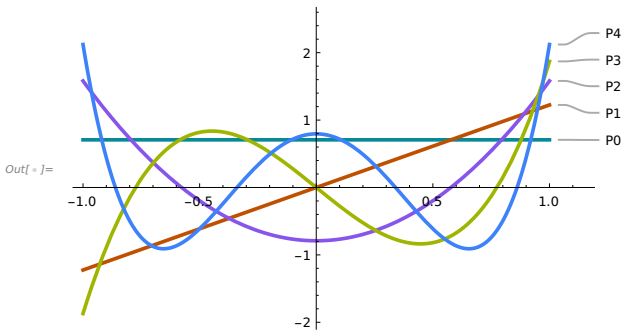
$$\text{Out}[*]= \sqrt{\frac{2}{7}}$$

Plot of the first 5 Legendre polynomials normed as above. Observe the symmetry properties and the numbers of zeros of them in [-1,1]. In the image they are labeled as P0 to P4:

```

In[ ]:= Plot[Evaluate[Table[LegendreP[n, x]/Sqrt[2/(2 n + 1)], {n, 0, 4}], {x, -1, 1},
  PlotStyle → Directive[Hue, Thickness[0.007]],
  PlotRange → All, PlotLabels → {"P0", "P1", "P2", "P3", "P4"}]

```



**Example 13.** Now to the approximation of  $\sin[3x]$  in mean square (often also called in root mean square RMS, if the root is taken) with the Legendre polynomials up to degree 5:

```

In[ ]:= Faktor[n_] := NIntegrate[LPol[n, x] Sin[3 x], {x, -1, 1}]
n2[x_] = Expand[Sum[Faktor[n] × LPol[n, x], {n, 0, 5}]]
(* (2/(2 n + 1)), {n, 0, 5}]], if you use LegendreP instead of LPol *)

0. + 2.97177 x - 4.23916 x^3 + 1.42043 x^5

```

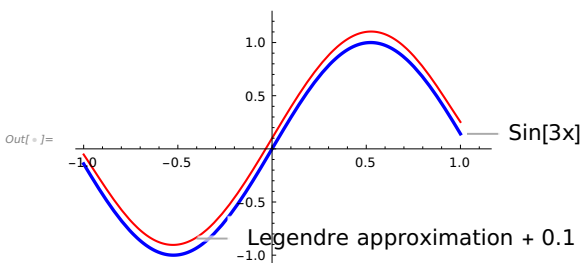
See the approximating polynomial, as above **plotted with an offset of +0.1**, so that one can distinguish it at all from the sine function.

Thus, we see that it is a better approximation over the whole interval than the Taylor polynomials. Finally, we look at the absolute errors for the Taylor polynomial and for the approximation with the Legendre polynomials.

```

In[ ]:= p3 := Plot[n2[x] + 0.1, {x, -1, 1},
  PlotStyle → Directive[Red, Thickness[0.005]], PlotRange → All,
  PlotLabels → Style["Legendre approximation + 0.1", 14]]
(* shown with offset +0.04 for visibility *)
Show[p1, p3]

```

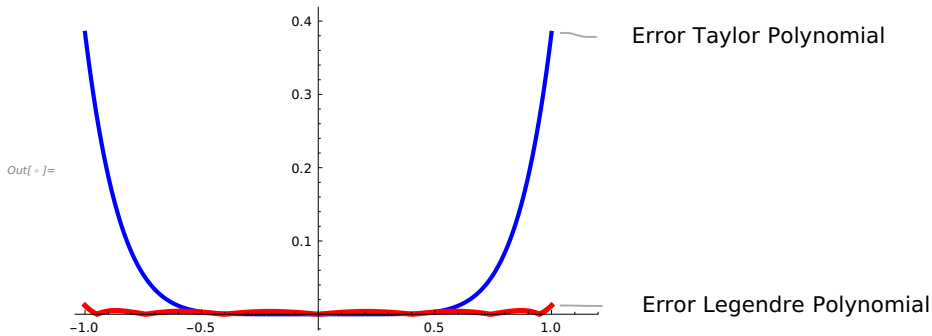


The approximation errors of the Taylor polynomial compared to the Legendre polynomial:

```

In[ ]:= p4 := Plot[Abs[Sin[3 x] - n1[x]], {x, -1, 1},
  PlotStyle -> Directive[Blue, Thickness[0.008]],
  PlotRange -> All, PlotLabels -> Style["Error Taylor Polynomial ", 14]]
p5 := Plot[Abs[Sin[3 x] - n2[x]], {x, -1, 1},
  PlotStyle -> Directive[Red, Thickness[0.01]], PlotRange -> All,
  PlotLabels -> Style["Error Legendre Polynomial ", 14]]
Show[p4, p5, ImageSize -> Medium ]

```



You realize that the concept of expanding a function according to an orthogonal system in spaces with an inner product can be a powerful tool, if you want or need to replace complicated functions by approximations with easier to handle functions. It should be pointed out that also when using orthogonal systems other than the trigonometric functions as in classical Fourier analysis for functions with jump points the Gibbs phenomenon often occurs again. To combat this, you can also use convolutions with suitable kernels, as we have seen above with the Fejér kernels for Fourier series.

### Approximation by Legendre polynomials

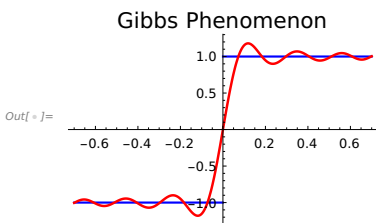
We see the Gibbs phenomenon and we also see increasing errors at the edges of the interval  $[-1,1]$ . The behavior at the edges can be influenced by introducing weight functions to the inner product. We will demonstrate this in examples of approximation and interpolation using Chebyshev polynomials in a subsequent section related to the discrete Fourier and discrete cosine transform (DFT and DCT).

**Example 14.** As an example, we take the sign function in the interval  $[-1,1]$  and use for approximation the Legendre polynomials as before, this time up to degree 25. The attempt to achieve a good approximation with a Taylor polynomial over the entire interval would be an unsuitable attempt a priori because of the jump point. For reasons of symmetry, we only have to consider the odd polynomials.

```

In[ ]:= f[x_] = 2 UnitStep[x] - 1;
p6 = Plot[f[x], {x, -0.7, 0.7},
PlotStyle -> Directive[Blue, Thickness[0.007]],
PlotRange -> All, PlotLabel -> Style["Gibbs Phenomenon", 14]];
factor2[n_] = Integrate[LegendreP[n, x] f[x], {x, -1, 1}];
n3[x_] = Sum[factor2[n] LegendreP[n, x] / (2 / (2 n + 1)), {n, 1, 25, 2}];
p7 = Plot[n3[x], {x, -0.7, 0.7},
PlotStyle -> Directive[Red, Thickness[0.008]],
PlotRange -> All, PlotLabel -> Style["Gibbs Phenomenon", 14]];
p7a = Show[p6, p7];
Show[p7a]

```



### A Fourier Series of a periodic function, which is not piecewise continuously differentiable

Since in my book - as in other introductory textbooks - mainly Fourier series representations of piecewise continuously differentiable periodic functions are treated for assertions on pointwise convergence, an example of a Fourier series for a periodic function is shown here, which does not fulfill this requirement.

```

In[ ]:= f[t_] := Log[Abs[2 Sin[t / 2]]] (UnitStep[t] - UnitStep[t - 2 Pi])

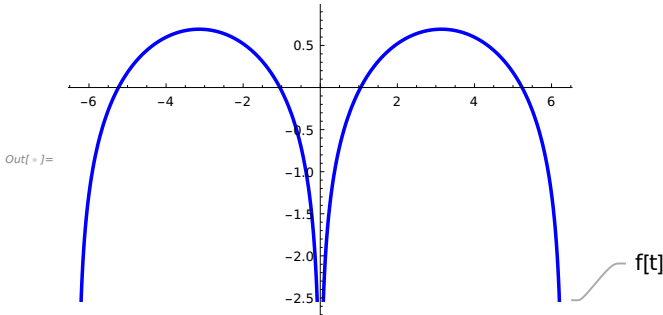
```

Below is the plot of the  $2\pi$  periodic extension in the interval  $[-2\pi, 2\pi]$ . Since there are no limits for  $t \rightarrow 0$  or  $t \rightarrow 2\pi$ , the function is not piecewise continuously differentiable, but can be integrated to  $[0, 2\pi]$ . It has a Fourier series representation for  $t \neq 2k\pi$ ,  $k$  in  $\mathbb{Z}$ .

We let Mathematica calculate a partial sum of this Fourier series. Although the series is very similar to the sawtooth series (there  $\sin(kt)$  in the summands of the partial sums, here it is  $-\cos(kt)$  instead), the function shown is completely different from the sawtooth. You can also see in the subsequent example that there are notable differences between sine series and cosine series.

**Example 15.**

```
In[ ]:= Plot[f[t + 2 Pi] + f[t], {t, -2 Pi, 2 Pi},
PlotStyle -> Directive[Blue, Thickness[0.007]],
ImageSize -> Small, PlotLabels -> Style["f[t]", 14]]
```



```
In[ ]:= FourierTrigSeries [f[t + 2 Pi] + f[t], t, 6]
```

$$\text{Out[ ]} = -\cos[t] - \frac{1}{2} \cos[2t] - \frac{1}{3} \cos[3t] - \frac{1}{4} \cos[4t] - \frac{1}{5} \cos[5t] - \frac{1}{6} \cos[6t]$$

### 1.4 Example of Convergent Trigonometric Series, which are not Classical Fourier Series

In the first example, a series is shown that is not a classical Fourier series.

In the second example below, we look at one of the first periodic functions you can think of, namely the **tangent** function. It is unbounded, thus not piecewise continuously differentiable. It does not have a classical Fourier series representation. It is possible only with distribution theory that the tangent can be understood as a periodic distribution with a so-called *generalized Fourier series representation* (see [1] and a later booklet on distributions).

#### Not a classical Fourier series

The example is a sine series converging everywhere, but it is not a classical Fourier series.

```
In[ ]:= f[t_] = Sum[Sin[k t] / Log[k], {k, 2, Infinity}]
```

$$\text{Out[ ]} = \sum_{k=2}^{\infty} \frac{\sin[kt]}{\log[k]}$$

#### A similar cosine series is a classical Fourier series

```
In[ ]:= Sum[Cos[k t] / Log[k], {k, 2, Infinity}]
```

$$\text{Out[ ]} = \sum_{k=2}^{\infty} \frac{\cos[kt]}{\log[k]}$$

Mathematica returns the definition and cannot determine an explicit simpler representation of the associated function. You can show that the sine series converges everywhere, in every interval  $[h, 2\pi - h]$ ,  $h > 0$ , even uniformly to a continuous function. The function  $f[t]$  cannot be a classical Fourier series of a function integrable on  $[0, 2\pi]$ , since the series  $\sum_{k=2}^{\infty} 1/(k \log[k])$  diverges. For the sine coefficients  $b_k$  of the Fourier series of an integrable function, however, the series  $\sum_{k=1}^{\infty} b_k/k$  must be convergent. For a cosine series, a corresponding coefficient condition does not apply. For example, the series  $\sum_{k=2}^{\infty} \cos(kt)/\log(k)$  is in fact a classical Fourier series (to be found in [11] A. Zygmund, *Trigonometric Series*, chapter V, section 1). These examples alone show subtle mathematical facts as soon as one is interested in point-wise representations of periodic functions by Fourier series. The difficulties are caused in particular by the integral definition used, because Fourier coefficients are to be calculated by integration. They were already the reason for the development of set theory by G. Cantor, the development of the integral concepts of B. Riemann and later of H. Lebesgue, for numerous works of great mathematicians such as G. H. Hardy, A. Zygmund and many others, as well as for the development of the modern concept of functions by A.L. Cauchy, P.L. Dirichlet and the entire mathematical development in analysis since about the middle of the 19th century.

**Now to the tangent function.** It does not have a classical Fourier series representation. Mathematica does not return a result for the Fourier series expansion.

We will deal with the representation of the tangent function as a periodic distribution in another booklet on "Distributions and Application Examples with Mathematica".

```
In[ * ]:= FourierTrigSeries [Tan[t], t, 5]
```

```
Out[ * ]= FourierTrigSeries [Tan[t], t, 5]
```

## 1.5 Graphical Illustrations of Trigonometric Polynomials

### Graphical representation of a trigonometric polynomial as a circular wave

```
Clear["Global`*"]
```

```
In[ * ]:=  $\omega_0 = \text{Pi} / 2$ ;  $T = 2 \text{ Pi} / \omega_0$ ;
```

```
 $P[t\_] = 1/2 \text{ Sin}[\omega_0 t] + i \text{ Sin}[2 \omega_0 t] - \text{Cos}[3 \omega_0 t]$ 
```

```
Table[{t, Re[P[t]], Im[P[t]]}, {t, 0, 6, 0.015}];
```

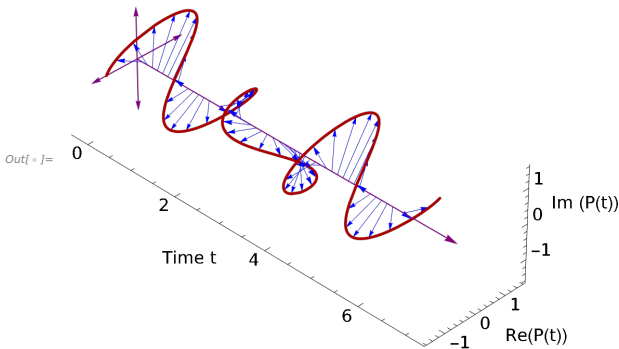
```
Out[ * ]=  $-\text{Cos}\left[\frac{3 \pi t}{2}\right] + \frac{1}{2} i \text{ Sin}\left[\frac{\pi t}{2}\right] + i \text{ Sin}[\pi t]$ 
```



```

In[ ]:= Show[(*circularly wave*)
  Normal[ParametricPlot3D[{t, Re[P[t]], Im[P[t]]}, {t, 0, 6}, Mesh → 50,
    MeshStyle → Directive[Thin, Blue],
    PlotStyle → Directive[Darker[Red], Thickness[0.006], Arrowheads[.02]]] /.
    Point[{x_, y_, z_}] := If[Chop[Norm[{y, z}]] < 0.1, Point[{x, 0, 0}],
      Arrow[{x, 0, 0}, {x, y, z}]], (*axes*)
  Graphics3D[{{Purple, {Arrowheads[.025], Arrow[{0, 0, 0}, {7, 0, 0}]},
    {Arrowheads[.025] {-1, 1}}, Arrow[{0, 1.5, 0}, {0, -1.5, 0}]},
    Arrow[{0, 0, 1.5}, {0, 0, -1.5}]}],
  Axes → True, AxesLabel → {"Time t", "Re(P(t))", "Im(P(t))"},
  LabelStyle → Directive[Black, FontSize → 11], Boxed → False,
  BoxRatios → {7, 2, 2}, PlotRange → All, ViewPoint → {5, -5, 5}]

```



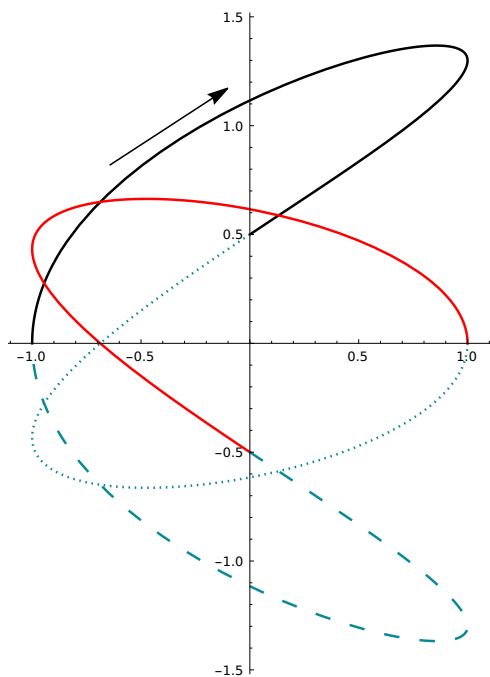
### Representation of the same example as a curve in the complex plane over one period.

The curve starts at  $(-1,0)$  and changes color and line dashing with increasing  $t$  all  $T/4$ .

```

In[ ]:= plot1a = ParametricPlot[{Re[P[t]], Im[P[t]]},
  {t, 0, T/4}, PlotStyle → {Black}, PlotRange → All];
plot1b = ParametricPlot[{Re[P[t]], Im[P[t]]}, {t, T/4, T/2},
  PlotStyle → {Dashing[Tiny]}, PlotRange → All];
In[ ]:= plot2a = ParametricPlot[{Re[P[t]], Im[P[t]]},
  {t, T/2, 3 T/4}, PlotStyle → {Red}, PlotRange → All];
plot2b = ParametricPlot[{Re[P[t]], Im[P[t]]}, {t, 3 T/4, T},
  PlotStyle → {Dashing[Large]}, PlotRange → All];
In[ ]:= Show[plot1a, plot1b, plot2a, plot2b, ImageSize → Medium]

```

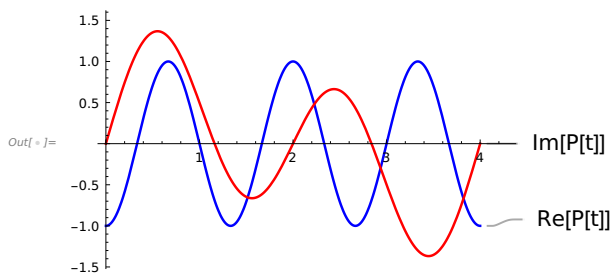


### Representation of Real Part and Imaginary Part over one period

```

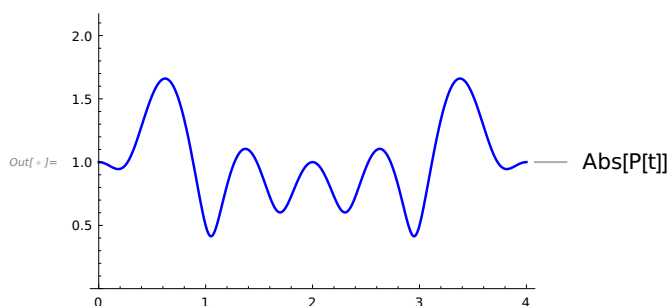
In[ ]:= plot3 = Plot[Re[P[t]], {t, 0, T}, PlotStyle → Directive[Blue],
  PlotRange → All, PlotLabels → Style["Re[P[t]", 14]];
plot4 = Plot[Im[P[t]], {t, 0, T}, PlotStyle → Directive[Red],
  PlotRange → All, PlotLabels → Style["Im[P[t]", 14]];
Show[{plot3, plot4}, ImageSize → Medium]

```

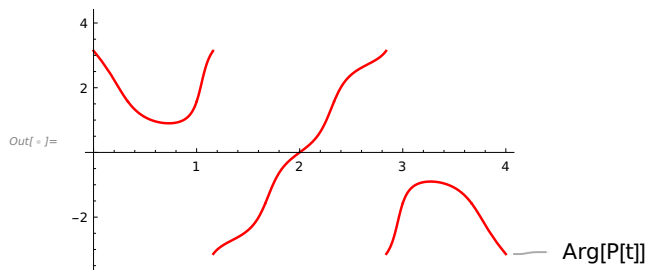


### Representation by Magnitude and Phase over one period

```
In[ ]:= plot5 = Plot[Abs[P[t]], {t, 0, T}, PlotStyle -> Directive[Blue],
  PlotRange -> {0, 2}, PlotLabels -> Style["Abs[P[t]]", 14]]
```



```
In[ ]:= plot6 = Plot[Arg[P[t]], {t, 0, T}, PlotStyle -> Directive[Red],
  PlotRange -> {-6/5 Pi, 6/5 Pi}, PlotLabels -> Style["Arg[P[t]]", 14]]
```



## 2 Application of Fourier Series to Linear Differential Equations

The aim of the chapter is to show some application examples of Fourier series with the help of Mathematica for the solution of linear differential equations with constant coefficients. These examples demonstrate the benefits of Fourier analysis for such differential equations. In a later booklet, when we explain distributions and Fourier transforms with Mathematica, we will also treat 3D heat and wave equations, further approximation tasks and signal processing examples. Here, we will look at the DFT, DCT and Chebyshev polynomials and some of their typical properties that are relevant for applications in the subsequent chapter.

### 2.1 Stable Ordinary Linear Differential Equations with Constant Coefficients

Asymptotically stable linear ordinary differential equations  $P(D)u=f$  of order  $n$  with constant coefficients have characteristic polynomials  $P(z) = \sum_{k=0}^n a_k z^k$  ( $D$  stands for the differential operator  $d/dt$ ), whose zeros all have negative real parts. It is necessary that  $P$  is a so-called Hurwitz polynomial, i.e., that all coefficients have the same sign.

In my textbook, cited at the beginning, it is shown that such equations with a continuous, piecewise continuously differentiable  $T$ -periodic right-hand side  $f$  have a uniquely determined  $T$ -periodic solution  $u$ , which can be obtained by the  $T$ -periodic convolution of  $f$  with the corresponding  $T$ -periodic transfer function  $h$ .

The Fourier series of this  $T$ -periodic transfer function has the Fourier coefficients  $h_k = 1/P(i\omega_0 k)$ ,  $i^2 = -1$ ,  $\omega_0 = 2\pi/T$ . The convolution is  $n$ -times continuously differentiable. It describes the long-term behavior of the solution after the (rapid) decay of the transient process and usually has a different amplitude and phase spectrum than the excitation  $f$ .

#### a) A forced periodic oscillation of a mass on a spring

**Example 1.** Let a mass  $m$  on a spring be excited by a periodic force  $F$ . The damping coefficient is  $k>0$ , the spring constant  $d>0$ . The describing differential equation for the displacement  $y[t]$  is

$$\text{In[ ]:= } \text{dgl} = m \, y''[t] + k \, y'[t] + d \, y[t] = F[t]$$

$$\text{Out[ ]:= } d \, y[t] + k \, y'[t] + m \, y''[t] = F[t]$$

To illustrate this, we set  $F(t) = F_0 \cos[\omega t]$ ,  $F_0 = 0.2$  N,  $m = 1$  kg,  $k = 0.2$  kg/s,  $d = 1/2$  N/m,  $\omega = 2\pi$  rad/s. The equation is asymptotically stable, as can be seen immediately from the zeros of the characteristic polynomial: It has conjugate complex zeros with negative real parts.

$$\text{In[ ]:= } m := 1; k := 0.2; d := 1/2; F_0 := 0.2; \omega := 2\pi; F[t_] := F_0 \cos[\omega t];$$

$$P[z_] := m z^2 + k z + d;$$

$$\text{zeros} = \text{Solve}[P[z] == 0, z]$$

$$\text{Out[ ]:= } \{\{z \rightarrow -0.1 - 0.7 i\}, \{z \rightarrow -0.1 + 0.7 i\}\}$$

The following is the periodic solution obtained by setting the solution of the homogeneous differential equation is set to zero, here by setting the occurring parameters  $C[1]$ ,  $C[2]$  to

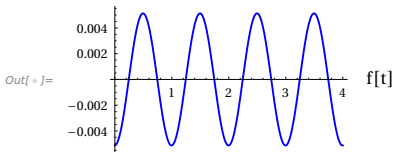
zero. This solution has the same frequency as the exciting force  $F$  with the oscillation period  $T=1s$ , but a different phase and a different, smaller amplitude. It describes the resulting oscillation after the transient decays. We first solve the equation using the existing Mathematica algorithms.

```
In[ ]:= solution = DSolve[dgl, y, t];
```

We call the periodic solution  $f$ , as defined by the Mathematica result and graphically shown in a section  $0 \leq t \leq 4$ . Observe the strong damping compared to  $F(t)$ .

```
In[ ]:= f := solution[[1, 1, 2]] /. {C[1] -> 0, C[2] -> 0}
```

```
Plot[f[t], {t, 0, 4}, PlotStyle -> Directive[
    Blue, Thickness[0.008]], PlotRange -> All,
    ImageSize -> Small, PlotLegends -> Style["f[t]", 12]]
```



The resulting amplitude and phase shift are simply obtained from the frequency response. In this case, the Fourier coefficients of the excitation  $F$  are to be multiplied by  $1/P(\pm i\omega)$ . We first generate the Fourier coefficients of  $F$  and thus once again the solution, which can be obtained by periodically convolving  $F$  with the corresponding periodic transfer function, i. e., by multiplying the associated Fourier coefficients. As always in time - invariant linear systems, "no new frequencies" are generated. We state the solution once again using our existing knowledge of Fourier series and this time call the result  $g$ :

```
In[ ]:= FourierCoefficient[F[t], t, n, FourierParameters -> {1, 2 Pi}]
```

```
Out[ ]:= { 0.1  n == -1 || n == 1
          0.    True
```

Now the trigonometric polynomial, which solves the equation and a plot

```
In[ ]:= g[t_] = FullSimplify[1/(10 P[-i ω]) Exp[-i ω t] + 1/(10 P[i ω]) Exp[i ω t]]
```

```
Out[ ]:= -0.00512572 Cos[2 π t] + 0.00016525 Sin[2 π t]
```

Now the resulting amplitude and phase shift, calculated from the frequency response  $1/P(i\omega)$ . The parameter  $\omega$  was the angular frequency of the excitation.

```
In[ ]:= amp = N[2 Abs[FourierCoefficient[F[t], t, 1, FourierParameters -> {1, 2 Pi}]/P[i ω]]]
```

```
Out[ ]:= 0.00512838
```

```
In[ ]:= phase = N[Arg[1/P[i ω]]]
```

```
Out[ ]:= -3.10936
```

The positive resonant circular frequency and the solution amplitude at such a frequency results from the maximum value of  $|1/P(i\omega)|$ ,  $\omega > 0$ . It is  $\omega_r = \sqrt{d/m - k^2/(2m^2)}$ . You can also search with Mathematica for this extreme point with the FindMaximum command.

Compare the value with the shown below. Since our excitation frequency was far above the resonance frequency, we observe a very strong amplitude damping in the resulting oscillation.

In control engineering, the differential equation of the same form, describes a so-called  $PT_2$  element, or in other words a 2nd order low-pass filter.

The **resonant angular frequency** is

```
In[*]:= resonant = N[Sqrt[d / m - k^2 / (2 m ^ 2)]] (* angular frequency *)
```

```
Out[*]:= 0.69282
```

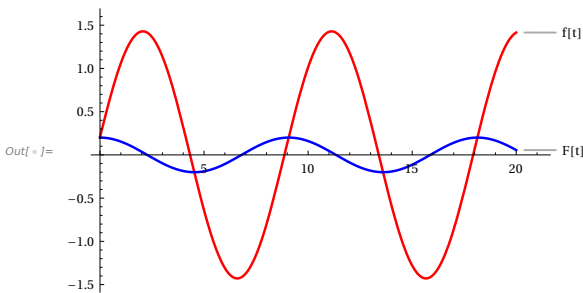
The amplitude at that angular frequency would be

```
In[*]:= N[2 Abs[FourierCoefficient [F[t], t, 1,
FourierParameters -> {1, 2 Pi}]] / P[i resonant ]]]
```

```
Out[*]:= 1.42857
```

*Let us test it and see the phase shift (delay) in the solution for excitation with the resonant frequency*

```
In[*]:= ω := 0.69282; F[t_] := F0 Cos[ω t]; solution := DSolve[dgl, y, t];
f = solution[[1, 1, 2]] /. {C[1] -> 0, C[2] -> 0};
Plot[{f[t], F[t]}, {t, 0, 20}, PlotStyle -> {Red, Blue}, PlotStyle -> Directive[
Thickness[0.005]], PlotRange -> All, ImageSize -> Medium,
PlotLabels -> {"f[t]", "F[t]"}] (* Force and Solution *)
FindMaximum[f[t], {t, 0, 6}]
```



```
Out[*]:= {1.42857, {t -> 2.06034}}
```

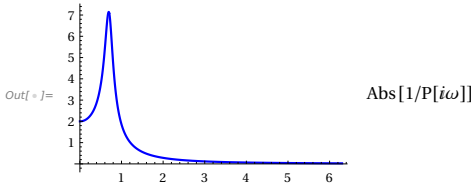
Below is the amplitude response and the phase response of the pendulum, considered as a linear transmission system in analogy to a 2nd order lowpass filter.

A representation only for positive angular frequencies is sufficient due to the known symmetries.

```

In[ ]:= p1 = Plot[Abs[1 / P[i ω]], {ω, 0, 6.3}, PlotStyle → Directive [
      Blue, Thickness [0.008]], PlotRange → All,
      ImageSize → Small, PlotLegends → Style["Abs[1/P[iω]]", 12]]
FindMaximum [Abs[1 / P[i ω]], {ω, 0, 2}]

```

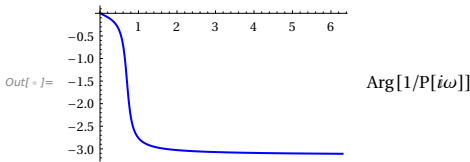


```
Out[ ]:= {7.14286, {0.69282 → 0.69282}}
```

```

In[ ]:= p2 = Plot[Arg[1 / P[i ω]], {ω, 0, 6.3}, PlotStyle → Directive [
      Blue, Thickness [0.008]], PlotRange → All,
      ImageSize → Small, PlotLegends → Style["Arg[1/P[iω]]", 12]]

```



The maximum of the frequency response shows again the amplification at the resonant frequency from amplitude 0.2 of  $F$  to 1.4287 in the system response  $f$ , when the system input  $F$  has that resonant frequency, and you can see the delay between input and response from the phase shift (about 2s in that example).

## b) A Fourier series as periodic force

**Example 2.** Now, we consider the same equation with excitation

$$F[t] = F_0 \sum_{k=1}^{\infty} \cos[3 k t] / k^2 = F_0 \sum_{k=-\infty, k \neq 0}^{+\infty} \exp[i 3 k t] / (2 k^2).$$

The series is the  $2\pi/3$ -periodic extension of  $f$ , on  $[0, 2\pi/3[$  defined by

$$f[t] = F_0 \left( (3t - \pi)^2 / 4 - \pi^2 / 12 \right).$$

The function is continuous and piecewise continuously differentiable.

With the periodic transfer function having the Fourier coefficients  $1/P(i 3 k)$  (the basic angular frequency is now  $3 \text{ rad/s}$ , the according period  $T=2\pi/3 \text{ s}$ ) the periodic solution  $u$  can immediately be written in the form of a Fourier series. It is

$$u[t] = \sum_{k=-\infty, k \neq 0}^{+\infty} F_0 e^{i 3 k t} / (2 k^2 P[i 3 k]).$$

For illustration purposes and to save excessive calculation time, we will content ourselves with a partial sum of the series for the force as an approximation and a corresponding trigonometric polynomial as an approximation for the solution. We plot over three periods of the excitation and of the solution. The amplitude attenuations and phase delays of all

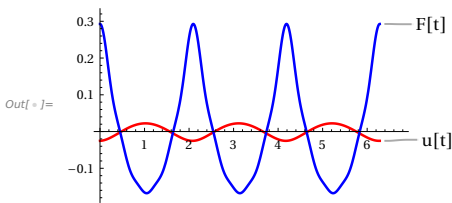
partial oscillations interact in the resulting pendulum motion and, as usual, the solution is characterized by smoothing, which is caused mathematically by the periodic convolution, is "less angular" than the excitation and has significantly less power than the excitation. Physically speaking, the cause of this smoothing lies in the mass inertia: due to its inertia, the mass can no longer follow the excitation frequency after just a few harmonics due to its inertia.

Therefore, for physical considerations, even a few members of the Fourier series solution in the analysis of damped oscillations, as in the example, are sufficient to very good approximations of the "exact" solution. Simply compare the approximate solutions with only 5 as in the following and up to 100 or 500 summands of a partial sum of this solution.

```

In[ ]:= F[t_] := F0 Sum[Exp[-i n 3 t]/(2 n^2) + Exp[i n 3 t]/(2 n^2), {n, 1, 5}]
u[t_] := F0 Sum[Exp[-i n 3 t]/(2 n^2 P[-i 3 n]) + Exp[i n 3 t]/(2 n^2 P[i 3 n]), {n, 1, 5}]
g1 := Plot[u[t], {t, 0, 2 π}, PlotStyle → Directive [
    Red, Thickness [0.008]], PlotRange → All, PlotLabels → Style["u[t]", 12]]
g2 := Plot[F[t],
    {t, 0, 2 π}, PlotStyle → Directive [Blue, Thickness [0.008]],
    PlotRange → All, PlotLabels → Style["F[t]", 12]]
Show[g1, g2]

```



For comparison the mean squares (squares of the  $L^2$ -Norms, Powers) of force and resulting solution, showing the loss of power:

```

In[ ]:= NIntegrate[Abs[F[t]]^2/(2 π/3), {t, 0, 2 π/3}] (* system input *)
NIntegrate[Abs[u[t]]^2/(2 π/3), {t, 0, 2 π/3}] (* system output *)

```

Out[ ]:= 0.021607

Out[ ]:= 0.000276477

**Summary:** All the considerations and calculations carried out here for the pendulum as an example can be applied analogously to stable time-invariant linear transmission systems, which are described by differential equations as above (also of higher orders).

In electrical engineering, they are the basis on which a subject such as “**AC calculations**” (alternating current calculations) with the concepts of frequency, amplitude and phase response and the quantities derived from these such as the “group delay” make sense in the first place, as real signals hardly ever have a pure sine shape, but if they are time-limited in a time interval  $[0, T]$ , they can be understood and treated with the theory of Fourier series. Fourier series are therefore a valuable theoretical tool for describing a large number of processes in technology.



## 2.2 Fourier series in homogeneous 1D Heat Equations

### Example 3. A homogeneous heat conduction equation with homogeneous boundary conditions.

The initial boundary value problem for the homogeneous heat conduction equation for a (thermally thin) rod of length  $L$ , whose ends are ice-cooled, described as the interval  $[0, L]$  and an initial temperature  $f[x]$  in  $x \in [0, L]$ , is given by the coefficient  $a$  for the thermal diffusivity (see textbooks on physics)

$$\partial_t u[x, t] = a \partial_{x,x} u[x, t], \quad u[x, 0] = f[x], \quad u[0, t] = u[L, t] = 0 \text{ for all } t \geq 0.$$

With a separation approach  $u[x, t] = v[x] w[t]$  one obtains -quite analogous to the solution of the differential equation for the vibrating string in [1], p. 2-4 and exercise A8 in [1], 5.7) - the Fourier series solution  $u[x, t] = \sum_{n=1}^{\infty} b_n e^{-\lambda_n^2 t} \sin[n\pi x / L]$  with the Fourier sine coefficients of the  $2L$ -periodic odd extension of  $f$  and  $\lambda_n = n\pi \sqrt{a} / L$ .

It has a uniformly convergent Fourier series. For demonstration, we choose  $L=1$  m and  $f$  as a parabolic arc  $f[x]=5x(L-x)$  in  $^{\circ}\text{C}$  (degrees Celsius) and the thermal diffusivity  $a = 117 \cdot 10^{-6} \text{ m}^2/\text{s}$  of copper at about  $20^{\circ}\text{C}$ . In the following, we calculate the Fourier coefficients of  $f$  and plot an approximation as a 3D graph that illustrates the temperature equalization in the rod for  $t \geq 0$ .

```

In[ ]:= Clear["Global`*"]; L := 1;
f[x_] := 40 x (L - x); a := 117 × 10-6;
b[n_] = 2 / L Integrate [f[x] Sin[n Pi x / L], {x, 0, L}]

Out[ ]:= -  $\frac{80 \times (-2 + 2 \cos[n \pi] + n \pi \sin[n \pi])}{n^3 \pi^3}$ 

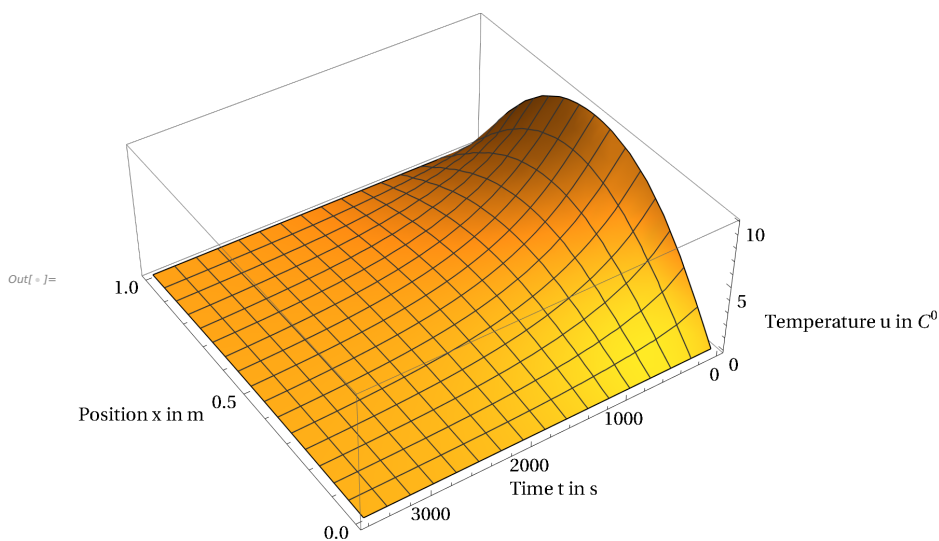
```

Here is a Fourier series approximation for the solution as a trigonometric polynomial of degree  $m$  and a representation with  $m=4$ . It decreases (slowly) to zero with increasing time:

```

In[ ]:= u[m_, x_, t_] := Sum[b[n] Exp[-(n Pi Sqrt[a]/L)^2 t] Sin[n Pi x / L], {n, 1, m}]
Plot3D[u[4, x, t], {x, 0, L}, {t, 0, 3600},
  PlotRange -> All,
  AxesLabel -> {" Position x in m ", " Time t in s ", " Temperature u in C^0 "},
  AxesStyle -> Directive[Black, 12], ViewPoint -> {-3, 2, 3}, ImageSize -> Medium ]

```



**Example 4. Now to the same problem with Neumann boundary conditions**, i. e., physically with insulated bar ends, so that no heat flows over the edge, mathematically  $\partial_x T[0, t] = \partial_x T[L, t] = 0$  for the temperature  $T[x, t]$ .

We solve the problem step by step analogous to the procedure behind the solution (for own work) in the previous example. The separation approach  $u[x, t] = v[x] w[t]$  leads to two ordinary differential equations  $v'' + cv = 0$  and  $w' + \alpha c w = 0$  and the boundary condition  $v'[0] = 0$ ,  $v'[L] = 0$  (the thermal diffusivity is now denoted by  $\alpha$ , the temperature by  $T$ ):  
With a separation approach, insertion into the equation and the boundary conditions we obtain for

$$\partial_t T[x, t] - \alpha \partial_{x,x} T[x, t] = 0 \text{ and } \partial_x T[0, t] = \partial_x T[L, t] = 0 :$$

`In[ ]:= sol1 = DSolve[{v''[x] + c v[x] == 0}, v[x], x]`

`Out[ ]:= {{v[x] -> c1 Cos[√c x] + c2 Sin[√c x]}}`

`In[ ]:= v[x_] = sol1[[1, 1, 2]]`

`In[ ]:= c1 Cos[√c x] + c2 Sin[√c x]`

`Out[ ]:= c1 Cos[√c x] + c2 Sin[√c x]`

We insert the boundary condition into the solution part  $v$  and obtain from  $v'[0] = 0$  that  $c_2$  must be zero, if  $c > 0$ . (For  $c < 0$  there is only the trivial solution).

`In[ ]:= v'[0]`

`Out[ ]:= √c c2`

It remains  $v'[L] = -C[1] \sqrt{c} \sin[\sqrt{c} L] = 0$ .

The resulting possibilities for the constant  $c$  have the form

$$c_n = n^2 \pi^2 / L^2 \text{ for } c, \text{ which provide all solutions.}$$

`In[ ]:= Solve[Sqrt[c] L == n Pi, c]`

... **Solve**: Solutions may not be valid for all values of parameters.

`Out[ ]:= {{c -> n^2 π^2}}`

**First result:** A sequence of possible solutions  $v_n[x]$ ,  $n=1,2, \dots$ ,  $v_n[x] = a_n \cos[\sqrt{c_n} x]$ , fulfilling all boundary conditions.

Now, to the second part of the separation approach:

`In[ ]:= sol2 = DSolve[w'[t] + α n^2 π^2 / L^2 w[t] == 0, w[t], t]`

`Out[ ]:= {{w[t] -> e^{-n^2 π^2 t α} c1}}`

We now obtain a Fourier series solution by superposition and write down a section of the solution series of degree  $m_2$

( $T[x, t] = c_0$  is also a solution):

$$T[m1_, m2_, x_, t_] = c_0 + \text{Sum}\left[a[n] \exp\left[-\frac{\alpha n^2 \pi^2 t}{L^2}\right] \cos[n \pi x / L], \{n, m1, m2\}\right]$$

According to the given initial condition, the  $a[n]$  must therefore be the Fourier cosine coefficients of the initial condition  $f$ ,  $c_0$  is the mean value of  $f$ . Using the same data as in the first example, we calculate and plot the approximate temperature curve with the partial sum of degree 8. With varying parameters  $m1$ ,  $m2$  you can see the development of sections of the

series.

```
In[1]:= Clear["Global`*"];
L := 1;
f[x_] := 40 x (L - x);
alpha = 117 × 10 ^(-6);
a[n_] = N[2/L Integrate[f[s] Cos[n Pi s/L], {s, 0, L}]]
c0 = 1/L Integrate[f[s], {s, 0, L}]

Out[5]= - 
$$\frac{2.58012 \times (3.14159 n + 3.14159 n \cos[3.14159 n] - 2. \sin[3.14159 n])}{n^3}$$


Out[6]= 
$$\frac{20}{3}$$

```

You can recognize an extensive temperature equalization after about 10 minutes in the entire rod.

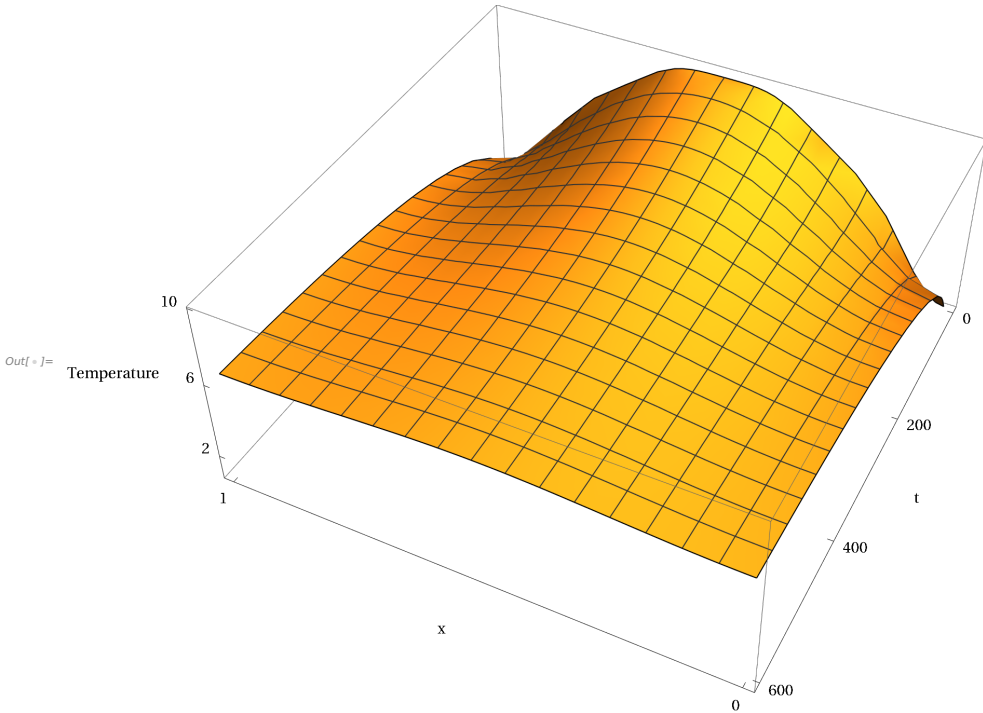
As the time increases, the temperature converges everywhere towards the average value  $c_0 = 20/3$  °C of  $f$ , as can be seen from the solution series. Here the temperature after 600 s and the mean value of  $f$ :

```
In[7]:= T[m1_, m2_, x_, t_] :=
c0 + Sum[a[n] Exp[-alpha n^2 Pi^2 t/L^2] Cos[n Pi x/L], {n, m1, m2}]
```

```

In[ ]:= p0 = Plot3D[N[T[1, 8, x, t]], {x, 0, 1}, {t, 0, 600},
  PlotRange → All, AxesLabel → {"x", "t", "Temperature"},
  AxesStyle → Directive[Black, 10],
  ViewPoint → {-1, 2, 2}, Ticks → {{0, 1}, {0, 200, 400, 600}, {2, 6, 10}}]

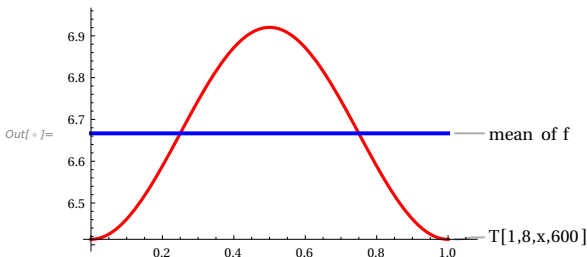
```



```

In[ ]:= p1 = Plot[T[1, 8, x, 600], {x, 0, L}, PlotStyle → Directive[
  Red, Thickness[0.008]], PlotRange → All, PlotLabels → Style["T[1,8,x,600]",
p2 = Plot[c0, {x, 0, L}, PlotStyle → Directive[
  Blue, Thickness[0.01]], PlotRange → All, PlotLabels → Style["mean of f", 12
p2a = Show[p1, p2]

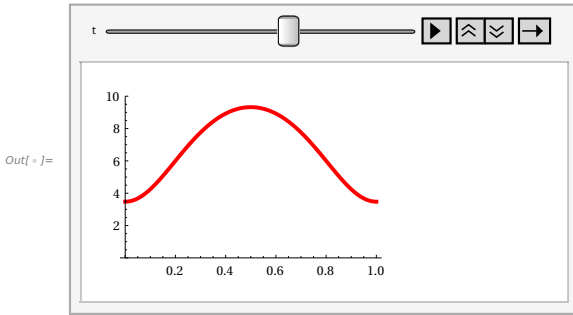
```



The following is a simple animation for temperature equalization using Mathematica over the course of 2 minutes. Since we are calculating with an approximate sum for the initial condition, the boundary temperature is not exactly zero at the beginning, as a partial sum for the 2 - periodically extended parabolic arc never reaches into the edges. You can use the

animation copying it into an own Mathematica notebook and taking it as first sample for similar purposes.

```
Animate[Plot[N[T[1, 8, x, t]], {x, 0, 1}, PlotRange -> {0, 10},
  PlotStyle -> Directive[Red, Thickness[0.015]], ImageSize -> Small], {t, 0, 120},
  AnimationRepetitions -> 1, AnimationRate -> 20, RefreshRate -> 10]
```



Finally, here is a quick test to check that everything is correct.  $T$  solves the homogeneous equation and each partial sum of the Fourier series solution fulfills the boundary conditions by construction:

```
In[ ]:= test = Chop[FullSimplify[ $\partial_t T[1, 8, x, t] - \alpha \partial_{x,x} T[1, 8, x, t]$ ]]
D[T[1, 8, x, t], x] /. x -> 0
Chop[D[T[1, 8, x, t], x] /. x -> L]
```

Out[ ]:= 0

Out[ ]:= 0.

Out[ ]:= 0

**Note:** Both examples are "unphysical" mathematical models in which no heat exchange with the environment is taken into account, which would play a role under real conditions.

### Example 5. A homogeneous heat equation with inhomogeneous boundary conditions

Consider the problem

$$\begin{aligned} \partial_t u[x, t] &= k \partial_{x,x} u[x, t], \quad u[x, 0] = f[x], \\ u[-\pi, t] + u[\pi, t] &= 2, \quad \partial_x u[-\pi, t] + \partial_x u[\pi, t] = 0 \text{ for } t \geq 0. \end{aligned}$$

We assume that  $k > 0$  and  $f$  is continuously differentiable in  $]-\pi, \pi[$  and [compatible with the boundary conditions](#), i.e.,  $f(-\pi) + f(\pi) = 2$  and there exist  $f'(\pm\pi)$  with  $f'(-\pi) + f'(\pi) = 0$ . For simplicity we set  $k=1$ .

This is a mathematical model for the evolution of the temperature  $u(x, t)$  at the point  $x$  at time  $t$  of a thin rod identified with  $[-\pi, \pi]$ . The boundary condition for  $u$  says that the mean temperature at the endpoints is kept at 1, while the boundary conditions for  $\partial_x u$  mean that the heat flux at the endpoints is equal in magnitude but with opposite sign, i.e., heat entering or leaving the rod at both ends at the same rate.

One can prove that the problem has a unique solution  $u \in C^1([-\pi, \pi] \times [0, \infty[) \cap C^2(]-\pi, \pi[ \times ]0, \infty[)$  (cf. [9]). The solution can be obtained with  $u[x, t] = v[x, t] + 1$ , where  $v$  solves the equation

$\partial_t v[x, t] = k \partial_{x,x} v[x, t]$ , so that  $v[x, 0] = f[x] - 1$ ,

$v[-\pi, t] + v[\pi, t] = 0$ ,  $\partial_x v[-\pi, t] + \partial_x v[\pi, t] = 0$  for  $x$  in  $[-\pi, \pi]$ ,  $t \geq 0$ .

With separation of the variables we obtain as before

```

In[ ]:= sol1 = DSolve[{V''[x] + c V[x] == 0}, V[x], x];
In[ ]:= V[x_] = sol1[[1, 1, 2]]
Out[ ]:= c1 Cos[√c x] + c2 Sin[√c x]
In[ ]:= Assuming[c ≥ 0, Solve[{V[-π] + V[π] == 0, V'[-π] + V'[π] == 0}, {c1, c2}]]
Out[ ]:= {{c1 → 0, c2 → 0}}
In[ ]:= V[-π] + V[π]
          V'[-π] + V'[π]
In[ ]:= 2 c1 Cos[√c π]
Out[ ]:= 2 c1 Cos[√c π]

```

Thus, imposing the boundary conditions we obtain the solution with negative  $c = -\lambda$ ,

$\lambda = -(n + 1/2)^2$ ,  $n \in \mathbb{N}_0$ .

Inserting into the second equation of the separation of variables approach  $w'[t] = \lambda w[t]$

```

In[ ]:= sol2 = DSolve[w'[t] + (n + 1/2)^2 w[t] == 0, w[t], t]
          solT[t] = sol2[[1, 1, 2]]
Out[ ]:= {{w[t] → e^(-t/4 - n t - n^2 t) c1}}
Out[ ]:= e^(-t/4 - n t - n^2 t) c1

```

we arrive at  $f(x) - 1 = v(x, 0) = \sum_{n=0}^{\infty} (A_n \cos(nx + x/2) + B_n \sin(nx + x/2))$ . We thus can **expand  $f(x)$  as a trigonometric series with the basis functions  $\cos(nx + x/2)$ ,  $\sin(nx + x/2)$ ,  $n \geq 0$ , which build also an orthogonal system in  $L^2([-\pi, \pi])$** .

We obtain, for example, with  $f(x) = (x/\pi)^2$  and  $u = v + 1$  an approximation  $u(x, t, m)$  of degree  $m$  as below

```

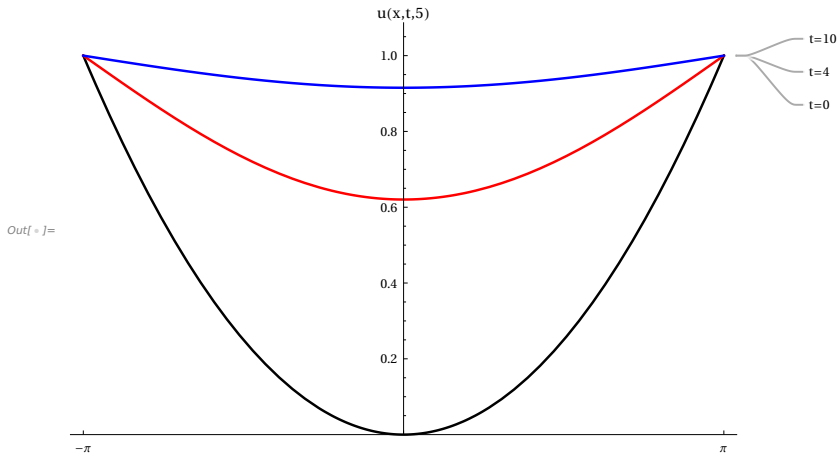
In[ ]:= f[x_] = (x/π)^2
          A[n_] = 1/π Integrate[(f[x] - 1) Cos[n x + x/2], {x, -π, π}]
Out[ ]:= x^2
          π^2
Out[ ]:= -16 × (2 Cos[n π] + (1 + 2 n) π Sin[n π])
          (1 + 2 n)^3 π^3
In[ ]:= B[n_] = 1/π Integrate[(f[x] - 1) Sin[n x + x/2], {x, -π, π}]
Out[ ]:= 0
In[ ]:= u[x_, t_, m_] := 1 + Sum[
          Exp[-t/4 - n t - n^2 t] (A[n] Cos[n x + x/2] + B[n] Sin[n x + x/2]), {n, 0, m}]
In[ ]:= u[x, t, 5]

```

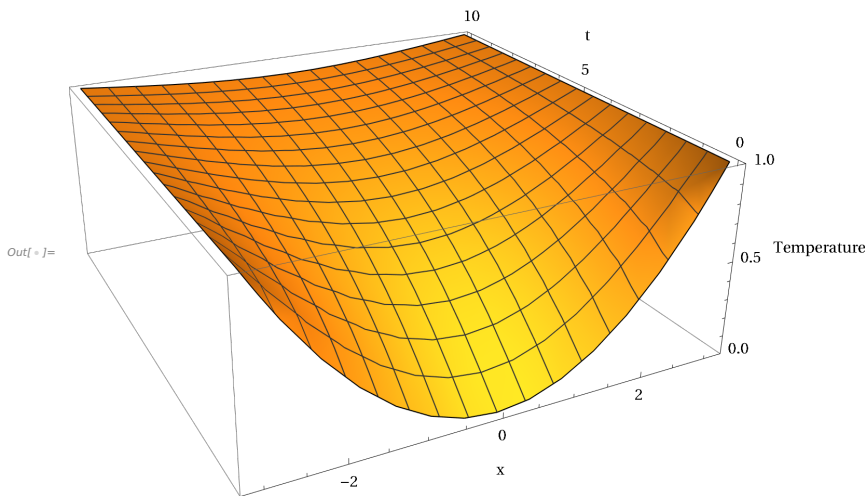
$$\begin{aligned} \text{Out}[*]:= & 1 - \frac{32 e^{-t/4} \cos\left[\frac{x}{2}\right]}{\pi^3} + \frac{32 e^{-9 t/4} \cos\left[\frac{3 x}{2}\right]}{27 \pi^3} - \frac{32 e^{-25 t/4} \cos\left[\frac{5 x}{2}\right]}{125 \pi^3} + \\ & \frac{32 e^{-49 t/4} \cos\left[\frac{7 x}{2}\right]}{343 \pi^3} - \frac{32 e^{-81 t/4} \cos\left[\frac{9 x}{2}\right]}{729 \pi^3} + \frac{32 e^{-121 t/4} \cos\left[\frac{11 x}{2}\right]}{1331 \pi^3} \end{aligned}$$

**Illustration of the solution**

`in[*]:= p1 = Plot[{f[x], u[x, 4, 5], u[x, 10, 5]}, {x, -Pi, Pi},  
 PlotLabel → "u(x,t,5)", PlotStyle → {Black, Red, Blue}, PlotRange → {0, 1},  
 PlotLabels → {"t=0", "t=4", "t=10"}, Ticks → {{-π, 0, π}, Automatic}]`



`in[*]:= p0 = Plot3D[u[x, t, 5], {x, -Pi, Pi}, {t, 0, 10},  
 PlotRange → {0, 1}, AxesLabel → {"x", "t", "Temperature"},  
 AxesStyle → Directive[Black, 10], ViewPoint → {-1, -2, 1}]`





### 2.3 Fourier Series in inhomogeneous 1D Heat Equations

**Example 6.** We now consider an inhomogeneous problem. For the equation

$$\partial_t U[x, t] - \alpha \partial_{x,x} U[x, t] = G[x, t]$$

with the right-hand side  $G$  we choose a heat flux density that is constant over time, so that

$G[x, t] = 0.6 (\text{UnitStep}[x - L/4] - \text{UnitStep}[x - 3L/4])$  (in  $C^0/s$ ). We also choose as initial condition  $U[x, 0] = 0$  for  $x$  in  $[0, L]$  and as last Neumann boundary conditions. The model describes a uniform heating of our copper rod around the center of the rod, which is otherwise thought to be perfectly insulated. (Temperature now denoted as  $U$ , so that - as long as the animation above is still running - there is no naming conflict).

We start with an approach using "variation of the constants", i.e., we use the solution approach

$U[x, t] = c_0[t] + \sum_{n=1}^{\infty} c_n[t] \cos[n\pi x/L]$ , insert this into the equation and use the boundary and initial conditions (please carry out for practice).

This results in  $c_n'[t] + (\alpha n^2 \pi^2 / L^2) c_n[t] = g_n$  for  $n > 0$ ,  $c_0[t] = g_0 t$ . The constants  $g_n$  denote the Fourier cosine coefficients of the inhomogeneity  $G$ ,  $g_0$  the mean value of  $G$ . Calculation for  $G[x, t]$  with  $L = 1$  m and  $\alpha := 117 \cdot 10^{-6}$  m<sup>2</sup>/s as above results in  $c_0[t] = g_0 t$  (from  $c_0'[t] = g_0$  and  $c_0[0] = 0$ ).

In[8]:=  $L = 1;$

$G[x_] = 6 / 10 (\text{UnitStep}[x - L/4] - \text{UnitStep}[x - 3/4 L])$

$g[n_] = 2/L \text{ Integrate}[G[x] \cos[n \pi x/L], \{x, 0, L\}]$

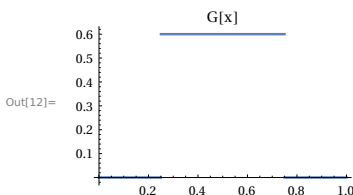
$g_0 = 1/L \text{ Integrate}[G[x], \{x, 0, L\}]$

$\text{Plot}[G[x], \{x, 0, L\}, \text{PlotLabel} \rightarrow "G[x]", \text{ImageSize} \rightarrow \text{Small}]$

Out[9]=  $\frac{3}{5} \left( -\text{UnitStep}\left[-\frac{3}{4} + x\right] + \text{UnitStep}\left[-\frac{1}{4} + x\right] \right)$

Out[10]=  $\frac{6 \left( -\sin\left[\frac{n\pi}{4}\right] + \sin\left[\frac{3n\pi}{4}\right] \right)}{5 n \pi}$

Out[11]=  $\frac{3}{10}$



In[13]:=  $\text{sol} := \text{DSolve}[$

$\{D[c[n, t], \{t, 1\}] + \alpha n^2 \pi^2 / L^2 c[n, t] = g[n], c[n, 0] == 0\}, c[n, t], \{n, t\}]$

$$\text{In[14]:= ccoeff}[n_, t_] := \text{sol}[1, 1, 2]$$

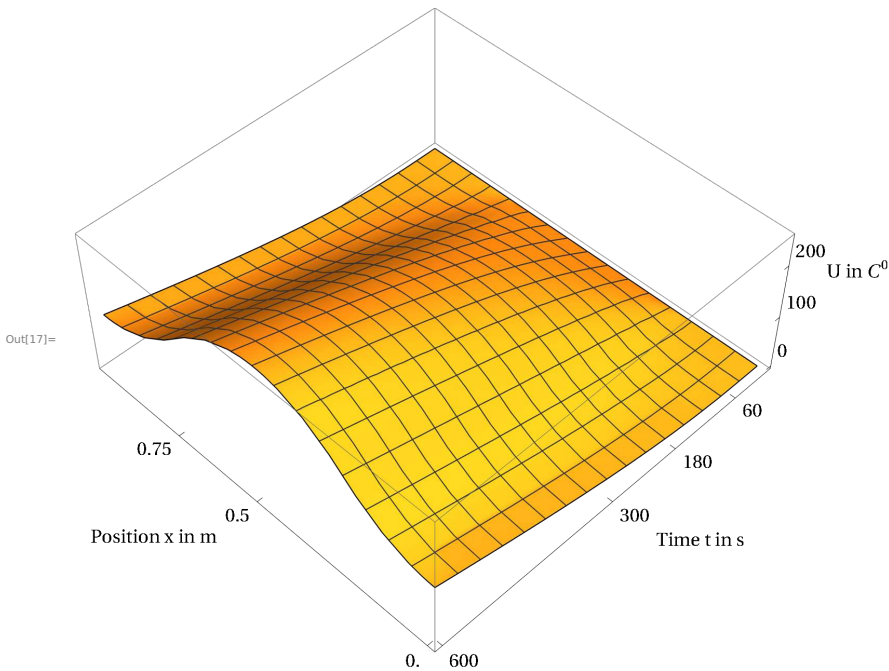
$$\text{Out[14]= } - \frac{6 e^{-n^2 \pi^2 t \alpha} \left( -1 + e^{n^2 \pi^2 t \alpha} \right) \left( \text{Sin}\left[\frac{n \pi}{4}\right] - \text{Sin}\left[\frac{3 n \pi}{4}\right] \right)}{5 n^3 \pi^3 \alpha}$$

We consider a partial sum of the Fourier series of the exact solution for the problem and plot it. Only every fourth Fourier coefficient is non - zero. In order to reproduce the step - like inhomogeneity well, we choose a higher order ( $m = 30$ ) of the trigonometric polynomial to approximate the heat flux density and a corresponding order of the trigonometric approximation polynomial for the solution. The "step form" of the initially inhomogeneity remains largely intact in the solution for quite some time before the heat balance takes effect.

```

In[15]:=  $\alpha = 117 \times 10^{-6}$ ;
U[m_, x_, t_] := g0 t + Sum[ccoeff[n, t] Cos[n Pi x / L], {n, 1, m}]
Plot3D[N[U[30, x, t]], {x, 0, L}, {t, 0, 600},
  PlotRange -> All, AxesLabel -> {"Position x in m ", " Time t in s ", "U in C° "},
  AxesStyle -> Directive[Black, 12], ViewPoint -> {-2, 2, 3},
  Ticks -> {{0.0, 0.5, 0.75}, {60, 180, 300, 600}, {0, 100, 200, 300}}]

```



Here again the temperature evolution as an animation.

A mathematically precise treatment of differential equations with discontinuous right-hand sides, such as  $G$  here, is possible within the framework of distribution theory (see[1]).

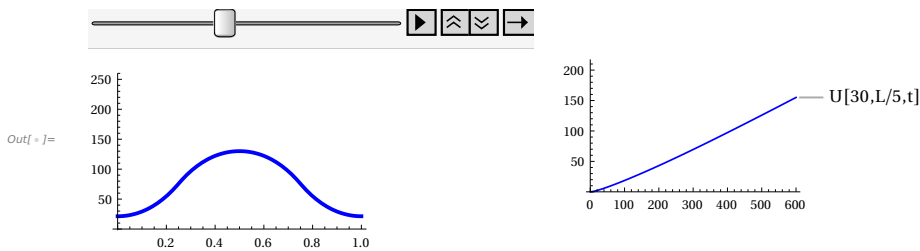
In the right graphics the temperature development is shown, smoothed by arithmetic averaging of the partial sums of the result at some distance from the directly heated interval  $[L/4, 3L/4]$ . We see an approximately linear increase in the temperature, which becomes stronger as you move closer to the interval that is heated (Test it yourself.) Here we look at

$x=L/5$ :

```

In[ ]:= a1 = Animate[Plot[N[U[30, x, t]], {x, 0, 1}, PlotRange → {-2, 260},
    PlotStyle → Directive[Blue, Thickness[0.015]], ImageSize → Small], {t, 0, 600},
    AnimationRepetitions → 1, AnimationRate → 5, RefreshRate → 50];
Usmoothed[m_, x_, t_] :=
    g0 t + Sum[ccoeff[n, t] (1 - n/(m + 1)) Cos[n Pi x/L], {n, 1, m}]
a2 = Plot[N[Usmoothed[30, L/5, t]], {t, 0, 600}, PlotRange → {0, 200},
    PlotStyle → Directive[Blue, Thickness[0.008]],
    PlotLabels → Style["U[30,L/5,t]", 12]];
GraphicsRow[{a1, a2}]

```



And the **final test** that everything is correct: The mean temperature after 600 s corresponds to the heat supplied corresponding to  $0.3\text{ }C^0/s$  on average for the whole rod with the assumed perfect insulation and the differential equation as well as the initial and the bound-ary conditions are fulfilled:

```

In[ ]:= 1/L Integrate[U[30, x, 600], {x, 0, L}]
test = Chop[Simplify[
     $\partial_t U[30, x, t] - \alpha \partial_{x,x} U[30, x, t] - (g_0 + \text{Sum}[g[n] \text{Cos}[n \text{Pi } x/L], \{n, 1, 30\}])$ ]]
D[U[30, x, t], x] /. x → 0
D[U[30, x, t], x] /. x → L

```

Out[ ]:= 180

Out[ ]:= 0

Out[ ]:= 0

Out[ ]:= 0

## 2.4 Fourier Series Solution for the Potential Equation on a Circular Disk

The potential equation on a circular disk of radius  $R>0$  is given in polar coordinates by

$$\Delta u = \frac{\partial^2}{\partial r^2} u + \frac{1}{r} \frac{\partial}{\partial r} u + \frac{1}{r^2} \frac{\partial^2}{\partial \phi^2} u \text{ for } 0 < r < R.$$

For a given boundary condition  $u(R, \phi) = U(\phi)$  with the Fourier coefficients  $c_k$  ( $0 \leq \phi < 2\pi$ ) the equation has the unique Fourier series solution  $u(r, \phi) = \sum_{k=-\infty}^{+\infty} c_k (r/R)^{|k|} e^{ik\phi}$ . The solution has no local extrema in the interior of the disk, it attains minimum and maximum at the boundary.

(For Details see [1], section 5.3).

**Example 7.** As an example, we illustrate the solution for the simple boundary condition  $U(\phi) = \cos(\phi) + 2 \sin(2\phi)$  for  $R=2$  with two possible representations.

**First a “conventional” view with a Cartesian x,y,z coordinate system.** The polar coordinates are transformed to Cartesian in the plot, i.e.,

$r = \text{Sqrt}[x^2 + y^2]$ ,  $\phi = \text{Arg}[x + I y]$ .

`In[ ] := R := 2;`

`u[r_, phi_] =`

`(r/R Cos[phi] + 2 (r/R)^2 Sin[2 phi]) (HeavisideTheta[r] - HeavisideTheta[r - R])`

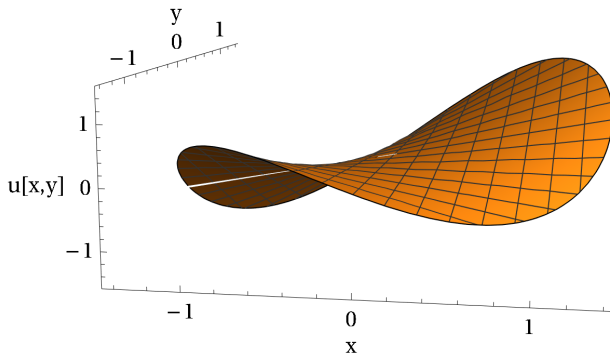
`Out[ ] := (-HeavisideTheta[-2 + r] + HeavisideTheta[r]) (1/2 r Cos[phi] + 1/2 r^2 Sin[2 phi])`

`In[ ] := Plot3D[u[Sqrt[x^2 + y^2], Arg[x + I y]], {x, -2, 2}, {y, -2.1, 2.1},`

`PlotRange -> All, Boxed -> False, PlotStyle -> Directive[Normal], Axes -> True,`

`AxesLabel -> {"x", "y", "u[x,y]"}, AxesStyle -> Directive[Black, Plain, 14],`

`RegionFunction -> Function[{x, y, z}, x^2 + y^2 ≤ R], ViewPoint -> {0.5, -3, 0.5}]`

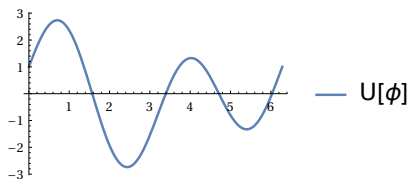
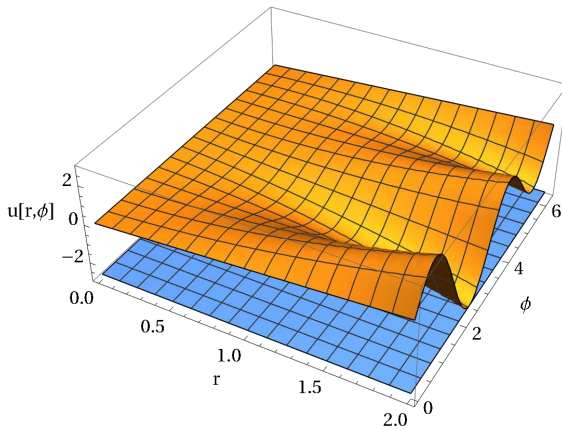


Now the same **illustrated with a rectangular r,φ coordinate system**, which better shows the boundary function for  $r=R=2$  and that there are no local extremal values of  $u$  in the interior of the circular disk. You can change the view and size by dragging the image with your mouse.

```

In[ ]:= p1 = Plot[Cos[φ] + 2 Sin[2 φ], {φ, 0, 2 Pi},
  ImageSize → Small, PlotLegends → {"U[φ]"}];
min = FindMinimum [Cos[φ] + 2 Sin[2 φ], φ];
p2 = Plot3D[{u[r, phi], min}, {r, 0, R}, {phi, 0, 2 Pi }, PlotRange → {-3, 3},
  AxesLabel → {"r", "φ", "u[r,φ]"}, AxesStyle → Directive [Black, Plain, 12]];
Show[p2]
Show[p1]

```



If the equation describes physically a Dirichlet problem for heat conduction with a given time-independent temperature  $U(\phi)$  on the boundary, then the solution is the stationary temperature of the disk generated by the boundary temperature.

## 2.5 An Initial Boundary Value Problem for a Force-Free Vibrating String

In the following we show the solution for a force-free vibrating string of length  $L$  according to the equation

$$\frac{\partial^2}{\partial t^2} u = c^2 \frac{\partial^2}{\partial x^2} u$$

with initial condition  $u(x,0)=f(x)$  and boundary conditions  $u(0,t)=u(L,t)=0$  for all  $t$ .

The solution  $u(x,t)$  is the transversal displacement of the string at  $x$  and time  $t$ . The constant  $c$  is the velocity of the wave.

By separation of the variables one can find the unique solution in the form of a Fourier series provided the oddly periodically extended initial condition  $f(x)=\sum_{n=1}^{\infty} a_n \sin(n\pi x/L)$  is twice continuously differentiable on the entire axis  $\mathbb{R}$  and  $f''$  is piecewise continuous. For the derivation of this, please see [1], 1.2 and 5.4.

The Fourier series of the solution is then

$$u(x,t) = \sum_{n=1}^{\infty} a_n \sin(n\pi x/L) \cos(cn\pi t/L),$$

where the constants  $a_n$  are the Fourier sine coefficients of  $f$ .

By the trigonometric addition theorems D'Alembert's representation of the solution can be obtained as (see again [1], 5.4) as

$$u(x, t) = \frac{1}{2} (f(x-ct) + f(x+ct)).$$

Thus, the solution is a superposition of two waves with the shape of  $f$ , which move in opposite directions with velocity  $c$  and are reflected at the string ends with opposite phase.

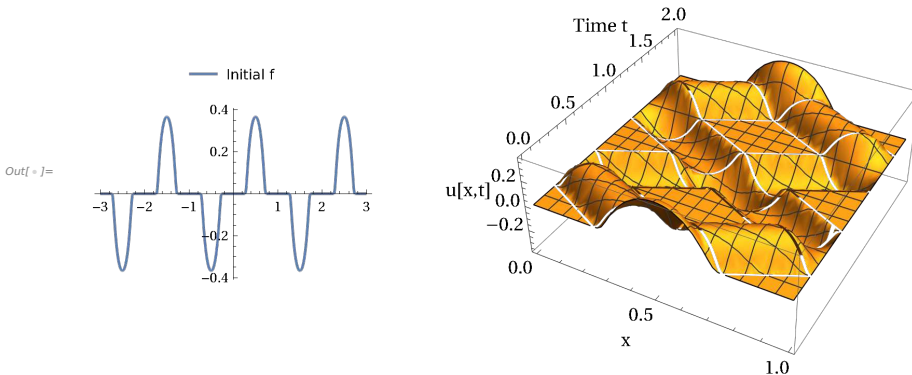
### Example 8. We illustrate the solution for a smooth initial condition $f$

First we set up the initial condition as function on the line from  $-3L$  to  $3L$ . Then we extend it to an odd  $6L$ -periodic function on the line. We plot the solution for  $x$  and  $t$  over 2 time units showing the superposition and reflection at the string ends. We use D'Alembert's representation, because it is hardly possible to calculate  $f$  and  $u$  as Fourier series.

```

In[ ]:= Clear["Global`*"];
h[x_] = Piecewise[{{Exp[-1/(1-x^2)], x < 1 && x > -1}, {0, x ≥ 1 && x ≤ -1}}];
f1[x_] = h[4 x - 2]; L := 1;
f[x_] = f1[x] - f1[x + 1] + f1[x + 2] - f1[x + 3] - f1[x - 1] + f1[x - 2];
(* this is an odd extension from -3l to 3L *)
ic = Plot[f[x], {x, -3 L, 3 L}, PlotRange → All,
  PlotLegends → Placed[{"Initial f"}, Above], ImageSize → Small];
c = 1;
u[x_, t_] = 1/2 (f[x - c t] + f[x + c t]);
sol = Plot3D[u[x, t], {x, 0, L}, {t, 0, 2}, PlotRange → All, AxesLabel →
  {"x", "Time t", "u[x,t]"}, AxesStyle → Directive[Black, Plain, 12]];
GraphicsRow [
  {ic,
   sol}]

```



**Problem:** Since it is not that easy to provide smooth initial conditions and their Fourier series expansions, it is desirable to work with simple mathematical models as initial conditions and later as right hand sides for inhomogeneous equations. We show an example with a piecewise linear function  $f$  and will readily recognize that one does not obtain solutions in the classical sense, because these are no more differentiable functions. The question then is, what they mean as solutions of second order differentiable equations. These questions could not be solved until the mathematical progress from classical theory to *distribution theory*.

Distribution theory was developed about 1935 by S. L. Sobolev (1908-1989), in the years 1945-1950 by L. Schwartz (1915-2002) and others. There, the concept of *generalized derivatives* and of *generalized solutions* is introduced, also called weak solutions for differential equations. This allows much easier calculations and has opened up a myriad of applications in mathematics and all other scientific areas.

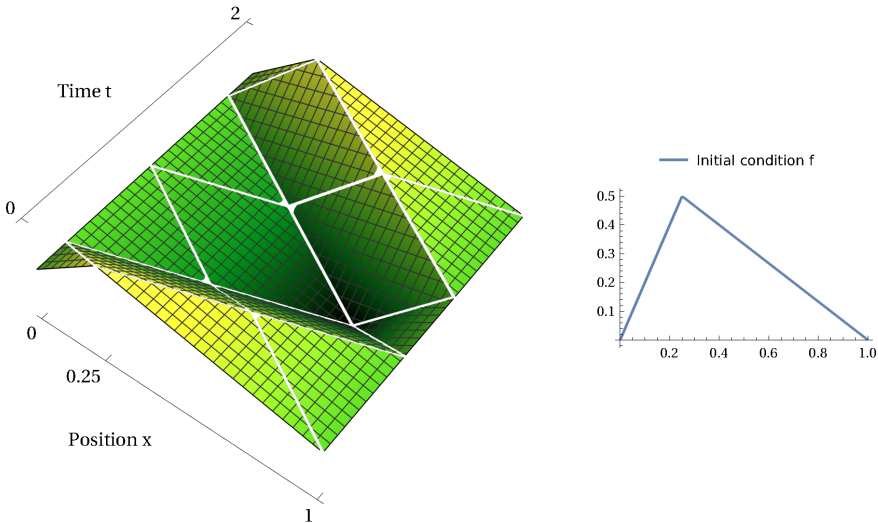
We will discuss this in a subsequent booklet. Here, we illustrate the problem with a piecewise linear initial condition, which could be a simple model of an initially deflected string, which is the released.

**Example 9. A simple model for an initially deflected string and the “solution” of the initial boundary value problem**

```

Clear[f, f1, x, t];
f1[x_] := 2 x (UnitStep[x + 1/4] - UnitStep[x - 1/4]) +
  2 x (1 - x)/3 (UnitStep[x - 1/4] - UnitStep[x - 1]) -
  2/3 (x + 1) (UnitStep[x + 1] - UnitStep[x + 1/4]);
pf1 = Plot[f1[x], {x, 0, 1}, PlotLegends -> Placed[{"Initial condition f"}, Above],
  PlotRange -> All, ImageSize -> Small];
f[x_] := f1[x + 2] + f1[x] + f1[x - 2]; (* extension as periodization *)
string[x_, t_] := 1/2 (f[x + t] + f[x - t]); (* D'Alembert's solution form *)
pf2 = Plot3D[string[x, t], {x, 0, 1}, {t, 0, 2}, Mesh -> 30,
  ColorFunction -> "AvocadoColors", Axes -> {True, True, False},
  Ticks -> {{0, 0.25, 1}, {0, 2}}, Boxed -> False,
  AxesLabel -> {"Position x", "Time t"}, ViewPoint -> {1.5, -2, 3},
  AxesStyle -> Directive[Black, Plain, 12]];
GraphicsRow[{pf2, pf1}]

```



You clearly observe that **this solution is not differentiable** and thus cannot be a solution in the classical sense. We come back to the problem, once we have seen how the theory of distributions can overcome these problems in classical theory. Distribution theory was a major step in Applied Mathematics solving linear problems, that had existed for centuries. Distribution theory offers new methods with algorithms for solving numerically uncountable problems in science and engineering.



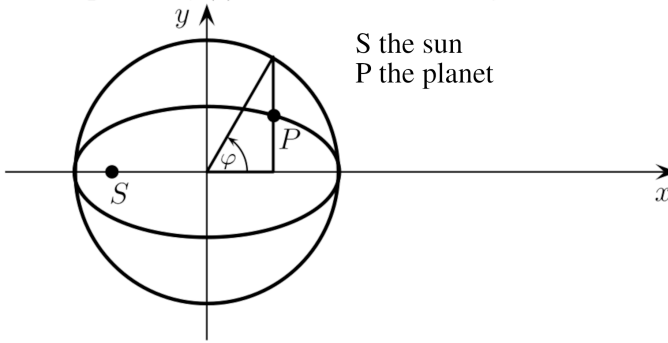
## 2.6 Solution of a Kepler Equation by Fourier Series Expansion

Kepler's equation for the eccentric anomaly

Kepler's equation (J. Kepler 1571-1630) for the elliptic orbit of a planet P is

$$\phi(t) - \epsilon \sin(\phi(t)) = \omega t \quad (1).$$

Here,  $\omega = 2\pi/T$  is the angular frequency with orbital period  $T$ ,  $0 \leq \epsilon < 1$  the eccentricity of the ellipse and  $\phi(t)$  the eccentric anomaly at time  $t$  (see Fig. below).



For all  $t \in \mathbb{R}$  the following is valid:  $\frac{d\phi}{dt}(t) = \omega/(1 - \epsilon \cos(\phi(t))) > 0$ . Furthermore,  $\phi(0)=0$  and  $\phi(T)=2\pi$ . Therefore,  $\phi(t)$  is monotonically increasing with  $t$  and  $\sin(\phi(t))$  must be an odd function of  $t$  due to the motion's symmetry. This motivates the solution approach  $\phi(t) = \omega t + \sum_{k=1}^{\infty} b_k \sin(k\omega t)$ . This solution by Fourier series expansion goes back to J. L. Lagrange and F. W. Bessel. To obtain the coefficients  $b_k$  we transform the according integral:

$$b_k = \frac{4}{T} \int_0^{T/2} \epsilon \sin(\phi) \sin(n\omega t) dt$$

$$= - \frac{4\pi \sin(\phi) \cos(n\omega t)}{n\omega T} \Big|_0^{T/2} + \frac{4}{n\omega T} \int_0^{T/2} \cos(n\omega t) \frac{d}{dt} (\epsilon \sin(\phi)) dt$$

through integration by parts. Thus, with  $\phi(T/2)=\pi$  and differentiation of (1) above and with substitution  $\phi=\phi(t)$

$$b_k = \frac{4}{n\omega T} \int_0^{T/2} (\dot{\phi} - \omega) \cos(n\omega t) dt = \frac{2}{n\pi} \int_0^{\pi} \cos(n\omega t) d\phi$$

$$= \frac{2}{n\pi} \int_0^{\pi} \cos(n(\phi - \epsilon \sin(\phi))) d\phi = \frac{2}{n} J_n(n\epsilon)$$

Here  $J_n$  is the Bessel function of the first kind implemented in Mathematica as `Bessel[n,z]`. It can be evaluated to arbitrary numerical precision.

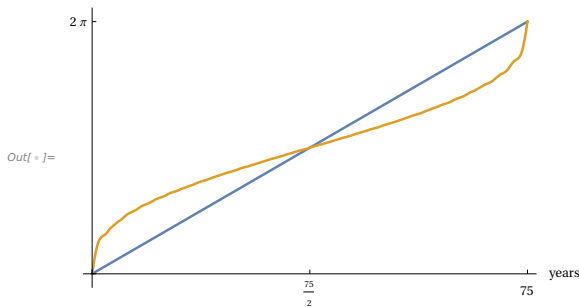
**Example 10. Eccentricity and orbit of Halley's Comet**

We show a plot of a  $\phi(t)$  with the eccentricity  $\epsilon=0.967$  of Halley's Comet by a partial sum of the computed Fourier series.

```

In[ ]:= eps = 0.967;
T = 75;
 $\omega = 2 \text{ Pi} / T$ ;
approxKeppler [t_] =  $\omega t + \text{Sum}[2 / n \text{ BesselJ}[n, n \text{ eps}] \text{Sin}[n \omega t], \{n, 1, 30\}]$ ;
Plot[{ $\omega t$ , approxKeppler [t]}, {t, 0, 75}, Axes → True,
  AxesLabel → {"years", None}, Ticks → {{0, T/2, T}, {0, 2 Pi}}]

```



On the following page the corresponding orbit of Halley's Comet with its astronomical data and the approximation from above. A picture of the comet presented on NSSDC's Photo Gallery, NASA ID LSPN-1725 is also shown at the cover page of this title.

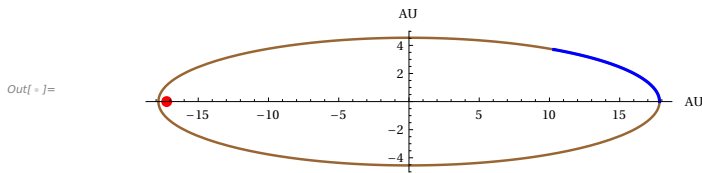
```

In[ ]:= eps = 0.967;
a = 17.834;
b = a Sqrt[1 - eps ^ 2];
y[x_] = b / a Sqrt[a ^ 2 - x ^ 2];
e = Sqrt[a ^ 2 - b ^ 2]; (*astronomical data*)
xpos[t_] = a Cos[approxKeppler [t]];
ypos[t_] = b Sin[approxKeppler [t]]; (* position according to the approximation
  with the above Fourier partial sum with the Bessel functions *)
p1 = Plot[{y[x], -y[x]}, {x, -18, 18}, AspectRatio → Automatic, PlotStyle →
  {Brown, Brown}, PlotLegends → Placed[{"Orbit of Halley's Comet,
  in blue from t=0 to t=2 years starting from aphelion ,
  AU about 150x10^6 km, the red focus is the sun"}, Above],
  Axes → True, AxesLabel → {AU, AU}];
p2 = Graphics [{PointSize [Large], Red, Point[{-e, 0}]}];
(*plot of the focus *) p3 = ParametricPlot [{xpos[t], ypos[t]},
  {t, 0, 2}, PlotStyle → {Blue, Thick}, AspectRatio → Automatic ];
(* orbit within 2 years starting from aphelion *)

```

`In[ ]:= Show[p1, p2, p3]`

- Orbit of Halley's Comet,  
in blue from  $t=0$  to  $t=2$  years starting from aphelion ,  
AU about  $150 \times 10^6$  km, the red focus is the sun



Finally the photography of Comet 1P/Halley as taken March 8, 1986 by W. Liller, Easter Island, part of the International Halley Watch (IHW) Large Scale Phenomena Network, NSSD-C's Photo Gallery, NASA ID LSPN-1725.



Further examples for Fourier series in linear differential equations can be found in the exercises and their solutions in [1], chapter 5, and in [10].

## 2.7 Solving a 2D Poisson Equation for a Rectangular Membrane by a Ritz-Galerkin Solution with Trigonometric Functions

In this section we use Fourier series expansions to calculate approximately the deformation of a loaded rectangular membrane with zero boundary conditions.

In the example, we compute a **Ritz-Galerkin-Solution**, which is based on the following theoretical setting, where the task is understood as a variational problem leading to a so-called **weak solution**. We start with the formulation in terms of Hilbert spaces  $V$ , i.e., complete function spaces with a norm  $\|f\| = \langle f|f \rangle^{1/2}$ ,  $\langle \cdot, \cdot \rangle$  an inner product,  $f \in V$ . This formulation is also the starting point for the widely used Finite Element Method (FEM), on which we learn more in the subsequent volume on Fourier Transforms and Distributions. For details see [1], chapter 9.

### Equilibrium State of a Loaded Membrane

Consider a bounded domain  $\Omega$  in the plane with a piecewise linear boundary  $\delta\Omega$ , where an elastic membrane is fixed. Under the influence of an external force acting perpendicular to the plane, the membrane deflects. The tension due to the fixing is isotropic, so it is described by a scalar quantity  $k$  (with the dimension N/m). If  $f$  denotes the surface density of the force, then for small displacements  $u$  in the equilibrium state it holds

$$-k \Delta u = f \text{ in } \Omega, u = 0 \text{ on } \delta\Omega.$$

Here,  $\Delta$  denotes the Laplace operator. Thus, the equilibrium position is the solution of a Dirichlet boundary value problem. The following considerations can also be translated to electrical potential problems or stationary heat conduction problems. A derivation of the above fact from Hooke's law can be found in works such as [5] R. Courant, D. Hilbert (1993) et al.

If the boundary  $\delta\Omega$  has a complicated shape, it will not be possible to calculate a solution using the classical analytical methods. However, a practical solution approach is opened up by distributional considerations. The equation  $-\Delta u = f$  is interpreted as an equation between distributions, i.e., one seeks a function  $u$  in a suitable Hilbert space  $V$ , so that - using Cartesian coordinates - for all functions  $v$  in  $V$  it holds

$$-\Delta u, v = k \int_{\Omega} \text{grad } u(x,y) \cdot \text{grad } v(x,y) \, d(x,y) = \langle f, v \rangle.$$

The first equality follows from the definition of the generalized derivatives of  $u$  and  $v$  in  $V$  (cf. [1], p. 173). Since one now has to solve a boundary value problem, one seeks a solution  $u$  that is regular and also allows for speaking about boundary values  $u=0$  on  $\delta\Omega$ . According to **S. L. Sobolev** (1908 - 1989), one seeks the solution  $u$  among those **functions  $v$  that are square - integrable along with their partial generalized derivatives on  $\Omega$  and vanish on the boundary  $\delta\Omega$** . The set of all such functions  $v$  forms a function vector space  $V$  over  $\mathbb{R}$ , which is denoted as  $V = H_0^1(\Omega)$ .

Even for complicated domains whose boundary has only minimal regularity properties, this vector space can be introduced in such a way that it is possible to meaningfully speak of boundary values of its elements. This is assumed for  $\Omega$  and  $V$  in the following. Two functions in  $V$  are identified if they differ only on a null set. The space  $V$  is an example of a function vector space called a Sobolev space. More generally, Sobolev spaces are vector spaces of regular distributions whose partial derivatives up to a certain order are also regular. Details about Sobolev spaces and their applications in partial differential equations can be found, for example, in [6] R. Dautray, J. L. Lions (1992). The needed properties of  $V = H_0^1(\Omega)$  in the following are to be found in Appendix B of [1]. The basis for statements on the solvability of the given problem and also for the construction of numerical approximate solutions in  $V$  using later on (see [1] or a subsequent volume of this booklet) the Finite Element Method is then the following formulation of the problem:

### Formulation of the problem in the Sobolev space $V$

The force density  $f$  is assumed to be square - integrable on  $\Omega$ , and  $V$  is the above described Sobolev space. Here, in the example we consider a rectangular domain  $\Omega$ . We seek a function  $u$  in  $V$ , such that for all  $v$  in  $V$  the following holds:

$$a(u,v) = l(v)$$

$$a(u,v) = k \int_{\Omega} \text{grad } u(x,y) \cdot \text{grad } v(x,y) \, d(x,y) = k \int_{\Omega} \left( \frac{\partial}{\partial x} u \frac{\partial}{\partial x} v + \frac{\partial}{\partial y} u \frac{\partial}{\partial y} v \right) d(x,y)$$

$$l(v) = \langle f, v \rangle = \int_{\Omega} f(x,y) v(x,y) \, d(x,y)$$

Due to the assumptions,  $a(u,v)$  and  $l(v)$  are well-defined for all  $u, v$  in  $V$ . The derivatives involved are to be understood as generalized derivatives. The boundary condition is included in the problem formulation by seeking the solution  $u$  in the vector space  $V$ , whose elements are functions that vanish on the boundary  $\delta\Omega$ .

The solution  $u$  is—if it exists—to be understood as a distributional solution and is also called a weak solution.

### Potential Energy and Energy Functional of the Membrane

The current task is closely related to the physical consideration that the equilibrium state of the membrane adjusts so that the total potential energy is minimal. Assuming a linear elastic material behavior according to Hooke's law, the deformation energy is proportional to the change in area.

The total potential energy  $E(v)$  of the membrane is then given for a displacement  $v$  by

$$E(v) = k \left( \int_{\Omega} (1 + |\text{grad } v(x,y)|^2)^{1/2} d(x,y) - \int_{\Omega} d(x,y) - \int_{\Omega} f(x,y) v(x,y) d(x,y) \right)$$

For small displacements, one obtains with

$$(1 + |\text{grad } v(x,y)|^2)^{1/2} - 1 \approx \frac{1}{2} |\text{grad } v(x,y)|^2$$

the approximation

$$E(v) \approx J(v) = \frac{1}{2} a(v,v) - l(v).$$

**The functional J is called the energy functional of the membrane** . If there is a function  $u$  for which  $J(u)$  is minimal, then  $u$  approximately describes the equilibrium position of the membrane. The connection of the posed boundary value problem with the variational problem of minimizing the functional  $J$  is established by the following version of a theorem by P. Lax and A. Milgram (see, for example, [6] R. Dautray, J. L. Lions (1992)). The theorem shows that both problems have a common solution in the Sobolev space  $V$ .

#### Theorem of Lax - Milgram

1. For a function  $u$  in  $V$ , the equation  $a(u,v)=l(v)$  holds for all  $v$  in  $V$  if and only if  $J(u) = \inf \{ J(v) \mid v \in V \}$ , i.e., if  $u$  minimizes the energy functional  $J$ .
2. Under the given conditions the energy functional  $J$  is bounded below on  $V$ , and there is a uniquely determined function  $u$  in  $V$  that minimizes  $J$ . This function  $u$  is thus also the desired distributional solution of the given boundary value problem.

This result teaches us that not only our exemplary problem, but also other problems of the same type can be solved in the same way.

Many boundary value problems can be formulated such that one seeks a function  $u$  in a function space  $V$  adapted to the respective task, so that an equation of the form  $a(u,v)=l(v)$  holds for all  $v$  in  $V$ . The statements of the theorem then also apply to all such problems for which the essential properties of the vector space  $V$  and for the (problem-dependent) functionals  $a$  and  $l$  are satisfied.

#### The Ritz-Galerkin Method

With the work done so far, we have learned how to formulate our boundary value problems, and that its (weak) solution is to be sought in a vector space  $V$  that has the inner product  $a(u,v)$  for  $u, v$  in  $V$  and the energy norm  $\|u\|_a = \sqrt{a(u, u)}$ . This now makes it easy to describe the basics of approximation methods according to Ritz and Galerkin and later as a special case the finite element principle.

In all vector spaces  $V$  where a norm  $\|f\|$  of elements  $f \in V$  is given by an inner product, one obtains an approximation in a subspace  $U$  of  $V$  by orthogonal projection of  $f$  onto  $U$ . The concept of orthogonality is directly related to the inner product:  $f, g$  from  $V$  are orthogonal if and only if their inner product is zero. The orthogonal projection  $f_U$  of  $f$  onto  $U$  is an optimal approximation for  $f \in V$  by an element of  $U$  in the following sense:

$$\|f - f_U\| = \inf \{ \|f - g\| : g \in U \}$$

i.e., the norm of the error  $\|f - g\|$  is minimal among all  $g \in U$  for  $g = f_U$ .

The exemplary problem now has an (unknown) solution  $u$  in the infinitely-dimensional function vector space  $V$ . In this vector space  $V$ , the bilinear form  $a(u,v)$  belonging to the problem defines an inner product and the norm  $\|\cdot\|_a$  for all  $u, v \in V$ . According to Ritz-Galerkin, one constructs a finite-dimensional subspace of  $V_N$  of  $V$  and calculates the orthogonal projection  $u_N$  of  $u$  onto  $V_N$  with the inner product given by  $a(u,v)$  as an approximation for the sought solution of the posed boundary value problem. Even if the

function  $u$  remains unknown, its orthogonal projection  $u_N$  can be determined from the specification of  $V_N$  and from the equation  $a(u,v)=l(v)$  valid for every  $v \in V$ . The function  $u_N$  is called the **Ritz-Galerkin solution belonging to  $V_N$** . The choice of  $V_N$  and hence how well a function  $u \in V$  can be approximated by functions from  $V_N$  is crucial for the error  $\|u - u_N\|_a$  of the approximation.

To achieve satisfactory numerical results, the specification of the subspace  $V_N$  and its approximation properties is the key to the construction of approximate solutions.

### The Linear System of Equations for a Ritz-Galerkin Solution

By specifying  $N$  linearly independent functions  $v_1, v_2, \dots, v_N$  in  $V$ , a basis of an  $N$ -dimensional subspace  $V_N$  of  $V$  is determined. The space  $V_N$  is the set of all linear combinations of the  $v_k$ ,  $1 \leq k \leq N$ . Thus, the Ritz-Galerkin solution  $u_N$  in  $V_N$  has a representation of the form

$$u_N = \sum_{k=1}^N u_{N,k} v_k$$

with uniquely determined real coefficients  $u_{N,k}$ . The orthogonality relations  $a(u - u_N, v_i) = 0$  and the equations  $a(u, v_i) = l(v_i)$  yield  $a(u_N, v_i) = l(v_i)$  for  $1 \leq i \leq N$ .

With the linearity of the bilinear form  $a$  and the above representation of  $u_N$ , one obtains the linear system of equations

$$\sum_{k=1}^N u_{N,k} a(v_k, v_i) = l(v_i)$$

for the sought coefficients  $u_{N,1}, \dots, u_{N,N}$ . In matrix form, with column vectors  $\mathbf{u}$  and  $\mathbf{l}$ , the task is thus

**Task.** Determine  $\mathbf{u} \in \mathbb{R}^N$ , so that  $\mathbf{A}\mathbf{u} = \mathbf{l}$  is satisfied for

$$\begin{aligned} \mathbf{A} &= (\alpha_{i,k}), \quad \alpha_{i,k} = a(v_k, v_i), \\ \mathbf{l} &= (l_i), \quad l_i = l(v_i) \quad \text{for } 1 \leq i \leq N, 1 \leq k \leq N \end{aligned}$$

The quantities  $\alpha_{i,k}$  and  $l_i$  can be calculated from the given functionals  $a$  and  $l$  and the chosen basis functions  $v_i$ . The matrix  $\mathbf{A}$  is symmetric and positive definite, particularly regular (cf. [1] for details).

In elasticity problems,  $\mathbf{A}$  is called the **stiffness matrix**. The uniquely determined solution  $\mathbf{u} = (u_{N,1}, \dots, u_{N,N})$  of the system of equations yields the desired approximate solution  $u_N$  of the original problem  $a(u,v) = l(v)$  for elements  $v$  in  $V$ .

### Example 11. A Ritz-Galerkin Solution with Trigonometric Function

We compute with Mathematica the Ritz-Galerkin solution in the described Sobolev space  $V = H_0^1(\Omega)$  for the Dirichlet problem on the rectangle  $\Omega = ]0, L[ \times ]0, L[$ .

As basis for a 4-dimensional subspace  $V_4$  of  $V$  we choose the trigonometric functions

$$\begin{aligned} v_1(x,y) &= L \sin(\pi x/L) \sin(\pi y/L), & v_2(x,y) &= L \sin(3\pi x/L) \sin(\pi y/L), \\ v_3(x,y) &= L \sin(\pi x/L) \sin(3\pi y/L), & v_4(x,y) &= L \sin(3\pi x/L) \sin(3\pi y/L). \end{aligned}$$

We calculate the solution of the linear equation system. We need only the diagonal elements of the matrix **A**, because all off-diagonal elements are zero.

$$\begin{aligned} \text{In[*]} := & \text{v1} = L \sin[\pi x / L] \sin[\pi y / L]; \text{v2} = L \sin[3 \pi x / L] \sin[\pi y / L]; \\ & \text{v3} = L \sin[\pi x / L] \sin[3 \pi y / L]; \text{v4} = L \sin[3 \pi x / L] \sin[3 \pi y / L]; \\ & \text{a11} = k \text{Integrate}[\text{Grad}[\text{v1}, \{x, y\}].\text{Grad}[\text{v1}, \{x, y\}], \{x, 0, L\}, \{y, 0, L\}] \\ & \text{a22} = k \text{Integrate}[\text{Grad}[\text{v2}, \{x, y\}].\text{Grad}[\text{v2}, \{x, y\}], \{x, 0, L\}, \{y, 0, L\}] \\ & \text{a33} = k \text{Integrate}[\text{Grad}[\text{v3}, \{x, y\}].\text{Grad}[\text{v3}, \{x, y\}], \{x, 0, L\}, \{y, 0, L\}] \\ & \text{a44} = k \text{Integrate}[\text{Grad}[\text{v4}, \{x, y\}].\text{Grad}[\text{v4}, \{x, y\}], \{x, 0, L\}, \{y, 0, L\}] \end{aligned}$$

$$\text{Out[*]} = \pi^2$$

$$\text{Out[*]} = 5 \pi^2$$

$$\text{Out[*]} = 5 \pi^2$$

$$\text{Out[*]} = 9 \pi^2$$

$$\begin{aligned} \text{In[*]} := & \text{l1} = f \text{Integrate}[\text{v1}, \{x, 0, L\}, \{y, 0, L\}] \\ & \text{l2} = f \text{Integrate}[\text{v2}, \{x, 0, L\}, \{y, 0, L\}] \\ & \text{l3} = f \text{Integrate}[\text{v3}, \{x, 0, L\}, \{y, 0, L\}] \\ & \text{l4} = f \text{Integrate}[\text{v4}, \{x, 0, L\}, \{y, 0, L\}] \end{aligned}$$

$$\text{Out[*]} = \frac{4}{\pi^2}$$

$$\text{Out[*]} = \frac{4}{3 \pi^2}$$

$$\text{Out[*]} = \frac{4}{3 \pi^2}$$

$$\text{Out[*]} = \frac{4}{9 \pi^2}$$

We set  $L = 1\text{m}$ ,  $f = 1 \text{ N/m}^2$ ,  $k = 2 \text{ N/m}$  and obtain the Ritz-Galerkin solution **u4** of the given Dirichlet boundary value problem, and compute the deflection at the point  $(L/2, L/2)$ :

$$\text{In[*]} := L = 1; f = 1; k = 2;$$

$$\text{u4}[x_, y_] = \text{l1} / \text{a11} \text{v1} + \text{l2} / \text{a22} \text{v2} + \text{l3} / \text{a33} \text{v3} + \text{l4} / \text{a44} \text{v4}$$

$$\text{Out[*]} = \frac{4 \sin[\pi x] \sin[\pi y]}{\pi^4} + \frac{4 \sin[3 \pi x] \sin[\pi y]}{15 \pi^4} + \frac{4 \sin[\pi x] \sin[3 \pi y]}{15 \pi^4} + \frac{4 \sin[3 \pi x] \sin[3 \pi y]}{81 \pi^4}$$

$$\text{In[*]} := \text{u4}[L/2., L/2.] (* \text{ deflection at } \{L/2, L/2\} \text{ in m} *)$$

$$\text{Out[*]} = 0.0360957$$

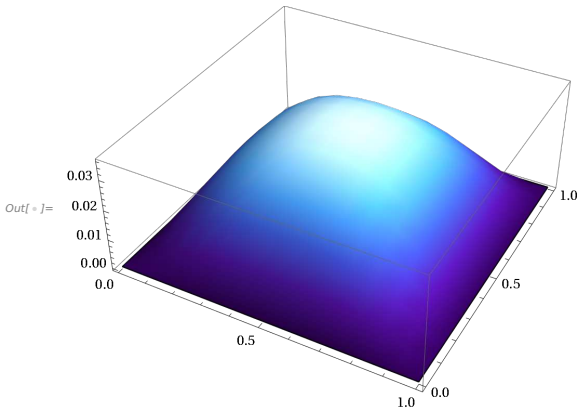
We can finally illustrate this weak solution by a 3D-Plot.



```

In[ ]:= Plot3D[u4[x, y], {x, 0, 1}, {y, 0, 1}, PlotRange -> All,
  Mesh -> None, ColorFunction -> "DeepSeaColors "]

```



Since Mathematica has numerical methods implemented, we can compare our Ritz-Galerkin solution with such an implemented method.

We define the given Dirichlet problem with the boundary condition and let Mathematica solve the problem numerically.

```

In[ ]:=  $\Omega$  = ImplicitRegion [0 < x < L & 0 < y < L, {{x, 0, L}, {y, 0, L}}];

```

```

In[ ]:= ImplicitRegion [0 < x < 1 && 0 < y < 1 && 0 ≤ x ≤ 1 && 0 ≤ y ≤ 1, {x, y}];

```

```

In[ ]:= op = -k Laplacian[solu[x, y], {x, y}] - f;

```

```

In[ ]:=  $\Gamma$  = {DirichletCondition[solu[x, y] == 0, True]};

```

```

In[ ]:= {DirichletCondition[solu[x, y] == 0, True]};



```

```

In[ ]:= solution = NDSolveValue[{op == 0,  $\Gamma$ }, solu, {x, y} ∈  $\Omega$ ]

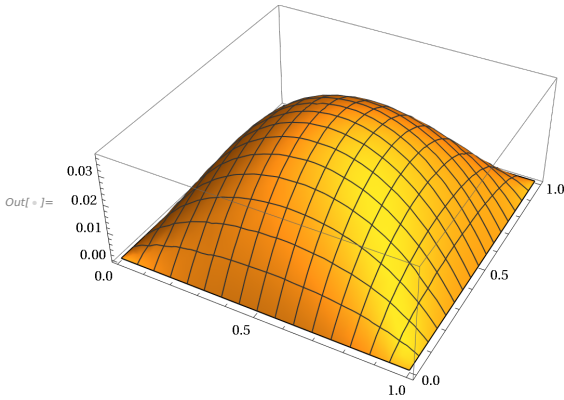
```

```

Out[ ]:= InterpolatingFunction [
  

  Domain: {{4.19 × 10-31, 1.}, {4.19 × 10-31, 1.}}
  Output: scalar
]

```

```
In[ ]:= Plot3D[solution[x, y], {x, y} ∈ Ω, PlotLegends → Automatic]
```



We finally check the difference between the numerical Mathematica solution and our above Ritz-Galerkin solution with only 4 trigonometric functions in the approximation. It is of order  $10^{-4}$ .

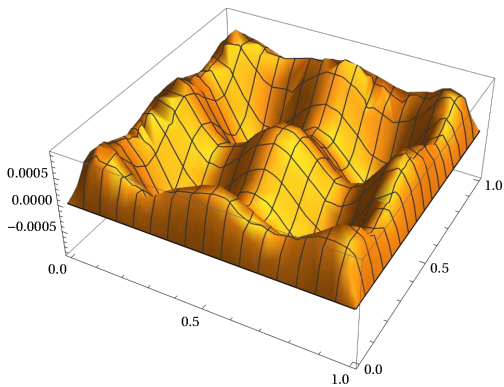
```
In[ ]:= solution[L/2, L/2]
```

```
Out[ ]:= 0.0368355
```

```
Plot3D[
  solution[x, y] - (
    
$$\frac{4 \sin[\pi x] \sin[\pi y]}{\pi^4} + \frac{4 \sin[3 \pi x] \sin[\pi y]}{15 \pi^4} + \frac{4 \sin[\pi x] \sin[3 \pi y]}{15 \pi^4} +$$


$$\frac{4 \sin[3 \pi x] \sin[3 \pi y]}{81 \pi^4}
  ), {x, y} \in \Omega, \text{PlotLegends} \rightarrow \text{Automatic}]$$

```



Below we see some pictures of great mathematicians, who have contributed essential parts to Fourier Analysis within the last 250 years.

© All pictures from Wikimedia Commons, in the public domain everywhere.



Daniel Bernoulli (1700 - 1782)



Jean Baptiste Fourier (1768 - 1830)



Peter G. L. Dirichlet (1805 - 1859)



Bernhard Riemann (1826 - 1866)

### 3 Discrete Fourier Transforms

In this chapter some fundamental properties of the discrete Fourier transform (DFT) are explored and applied in examples.

Important aspects are connected with the alias effect. This appears to beginners at first as a limitation for applications of the DFT, but has in fact a huge advantage when it comes to modern signal transmission in high frequency bands above the clock rates of digital devices like mobile phones etc. There, it allows cheap processing without costly hardware by simply subsampling. We will point out this in the following (see examples 4 and 5 in 3.1 below).

---

#### 3.1 Fundamentals on the DFT

In the following, we use the DFT and the DCT in the form as in my textbook [1], i. e., an N-point DFT has the prefactor  $1/N$ . The DFT coefficient  $c_k$  for a piecewise continuously differentiable  $f$  on  $[0, T]$  is defined by

$c_k = \frac{1}{N} \sum_{n=0}^{N-1} y_n e^{-ikn2\pi/N}$  with  $y_n = f(nT/N)$ . Thus we use the prefactor  $1/N$  here. Otherwise you would not get the correct amplitudes of a signal.

This prefactor is set in Mathematica with the option `FourierParameters->{-1,-1}`. The DFT is already implemented in Mathematica with an FFT algorithm (Fast Fourier Transform). We consider a first example:

##### DFT and Frequency Assignment, Handling of Alias Effects

One of the main effects in a DFT is the alias effect, i. e. that in an N - point DFT, circular frequencies of the form  $(k + mN) \omega_0$  in the observed signal cannot be distinguished ( $\omega_0 = 2\pi/T$ , T observation time,  $m \in \mathbb{Z}$ ).

**Example 1.** We consider a DFT for  $f[t] = \sin[8\pi t] + \sin[28\pi t]$  with  $T = 1$  s and  $N = 10$  and plot an interpolating polygonal train between the absolute values of the obtained DFT coefficients. In the Mathematica numbering, the coefficient with the number  $n$  belongs to the DFT coefficient with the number  $n - 1$ , if we refer to my notation in [1]. Here, the coefficients with the numbers 4 and 8 are close to zero. They are due to numerical rounding errors and should be exactly zero. The values 1 for the numbers 5 and 7 deceive due to the alias effect, and fake a single oscillation of 4 Hz with an amplitude of 2.

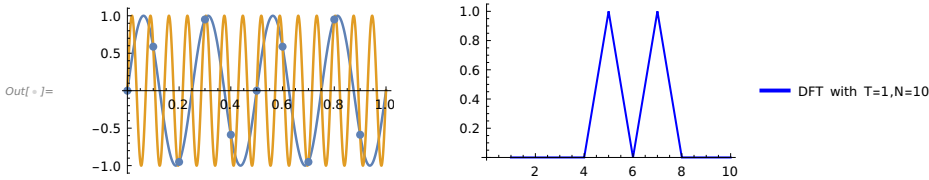
```
In[ ]:= p1 = Plot[{Sin[8. Pi t], Sin[28. Pi t]}, {t, 0, 1}, ImageSize -> Medium];
tab = N[Table[{n / 10, Sin[8. Pi n / 10.]}, {n, 0, 9}]];
p2 = ListPlot[tab, PlotStyle -> PointSize[0.03]];
pa = Show[p1, p2];
```

```
In[ ]:= exmpl1 = Table[Sin[8 Pi n / 10] + Sin[28 Pi n / 10], {n, 0, 9}];
Abs[Fourier[exmpl1, FourierParameters -> {-1, -1}]];
```

```

In[ ]:= p3 = ListLinePlot [% , PlotStyle → Directive [Blue, Thickness [0.008]],
          PlotLegends → {"DFT with T=1,N=10"}];
GraphicsRow [{pa, p3}]

```



You cannot distinguish the two oscillations at the sampling points

The oscillations with 4 Hz and with 14 Hz cannot be distinguished in this DFT, their amplitudes add up there due to the "undersampling". The symmetry of the DFT spectrum can also be explained by the alias effect (see [1], 6.1). In the example, a list of sampled values is generated, which is subjected to an FFT using the Mathematica command `Fourier` to perform an FFT. We then plotted a representation of the DFT magnitude spectrum. A polygonal line is displayed with `ListLinePlot`, which connects the magnitudes of the spectral values.

### This fact is called the Alias Effect

Given a continuous, piecewise continuously differentiable signal  $f$  on  $[0, T]$  with the limit  $f(T-)$  for  $t \rightarrow T$  and a  $T$ -periodic extension  $f_p$  the formula for the alias effect is

$$C_k(f) = \sum_{m=-\infty}^{+\infty} c_{k+mN}(f_p) + \frac{1}{2N} (f_p(0) - f_p(T-)). \quad (***)$$

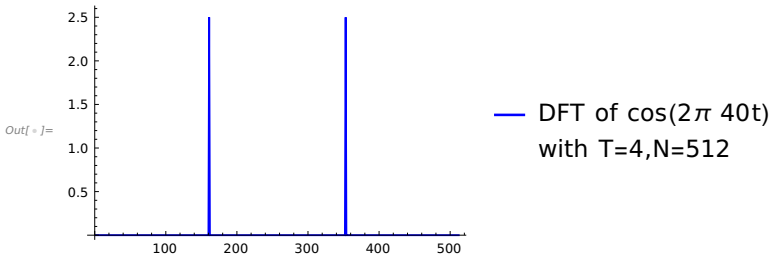
Here  $C_k(f)$  is the  $k$ -th DFT coefficient of an  $N$ -point DFT of the analyzed signal  $f$ , while  $c_{k+mN}(f_p)$  are the Fourier coefficients of the chosen  $T$ -periodic extension  $f_p$  of  $f$  (see [1], 6.1).

To achieve an unambiguous frequency assignment in frequency bands of the form  $[mN/(2T), (m+1)N/(2T)]$  bandpass filters are used in signal processing, which for the selected integer  $m \geq 0$  only allow signal components in the desired frequency band to pass (cf. [1], 6.1).

The sampling frequency to cover a frequency band of width  $N/(2T)$  without aliasing must be at least  $N/T$ . If this condition is fulfilled, the DFT coefficient  $C_k(f)$  can be used as estimate for the Fourier coefficient  $c_k(f_p)$  with frequency  $k/T$ . Otherwise we have aliasing.

**Example 2.** We test a signal composed with frequencies 40 Hz, 216 Hz and 296 Hz with a 512-point DFT and  $T=4$ . Since this DFT covers only 64 Hz as bandwidth  $N/(2T)$ , we are faced with aliasing: The 216 Hz and 296 Hz oscillations are aliased with the 40 Hz oscillation, all amplitudes are added in the DFT coefficients  $C_{160}$  and  $C_{352}$  (my numbering = Mathematica number -1). The DFT coefficient  $C_{352}$  represents the Fourier coefficient  $C_{-160}$  of  $\cos(2\pi 40t) = \cos(\omega T 40 t) = \cos(\pi/2 160 t)$  with  $\omega=2\pi/T$ .

```
In[ ]:= T = 4;
NN = 512;
f[t_] = Cos[2 π 40 t] + 2 Cos[2 π 216 t] + 2 Cos[2 Pi 296 t];
exmpl2 = Table[f[n T / NN], {n, 0, 511}];
absdft = Abs[Fourier[exmpl2, FourierParameters → {-1, -1}]];
ListLinePlot [absdft, PlotStyle → Directive [Blue, Thickness [0.005]],
  PlotLegends → {"DFT of cos(2π 40t)
with T=4,N=512"}]
absdft[[161]]
absdft[[353]]
1 / 4 Integrate [f[t] Exp[-I 160 Pi / 2 t], {t, 0, 4}] (* with period T=4 as in the DFT,
the Fourier coefficient  $c_{160}$  of the 40 Hz oscillation *)
```



Out[ ]:= 2.5

Out[ ]:= 2.5

Out[ ]:=  $\frac{1}{2}$

**Example 3.** What are the non-zero DFT coefficients for the 8 Hz oscillation  $2 \cos(16\pi t)$ , when you make a DFT with  $T=4s$ ,  $N=20$  samples?

Answer:  $2 \cos(16\pi t) = 2 \cos(32 \omega_0 t)$  with  $\omega_0 = 2\pi/T = \pi/2$ . Then  $32-N=12$  and  $-32+2N=8$  yield the according non-zero DFT coefficients  $C_8=i$  and  $C_{12}=-i$ .

This gives a  $8/T=2$  Hz oscillation  $-2\sin(8\omega_0 t)$  with phase reversal as alias due to under-sampling. The base band covered by this DFT is only  $[-2.5 \text{ Hz}, 2.5 \text{ Hz}]$ .

The examples so far show the difficulty in analyzing unknown signals with a DFT without further knowledge on their bandwidth. We now show that the alias effect on the other hand has enormous advantages for processing of signals in a very high frequency range.

**Example 4.** For EMC radiation measurements in the GHz range, you simply avoid sampling rates of several gigasamples per second by using bandpass filters.

For example, bandpass filters with a bandwidth of 1 MHz in subbands are available as analog circuits to achieve the filtering in advance to a DFT.

Then, with only a 512 - point DFT and an observation time of  $T=0.2$  ms per frequency band, i. e. approximately 2.5 MHz sampling frequency  $N/T$ , you achieve a frequency resolution  $1/T$  of about 5 kHz. The analysis of the subbands in a measurement lab can then be put together to form an overall picture ("Undersampling Solution for High Frequency FFT Analysis"). This saves a lot of time and costs for such radiation measurements.

We look at the DFT magnitude spectrum of such an example in a single subband of width 1 MHz, which shows that the alias effect must be carefully considered when to make statements with correct frequency assignments.

It is assumed that the signal is in the frequency band [1GHz, 1GHz + 1MHz], e.g. generated at the output of a corresponding bandpass filter.

*In[ ]:=*  $T = 0.2 \times 10^{-3};$

$NN = 512;$  (\* T observation time, NN number of samples

(NN instead of N here, since N is protected by Mathematica) \*)

$\text{expml2} = \text{Table}[\text{Cos}[2 \text{ Pi } (10^9 + 5000) n T / NN] +$

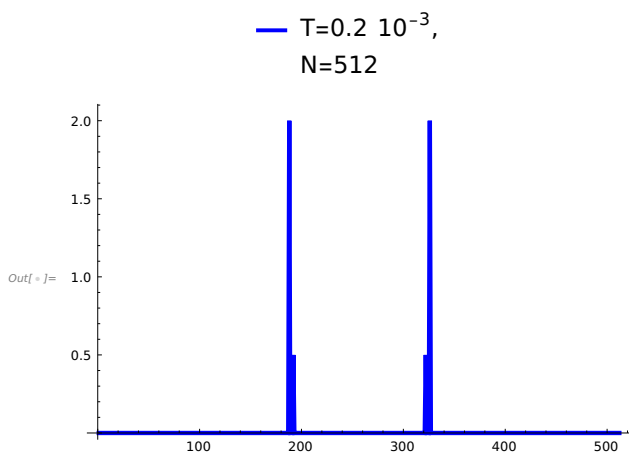
$4 \text{ Cos}[2 \text{ Pi } (10^9 + 25000) n T / NN], \{n, 0, 511\}];$  (\* high frequency signal \*)

We are therefore looking at the superposition of two high - frequency oscillations in the GHz range. We then plot the entire DFT magnitude spectrum as a polygonal curve as well as the relevant parts of it and see how the frequency assignment in the example has to be done. We have taken only 512 samples of the signal in the time  $T = 0.2$  ms.

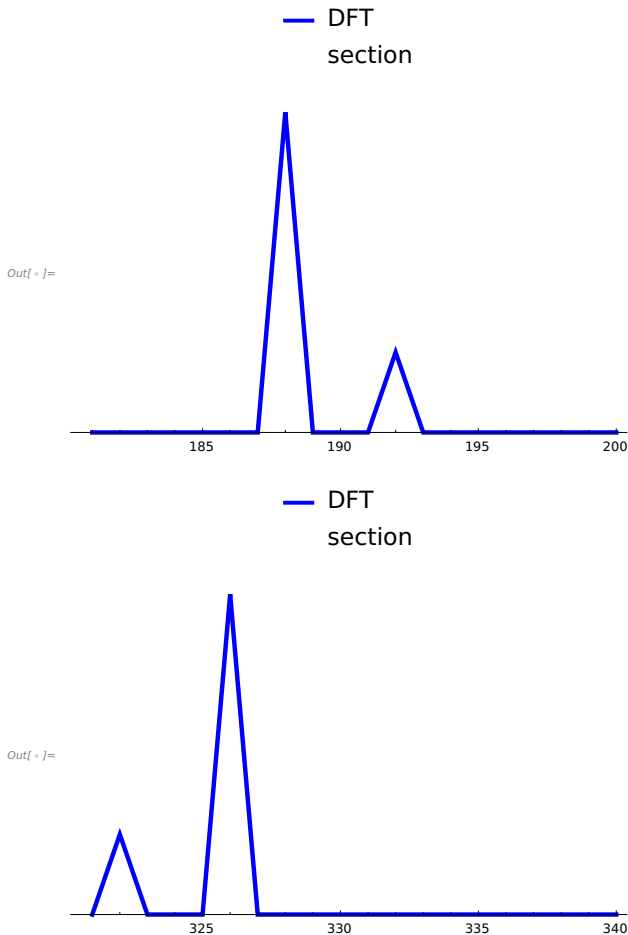
```

In[ ]:= absdft = Abs[Fourier[expml2, FourierParameters -> {-1, -1}]];
p1 = ListLinePlot[%, PlotRange -> All, PlotStyle ->
    Directive[Blue, Thickness[0.008]], PlotLegends -> Placed[{"T=0.2 10-3,
    N=512"}, Above]];
list1 = Table[absdft[[n]], {n, 181, 200}]; (* Extraction of part of the DFT list*)
list2 = Table[absdft[[n]], {n, 321, 340}];
p2 = ListLinePlot[list1, PlotStyle -> Directive[Blue, Thickness[0.008]],
    PlotRange -> All, DataRange -> {181, 200},
    Axes -> {True, False}, PlotLegends -> Placed[{"DFT
section"}, Above]];
p3 = ListLinePlot[list2, PlotStyle -> Directive[Blue, Thickness[0.008]],
    PlotRange -> All, DataRange -> {321, 340},
    Axes -> {True, False}, PlotLegends -> Placed[{"DFT
section"}, Above]];
Show[p1]
Show[p2]
Show[p3]

```







**Frequency assignment:** The two "peaks" of height 2 belong to the oscillation with frequency  $10^9 + 25000$  Hz and amplitude 4.

In Mathematica, compared to the notation in [1] they have a number increased by 1. The peaks with numbers 188 and 326 therefore belong - this is where the alias effect comes into play - to  $4 \cos[2 \pi (10^9 + 25000) t] = 4 \cos[(325 + 390 \cdot N) \omega_0 t]$  with  $\omega_0 = 2 \pi / T = 2 \pi \cdot 5 \cdot 10^3$  rad/s,  $T = 0.2 \cdot 10^{-3}$  s observation time as above, because  $187 = -325 + 512$  and  $(325 + 390 \cdot 512) \cdot 5 \cdot (10^3) = 1000025000$ . The peak with the number 188 in Mathematica then belongs as an alias to the oscillation component  $2 \exp[-i (325 + 390 N) \omega_0 t] = 2 \exp[+i (187 - 391 N) \omega_0 t]$  in the Fourier series of the  $T$ -periodically extended signal.

In the same way, you can assign the other oscillation frequency corresponding to the two peaks with the numbers 192 and 322 and the height 1/2. Please carry out the small analog calculation yourself. In particular, we note that the values associated with the positive signal frequencies with the numbers 322 and 326 lie in the upper half of the DFT spectrum, while those with the numbers 188 and 192 are "alias values" of parts with negative frequencies.

The alias effect therefore sometimes requires a little thought in order to assign the DFT line spectra to the correct frequencies. However, if necessary, you can write a small program for this.

**Exercise:** Consider (with pencil and paper if necessary) which numbers have the peaks of a 512-point DFT with Mathematica for an observation time  $T=0.2 \cdot 10^{-3}$  s of an oscillation with the frequency 1001.06 MHz?

**Example 5. (Subsampling in digital transmission systems)** A decisive advantage of the alias effect with a DFT is found in any kind of digital transmission (WLAN, mobile phones, DVB etc.). The point is that the transmissions take place in very high frequency bands outside the bandwidth of the used digital devices like phones et al. Since the transmission bands are known at the receiver side, the signals are accordingly undersampled, what automatically can generate the signal spectrum in a lower frequency band by aliasing. For example, 5G transmissions can use frequency bands up to 26 GHz, while mobile phones at present have 2.2-2.6 GHz CPU's. For the digital signal processing direct IF subsampling receivers (IF, intermediate frequency) can be used to shift the signal spectrum **without analog mixers by the alias effect** from a high to a low frequency band, where the phone signal processing works. This reduces considerably receiver complexity, power consumption and costs of hardware. We have seen that the bandwidth of a segment of a signal spectrum by a DFT is determined by  $N$  and  $T$ , and thus a segment of the spectrum is representable by a DFT without aliasing. Not a priori determined is the position of such a spectral part on the frequency axis. Its position can be determined from a priori knowledge or deliberately. This has disadvantages in observing unknown signals, but also enormous advantages in signal processing for technical systems as for example in communications engineering. Because there the signals and the allocation of signal frequencies in the spectrum can be chosen intentionally. Thus, the DFT with subsampling offers the opportunity to bring a signal spectrum automatically into a frequency band where device processing works. This is one of the reasons, why digital transmission nowadays is so successful and cheap, because otherwise with analog technique you would need expensive mixers to achieve the same by amplitude modulations. Modern digital transmission with multi-carrier methods like OFDM transmission in high frequency bands the information in spectra of trigonometric polynomials, which can be reconstructed with a DFT by aliasing in a desired lower frequency band. **This is a cornerstone in modern communication systems.** We can see more on this in [1], 12.3 or in a later booklet on Fourier transforms and the principle of OFDM transmissions with Mathematica.

**Example 6. (Gain in Computational Effort by Undersampling in the Radio-Frequency Band)**

Assume we have signals in the radio-frequency band FM from 87.5 MHz to 108 MHz, which shall be digitally processed. Sampling with 216 MHz according to the Nyquist frequency would require an anti-alias lowpass filter with cutoff frequency 108 MHz, and yield a data stream of  $2 \times 216 = 432$  MB/s from a 16 Bit ADC to the signal processing unit. Undersampling with sampling frequency  $f_s = 43.5$  MHz shifts the signal spectrum to [0.5 MHz, 21 MHz]. This would result in a data stream of only 87 MB/s for further signal processing, which is a gain of about 80% in computation time, compared to 432 MB/s, without the need of a (costly) analogue mixer.

**Example 7. (Delayed Sampling, Correction in the Spectrum of Trigonometric Polynomials)**

We consider a sampling of

$$f(t) = 2 \exp[i \omega_0 t] + (1+i) \exp[2i \omega_0 t] + (1-i) \exp[3i \omega_0 t] \text{ with } T=1\text{s}, \omega_0=2\pi/T.$$

Assume that the sampling times are  $t_n = nT/N + 0.1\text{s}$  with  $N=4$  and  $n=0, \dots, N-1$ .

Then the DFT spectrum undergoes changes, compared to a corresponding one beginning at  $t=0$ .

The DFT spectrum of the delayed sampling with the "synchronization error"  $\Delta_t = 0.1\text{s}$  is

$$(0, 1.6180 + 1.1755 i, -0.6420 + 1.2600 i, 0.6420 + 1.2600 i).$$

Since  $f$  is a  $T$ -periodic trigonometric polynomial with frequencies only in the pass-band  $[0, N/T]$ , the DFT coefficients  $C_k$  of  $f$  are simply phase-shifted towards  $D_k = C_k z^k$ ,  $z = \exp[i \omega_0 \Delta_t]$ , due to the delayed sampling.

The spectrum can be corrected using known pilots.

If the amplitude  $A$  of a "pilot carrier" in a transmitted trigonometric polynomial is known (here for example  $A=2$  for the carrier frequency 1 Hz), one can recognize the phase shifts from the obtained DFT coefficient of this carrier and correct the entire DFT spectrum. In the example, the products  $D_k z^{-k}$  with  $z = D_1/A$ ,  $k=0, \dots, 3$  yield the true spectrum  $(0, 2, 1+i, 1-i)$  of  $f$  in the frequency band up to 3 Hz (see below).

$$\text{In[ ]:= } \omega_0 = 2 \pi; A = 2;$$

$$\text{exmpl7} = \text{Table}[A \exp[i \omega_0 (n/4 + 0.1)]$$

$$+ (1+i) \exp[2 i \omega_0 (n/4 + 0.1)] + (1-i) \exp[3 i \omega_0 (n/4 + 0.1)], \{n, 0, 3\}];$$

$$\text{dft} = \text{Fourier}[\text{exmpl7}, \text{FourierParameters} \rightarrow \{-1, -1\}]$$

$$\text{Out[ ]:= } \{-7.21645 \times 10^{-16} + 7.77156 \times 10^{-16} i,$$

$$1.61803 + 1.17557 i, -0.64204 + 1.26007 i, 0.64204 + 1.26007 i\}$$

Now, we correct the spectrum using the pilot carrier as described above:

```
In[ ]:= dftcorrected = Chop[Table[dft[[k]]  $\times$  (dft[[2]]/A)^(-k+1), {k, 1, 4}]]
(* Observe the index numbers of Mathematica *)
```

```
Out[ ]:= {0, 2., 1. + 1. i, 1. - 1. i}
```

```
In[ ]:= DeltaT = Arg[dft[[2]]/A]/ $\omega_0$ 
```

```
Out[ ]:= 0.1
```

Of course, from  $z = \text{Exp}[i\omega_0\Delta_t]$  the time delay  $\Delta_t = \arg(z)/\omega_0$  is obtained as above. In a transmit-receive scenario with a delay of received signals, where the amplitudes of transmitted trigonometric polynomials represent the encoded information in a suitably chosen frequency band, the use of known amplitudes on known carriers (preambles and pilot symbols) is standard in transmissions such as DAB, DVB-T, DSL, WLAN, LTE, 5G. They are used for **synchronization** and generally for **channel estimation**.

### 3.2 Application: Estimation of Signal Spectra, Aliasing

As can be seen from the examples above, the DFT can be used to estimate spectra of unknown signals if knowledge of their bandwidth is available. In general, "truncation effects" happen due to the time window used in the DFT (see[1]). They are caused, among other things, by the uncertainty principle (see[1], time-bandwidth product). We consider signals  $f$  on  $[0, T]$ , which are continuous with a limit  $f(T-)$  and having a piecewise continuously differentiable  $T$ -periodic extension on  $\mathbb{R}$ . If  $w_T$  denotes the rectangular window  $(\text{UnitStep}[t] - \text{UnitStep}[t - T])$  of duration  $T$ , then, due to the alias effect, the DFT coefficient  $C_k(f w_T)$  of an  $N$ -point DFT for signals  $f$  as above compared with the coefficients  $c_k(f w_T)$  of the Fourier series representation of  $f w_T$  is given by

$$C_k(f w_T) = \sum_{m=-\infty}^{+\infty} c_{k+mN}(f w_T) + \frac{1}{2N} (f(0) - f(T-)).$$

**Conclusion:** If the  $T$ -periodic continuation of  $(f w_T)$  has a jump point at  $T$  or if  $f$  has oscillation components with a circular frequency  $\omega \neq 2\pi k/T$ ,  $k \in \mathbb{Z}$ , then distortions occur in the DFT compared to the true signal spectrum (see [1], 12.6). The distortions are referred to in the literature as aliasing and leakage. Let us look at a simple example:

**Example 8.**  $f_1(t) = \cos[t]$  has a discontinuous  $T$ -periodic continuation for  $T = \pi$ .

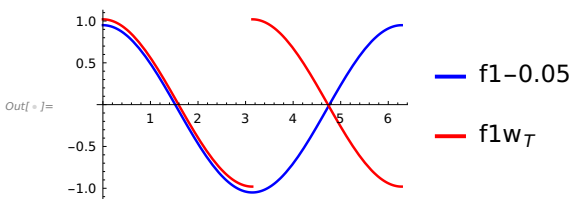
For the function  $f_2(t) = -\cos(t/2) + \cos(t)/2$  the  $T$ -periodic extension of  $f_2 \cdot w_T$  is continuous,  $f_2$  itself, however, is not  $T$ -periodic, but  $4\pi$ -periodic. We first look at sections of the graphs of  $f_1$  and  $f_1 \cdot w_T$ , extended  $T$ -periodically, and sections of the graphs of  $f_2$  and the  $T$ -periodic extension of  $f_2 \cdot w_T$ , each with the rectangular window  $w_T$ ,  $T = \pi$ .

```

In[ ]:= T := Pi; f1[t_] := Cos[t]; wi[t_] := UnitStep[t] - UnitStep[t - T];
f2[t_] := -Cos[t/2] + Cos[t]/2; f1w[t_] := f1[t] × wi[t];
f2w[t_] := f2[t] × wi[t];

In[ ]:= p1 := Plot[f1[t] - 0.05, {t, 0, 2 T}, PlotRange → All, Frame → False,
  FrameStyle → Directive[Black, FontSize → 18, FontWeight → Plain],
  PlotLegends → {"f1-0.05"}, PlotStyle → {Blue, Thickness[0.008]}]
p2 := Plot[f1w[t] + f1w[t - T] + 0.02, {t, 0, 2 T}, PlotRange → All, Frame → False,
  FrameStyle → Directive[Black, FontSize → 14, FontWeight → Plain],
  PlotLegends → {"f1w_T"}, PlotStyle → {Red, Thickness[0.008]}]
p12 = Show[{p1, p2}, ImageSize → Small](* Shown with
  offsets: Here the difference between f1 and the T-periodic extension of f1·w_T *)

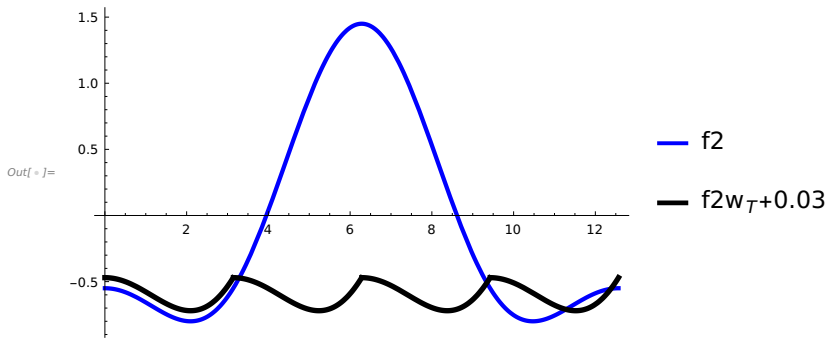
```



```

In[ ]:= p3 := Plot[f2[t] - 0.05, {t, 0, 4 T}, PlotRange -> All, Frame -> False,
        FrameStyle -> Directive[Black, FontSize -> 18, FontWeight -> Plain],
        PlotLegends -> {"f2"}, PlotStyle -> {Blue, Thickness[0.008]}}
p4 := Plot[f2w[t] + f2w[t - T] + f2w[t - 2 T] + f2w[t - 3 T] + 0.03,
        {t, 0, 4 T}, PlotRange -> All, Frame -> False,
        FrameStyle -> Directive[Black, FontSize -> 14, FontWeight -> Plain],
        PlotLegends -> {"f2w_T+0.03"}, PlotStyle -> {Black, Thickness[0.01]}}
p34 = Show[p3, p4]

```



Of course, there are large differences to be expected between the DFT spectra and the true spectra of the periodic signals observed with a rectangular window: First, the DFT magnitude spectrum of  $f_1$  calculated with the rectangular window of duration  $T=2\pi$  with  $N=16$  points. DFT magnitude spectrum of  $f_1$ : This clearly shows the signal circuit frequency of 1 [rad/s]. It corresponds to the peaks at numbers 2 and 16 in the Mathematica numbering.

```

In[ ]:= data1 = Table[N[f1[2 \pi n / 16]], {n, 16}];
absdft1 = Abs[Fourier[data1, FourierParameters -> {-1, -1}]];
plot1 = ListLinePlot[absdft1, PlotRange -> All,
        PlotStyle -> {Blue, Thickness[0.008]}, ImageSize -> Small,
        PlotLegends -> {"DFT of f1w_T",
        T=2\pi, N=16"}];

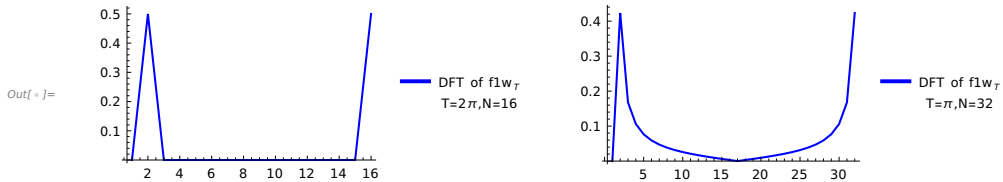
```

The right graphics shows a DFT of  $f_1$  with  $T = \pi$ ,  $N = 32$  sampling points, and thus the magnitude spectrum of  $f_1 \cdot w_T$  with the rectangular window. You can clearly recognize the alias effects, which suggest oscillation components with angular frequencies  $\neq 1$  [rad/s], if the DFT is used naively and the values are taken at face for signal interpretation.

```

In[ ]:= data2 = Table[N[f1w[Pi n/32]], {n, 32}];
absdft2 = Abs[Fourier[data2, FourierParameters -> {-1, -1}]];
plot2 =
  ListLinePlot[absdft2, PlotRange -> All, PlotStyle -> {Blue, Thickness[0.008]},
    ImageSize -> Small, PlotLegends -> {"DFT of f1w_T
T=pi,N=32"}];
GraphicsRow[{plot1, plot2}]

```



Now the corresponding DFT representations for  $f_2$  and  $f_2 w_T$ , in the first case left with  $T=8\pi$ , i.e., twice the exact period of  $f_2$ , in the second case right with  $T=\pi$ , in each case with the rectangular window and  $N=16$  points for the DFT. In the 2nd case, the actual signal spectrum cannot be estimated correctly (note, for example, the high “DC component” with the number 1, which in fact is zero for  $f_2$ ).

```

In[ ]:= data3 = Table[N[f2[8 Pi n/16]], {n, 16}];
absdft3 = Abs[Fourier[data3, FourierParameters -> {-1, -1}]];
plot3 =
  ListLinePlot[absdft3, PlotRange -> All, PlotStyle -> {Blue, Thickness[0.008]},
    ImageSize -> Small, PlotLegends -> {"DFT of f2w_T
T=8pi,N=16"}];

```

### Let us repeat which frequencies are represented by the DFT peaks:

The peak with number 3 in Mathematica corresponds in my notation above to the DFT coefficient  $C_2$ .

The corresponding frequency is  $k/T$  with  $k=2$  and our used  $T=8\pi$ , i.e., it belongs to the frequency  $\nu = 1/(4\pi)$  of  $\cos(t/2)$ .

Correspondingly, the peak with Mathematica number 5 belongs to the frequency  $\nu = 4/(8\pi) = 1/(2\pi)$  of  $\cos(t)$ .

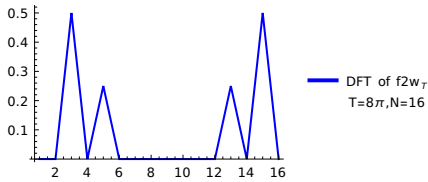
Both magnitudes are halved due to  $\cos(t) = \frac{e^{it}}{2} + \frac{e^{-it}}{2}$ . The peaks in the upper half of the DFT spectrum thus represent the complex Fourier coefficients  $C_{-2}$  and  $C_{-4}$  of the signal  $f_2$ , aliased to the DFT coefficients  $C_{14}$  and  $C_{12}$  (in my notation).

**The same example with  $T=\pi$  is useless, since  $T$  is not large enough for the  $4\pi$ -periodic signal, but shows again the problem in analyzing unknown signals with a DFT:**

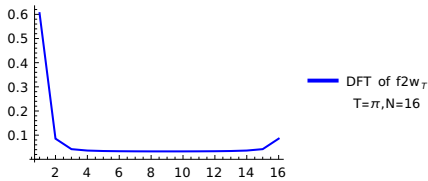
```

In[ ]:= data4 = Table[N[f2w[Pi n / 16]], {n, 16}];
absdft4 = Abs[Fourier[data4, FourierParameters -> {-1, -1}]];
plot4 =
  ListLinePlot[absdft4, PlotRange -> All, PlotStyle -> {Blue, Thickness[0.008]},
    ImageSize -> Small, PlotLegends -> {"DFT of f2w_T
T= $\pi$ , N=16"}];
GraphicsColumn[{plot3, plot4}]

```



Out[ ]:=



**Equally problematic** is the detection of vibration components with closely neighboring frequencies, especially if such components have very different amplitudes. Small amplitude values next to larger amplitude values are obscured by distortion effects of the DFT, or if the observed signal is superimposed by short-term disturbances. See the examples in [1] and the relationship between the smoothness properties of the signal on the one hand and the decay of the magnitude spectrum of periodic functions on the other hand. In practice, attempts are made to obtain the best possible spectral estimates by long observation periods, high sampling frequencies and a correspondingly high number of samples and by using weight functions, so-called time windows. More on this in the next section.

**We note:** An increase in the observation time (and thus an increase in the number of samples) improves the frequency resolution  $1/T$ , an increase in the sampling frequency (again with a corresponding increase in the number of samples) increases the recorded bandwidth. Both effects counteract distortions caused by aliasing and leakage. A further tool for such an improvement is the use of weighting functions in the time domain.



### 3.3 Leakage, Time Windows

In a DFT with, for instance, an even number  $N$  of samples and a time window  $w_T$  one uses the DFT coefficient  $C_k$  for  $k=0,\dots,(N-2)/2$  as an approximation for the Fourier coefficient  $c_k$  of  $fw_T$ . For  $k=(N+2)/2,\dots,N-1$  the coefficient  $C_k$  serves accordingly as an approximation for  $c_{-N+k}$  and  $C_{N/2}$  as an approximation for  $(c_{-N/2} + c_{N/2})/2$ . The corresponding oscillations to the fundamental circular frequency  $\omega_0=2\pi/T$

$$v_0(t)=1, v_1(t)=e^{i\omega_0 t}, \dots, v_{(N-2)/2}(t)=e^{i(N-2)\omega_0 t/2}, v_{N/2}(t)=\cos(N\omega_0 t/2), \\ v_{(N+2)/2}(t)=e^{-i(N-2)\omega_0 t/2}, \dots, v_{N-1}(t)=e^{-i\omega_0 t}$$

generate an  $N$ -dimensional function vector space  $V$  in  $L^2([0,T])$ . We have seen that  $C_k$  contains in sum all Fourier coefficients  $c_{k+mN}$  of a  $T$ -periodic extension  $f_p$  of  $fw_T$ ,  $m$  in  $\mathbb{Z}$ .

For the rectangular window  $w_T$ , the  $T$ -periodic extension of  $fw_T$  has discontinuities at  $t=kT$ ,  $k$  in  $\mathbb{Z}$ , if  $f(0) \neq f(T-)$ .

If the signal  $f$  is a mixture of harmonic oscillations with circular frequencies  $k\omega_0$ ,  $k=0,\dots,N/2$ , i.e., if  $f(t)=\sum_{k=0}^{N-1} \alpha_k v_k(t)$  is a linear combination of the functions  $v_0,\dots,v_{N-1}$ , then  $f(0)=f(T-)$  and it follows

$$C_k = \langle f | v_k \rangle = \frac{1}{T} \int_0^T f[t] \text{Conjugate}[v_k[t]] dt = \alpha_k.$$

The orthogonal projections of  $f$  onto the one-dimensional subspaces of  $V$  generated by the functions  $v_k$  then yield with the DFT coefficients the exact spectral values of  $f$ . That is different, if the periodic extension of  $fw_T$  has a jump discontinuity at  $t=T$  or if the originally observed signal  $f$  contains harmonic oscillations, whose period duration does not match  $T$ . In practice, this will often be the case when analyzing unknown signals  $f$ , which are sampled over an arbitrarily chosen time period. Simple examples of such cases are given by the functions  $f_1(t)=\cos(t)$  and  $f_2(t)=-\cos(t/2)+\cos(t)/2$ , as we have seen above in the example 8. For  $T=\pi$ , the  $T$ -periodic extension of  $f_1 w_T$  with the rectangle window  $w_T$  has a jump discontinuity at  $T$ , while that of  $f_2 w_T$  is continuous, but  $f_2$  is not  $T$ -periodic.

If  $f w_T(0) \neq f w_T(T-)$ , then every  $T$ -periodic extension,  $T=N\Delta_t$ , of  $f$  beyond the interval  $[0, T_a]$ ,  $T_a=(N-1)\Delta_t$ , has jump discontinuities or steep flanks in the vicinities of the points  $kT$ ,  $k$  in  $\mathbb{Z}$ . From considerations on the asymptotics of Fourier coefficients (cf. [1]), it follows that the magnitudes of the coefficients  $c_k$  of a  $T$ -periodic extension of the signal section decrease only slowly for  $|k| \rightarrow \infty$ . Consequences are aliasing effects in the coefficients  $C_k$  of the discrete Fourier transform. Even if by chance  $f w_T(0) = f w_T(T-)$  as in the example  $f_2 w_T$ , effects arise as soon as  $f$  contains oscillation components with frequencies  $\nu \neq k/T$ , and also if they lie within the Nyquist interval with the cutoff frequency  $N/(2T)$ . Such effects are called leakage effects.

Every signal component with a circular frequency  $\omega_1 \neq 2\pi k/T$  for any  $k$  has non-zero projections in all subspaces of  $L^2([0,T])$ , which are generated by the functions  $v_k$  for  $k=0,\dots,N-1$ , and causes leakage effects:

$$\langle e^{i\omega_1 t} w_T | v_k \rangle \neq 0 \text{ for all } k=0,\dots,N-1.$$

**Example 9.** Consider for example the signal  $g(t) = A \exp[i\omega_1 t]$ . Then, for the  $k$ -th Fourier coefficient  $c_k(gw_T)$  of  $gw_T$  with the rectangle window  $w_T$  for the interval  $[0, T]$  it can be shown (with the Fourier transforms of  $gw_T$ , see [1], 11.5 and 12.6)

$$c_k(gw_T) = (-1)^k A \exp[i\omega_1 t] \sin(\pi k - \omega_1 T/2) / (\pi k - \omega_1 T/2)$$

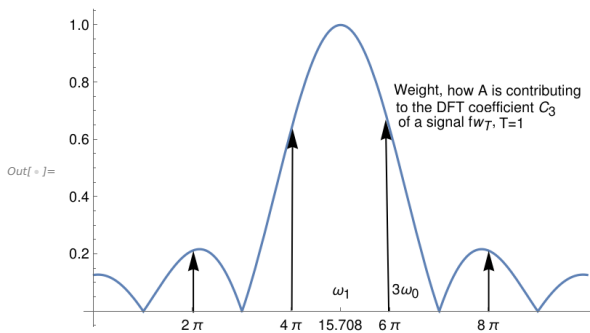
(\*\*\*)

These coefficients distort the amplitudes and phases of the estimates  $C_k$  of signal components at all frequencies  $k/T$ ,  $k \leq N/2$ , if  $\omega_1 \neq 2\pi k/T$ .

They contribute as alias effects to all DFT coefficients  $C_k$ . They are spread onto the oscillations at all frequencies  $k/T$  (see the next figure). This phenomenon is referred to in signal processing as the "spectral leakage effect". Additionally, for all  $C_k$ , there is a constant additive component  $(g(0) - g(T-))/(2N)$ , if the  $T$ -periodic extension of  $gw_T$  has a jump discontinuity at  $T$ .

The spectral leakage effect occurs with modified coefficients  $c_k(gw_T)$  even when using other window functions  $w_T$  instead of the rectangular window, and it results from the uncertainty principle for the time-duration-bandwidth product of the window  $w_T$  (see [1], 12.4).

The arrows in the following figure show some absolute weights  $g_k = |c_k(gw_T)/A|$ , through which the amplitude  $A$  of  $gw_T$  is distributed onto the Fourier coefficients belonging to frequencies  $2\pi k/T$  adjacent to  $\omega_1$  by the periodicity induced by  $w_T$  according to (\*\*\*) above. For the figure the following data are used:  $T=1$ , i.e.,  $\omega_0=2\pi$ ,  $\omega_1=5\pi=15.708$ ,  $A=1$ . Shown is  $|(\mathcal{F}w_T)(\omega - \omega_1)/T|$ ,  $\mathcal{F}w_T$  the Fourier transform of  $w_T$  (see [1], chapter 10).



## Time Windows

We observe that a possibility to mitigate alias and leakage effects is the use of windows  $w_T$  whose Fourier transforms decay faster than that of the rectangle function. The faster the decrease of  $\mathcal{F}w_T(\omega)$  for  $|\omega| \rightarrow \infty$  the better is the frequency localization in  $fw_T$ . This is a **consequence of Heisenberg's uncertainty principle and properties of Fourier transforms**. These facts are explained in detail in a subsequent booklet on Fourier transforms with Mathematica (see also [1], chapter 12). From that, it also a fact that the bandwidth of a window function  $w_T$  increases, when  $T$  becomes smaller.

Thus, with decreasing  $T$  again the decay of  $\mathcal{F}w_T$  becomes slower and thus increases aliasing and leakage. Therefore, when using the discrete Fourier transform, some fundamental aspects of the interaction between the observation duration  $T$ , the properties of the weighting function  $w_T$ , and the sampling rate of the DFT must be considered.

In practice, many different weighting functions  $w_T$  are used. The use of special window functions and thus the compromise that must always be made due to the uncertainty principle depends on the aim of the respective application. Criteria besides the decay  $w_T$  and the bandwidth of the window include, for example, its energy concentration in a given frequency band or simple calculation and implementation possibilities in software applications.

### General aspects for choosing a window:

1. One usually chooses a window function  $w_T$  that is as smooth as possible with support in  $[0, T]$  and  $w_T(0)=w_T(T)=0$  or a little bit wider. Then, with support in  $[0, T]$ , the  $T$ -periodic extension of  $fw_T$  for continuous signals  $f$  has no jump discontinuities, and the aliasing effects described above are reduced, if the Fourier coefficients of this extension decrease rapidly. One then obtains better **estimates with  $C_k T$**  than with the rectangular window **for the values  $\mathcal{F}f(2\pi k/T)$** , which are often sought in applications.
2. One chooses the observation duration  $T$  to be as long as possible. The smaller  $T$  is, the larger the bandwidth of  $\mathcal{F}w_T$ , i.e., the worse the frequency localization.
3. One chooses the number  $N$  of samples to be as high as possible. More signal frequencies are then resolved exactly. For fully observed time-limited signals  $f$ , "zero padding" improves the approximations for  $\mathcal{F}f$ .
4. The leakage effect is less significant, the faster the side lobes of  $|\mathcal{F}w_T|$  decrease compared to the main lobe (cf. the preceding image). Therefore, window functions are often chosen where these side lobes of  $|\mathcal{F}w_T|$  decrease rapidly.

**Example 10.** We consider as an illustrative example the signal  $f(t) = A \cos(2\pi \nu_1 t) + B \cos(2\pi \nu_2 t)$  with  $A=1$ ,  $B=0.02$ ,  $\nu_1=10.25$  Hz,  $\nu_2=12$  Hz.

The figure left shows the discrete Fourier transform with the rectangular window  $w_T$ ,  $T=2$  s, for  $N=128$ . The signal frequency  $\nu_2$  cannot be detected. With the same  $T$  and  $N$ , the often-used **Hann window  $w_T$**  is used in the middle figure,  **$w_T(t) = 0.5 - 0.5 \cos(2\pi t/T)$  for  $0 \leq t \leq T$** , and finally the same  $w_T$  with  $T=5$  s and  $N=1024$  at the right. Displayed are single-sided DFT magnitude spectra.

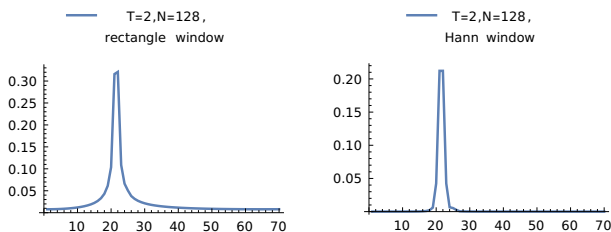
$$\text{In}[*]:= \text{f}[t\_]= \text{Cos}[2 \text{ Pi } 10.25 \text{ t}] + 0.02 \text{ Cos}[2 \text{ Pi } 12 \text{ t}]$$

$$\text{Out}[*]= \text{Cos}[64.4026 \text{ t}] + 0.02 \text{ Cos}[24 \pi \text{ t}]$$

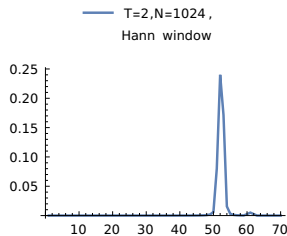
```

In[ ]:= T = 2; NN = 128; list1 = Table[f[n T / NN], {n, 0, NN - 1}];
dft1 = Fourier[list1, FourierParameters -> {-1, -1}];
list1a = Table[Abs[dft1[[k]]], {k, 1, 70}];
p1 = ListLinePlot[list1a, PlotRange -> All, PlotLegends -> Placed[{"T=2,N=128,
rectangle window"}, Above]];
hann[t_] := 0.5 - 0.5 Cos[2 Pi t / T];
fwt[t_] := f[t] × hann[t];
list2 = Table[fwt[n T / NN], {n, 0, NN - 1}];
dft2 = Fourier[list2, FourierParameters -> {-1, -1}];
list2a = Table[Abs[dft2[[k]]], {k, 1, 70}];
p2 = ListLinePlot[list2a, PlotRange -> All, PlotLegends -> Placed[{"T=2,N=128,
Hann window"}, Above]];
T = 5;
NN = 1024;
list3 = Table[fwt[n T / NN], {n, 0, NN - 1}];
dft3 = Fourier[list3, FourierParameters -> {-1, -1}];
list3a = Table[Abs[dft3[[k]]], {k, 1, 70}];
p3 = ListLinePlot[list3a, PlotRange -> All, PlotLegends -> Placed[{"T=2,N=1024,
Hann window"}, Above]];
p4 = GraphicsRow[{p1, p2}];
GraphicsColumn[{p4, p3}]

```



Out[ ]:=



From the DFT result in the third image, the 12 Hz signal frequency can at least be suspected. Observe also that the amplitudes are diminished by the use of the damping window function.

### 3.4 Inverse Discrete Fourier Transform IDFT, Interpolation with a DFT

#### 1. Inverse Discrete Fourier Transform IDFT

The DFT is invertible. Its inverse IDFT is given for  $N$  samples and duration  $T$  with the DFT coefficients  $C_k$  of a function  $f$  by

$$\text{IDFT} \quad y_n = f(nT/N) = \sum_{k=0}^{N-1} C_k \text{Exp}[i k n 2\pi/N] \text{ for } 0 \leq n \leq N-1.$$

For calculations we have the properties of a DFT summarized in the table below, compared with analog properties of Fourier series. This table is copied from [1], where also the according proofs can be found. The DFT coefficients  $C_k$  are denoted by  $\hat{c}_k$  in the table as in [1].

Fourier Series	DFT	
Time Domain Function	Samples	Spectral Values
$f(t)$ $T$ -periodic $f(t) = \sum_{k=-\infty}^{+\infty} c_k e^{j k \omega_0 t}, \quad \omega_0 = \frac{2\pi}{T}$ $c_k = \frac{1}{T} \int_0^T f(t) e^{-j k \omega_0 t} dt$	$y_n = f(nT/N)$ $N$ -periodic $y_n = \sum_{k=0}^{N-1} \hat{c}_k e^{j k n 2\pi/N}$	$\hat{c}_k =$ $\frac{1}{N} \sum_{n=0}^{N-1} y_n e^{-j k n 2\pi/N}$
<b>Similarity</b> $f(\alpha t) = \sum_{k=-\infty}^{+\infty} c_k e^{j k \alpha \omega_0 t}$ $T/\alpha$ -periodic, $\alpha > 0$	$y_n = f(nT/(\alpha N))$ otherwise as above	as above
<b>Translations, Amplitude Modulation</b> $f(t + t_0) = \sum_{k=-\infty}^{+\infty} c_k e^{j k \omega_0 (t+t_0)}$ $e^{j m \omega_0 t} f(t) = \sum_{k=-\infty}^{+\infty} c_{k-m} e^{j k \omega_0 t}$	$(y_{n+m})_{n \in \mathbb{Z}}, (m \in \mathbb{Z})$ $(z^{nm} y_n)_{n \in \mathbb{Z}}$	$(z^{km} \hat{c}_k)_{k \in \mathbb{Z}}, z = e^{j 2\pi/N}$ $(\hat{c}_{k-m})_{k \in \mathbb{Z}}$
<b><math>T</math>-periodic Convolution</b> For $f(t) = \sum_{k=-\infty}^{+\infty} c_k e^{j k \omega_0 t}$ $g(t) = \sum_{k=-\infty}^{+\infty} d_k e^{j k \omega_0 t}$ $(f * g)_T(t) = \sum_{k=-\infty}^{+\infty} c_k d_k e^{j k \omega_0 t}$	<b><math>N</math>-periodic Convolution</b> For $y_n = f(nT/N)$ $x_n = g(nT/N)$ $(y * x)_{n \in \mathbb{Z}} = \frac{1}{N} \sum_{m=0}^{N-1} x_m y_{n-m}$	with DFT Coefficients $(\hat{c}_k)_{k \in \mathbb{Z}}$ $(\hat{d}_k)_{k \in \mathbb{Z}}$ $(\hat{c}_k \hat{d}_k)_{k \in \mathbb{Z}}$
<b>Parseval Equality</b> $\ f\ ^2 = \sum_{k=-\infty}^{+\infty}  c_k ^2$	$\frac{1}{N} \sum_{n=0}^{N-1}  y_n ^2 = \sum_{k=0}^{N-1}  \hat{c}_k ^2$	

## 2. Trigonometric Interpolation

By definition of the DFT we immediately obtain trigonometric interpolation polynomials for functions  $f$  on an interval  $[0, T]$  with interpolation nodes  $f(kT/N)$ ,  $k=0, \dots, N-1$ . This means that the IDFT of the DFT list  $\{C_k, k=0, \dots, N-1\}$  yields the list of samples  $\{y_k=f(kT/N), k=0, \dots, N-1\}$ , whose DFT is just  $\{C_k, k=0, \dots, N-1\}$ . Thus, we obtain from the formula for the IDFT that  $Q(t)$  yields a trigonometric interpolation polynomial with  $Q(t)=\sum_{k=0}^{N-1} C_k \text{Exp}[i k 2\pi t/T]$  for these interpolation nodes  $y_k$ . By the alias relation, this can be written in another form, when the samples and the DFT coefficients are  $N$ -periodically extended. We consider two cases:

### 1. The Number of Samples $N=2m+1$ is Odd

Then the function

$$P(t)=\sum_{k=-m}^m C_k \text{Exp}[i k 2\pi t/T]$$

is the uniquely determined trigonometric interpolation polynomial of degree at most  $(N-1)/2$ . If the samples are real, then  $P(t)$  is also real-valued. In particular  $P = f$ , if the function  $f$  is a  $T$ -periodic trigonometric polynomial of degree at most  $m$ .

### 2. Trigonometric Interpolation with an Even Number of Samples

If  $N = 2m$  is even, then the interpolation problem is not uniquely solvable.

The trigonometric polynomial  $P(t)=\sum_{k=-m}^m \alpha_k \text{Exp}[i k 2\pi t/T]$  has  $N+1$  coefficients. The DFT yields  $N$  coefficients  $C_0, \dots, C_{N-1}$ . The function

$$P_1(t)=\sum_{k=-m}^{m-1} C_k \text{Exp}[i k 2\pi t/T]$$

is an interpolation function for  $f$ , but in general is not real-valued. By the alias relation, we can obtain as a second interpolation

$$P_2(t)=\sum_{k=-m+1}^{m-1} C_k \text{Exp}[i k 2\pi t/T] + C_{N/2} \cos[m 2\pi t/T].$$

If the samples are real-valued, then  $P_2$  is also a real-valued function. It is the unique trigonometric interpolation polynomial in the vector space  $V_m$  spanned by  $1, \cos(k\omega_0 t), \sin(k\omega_0 t)$  for  $k=1, \dots, m-1$ , and  $\cos(m\omega_0 t)$ ,  $\omega_0=2\pi/T$ . With  $a_k = C_k + C_{-k} = 2 C_k$ ,  $b_k = i(C_k - C_{-k})$  we can write  $P_2$  in the form

$$P_2(t)=\frac{a_0}{2} + \sum_{k=1}^{m-1} a_k \cos(\omega_0 k t) + b_k \sin(\omega_0 k t) + \frac{a_m}{2} \cos(m\omega_0 t).$$

If  $f$  can be extended to a  $T$ -periodic even function, then all coefficients  $b_k=0$ . If an odd  $T$ -periodic extension is possible, then all  $a_k=0$ .

**Example 11.** For  $N=4$ ,  $t_n=n\pi/2$ ,  $T=2\pi$  with samples  $y_0=1$ ,  $y_1=2$ ,  $y_2=1$  and  $y_3=3$  we compute  $P_2$  as above and obtain  $P_2(t)=7/4 - 1/2 \sin(t) - 3/4 \cos(2t)$ . Also  $P(t)=P_2(t)+\alpha \sin(2t)$  with arbitrary real  $\alpha$  is a trigonometric interpolation polynomial of degree 2, since  $\sin(2t_n)$  always is zero. However, such a function  $P$  is not in the space  $V_2$  as defined above for  $\alpha \neq 0$ .

The given interpolations  $P_1$  and  $P_2$  are trigonometric polynomials in the baseband to a DFT. For bandpass signals  $f$ , trigonometric interpolation polynomials in the corresponding passband can also be given with the help of a DFT and bandpass sampling. In particular, trigonometric polynomials in a passband can be reconstructed exactly with a DFT. The formulation of this is left to the readers.

**Example 12.** *Interpolation of Fourier transforms of time-limited functions, upsampling by zero-padding.*

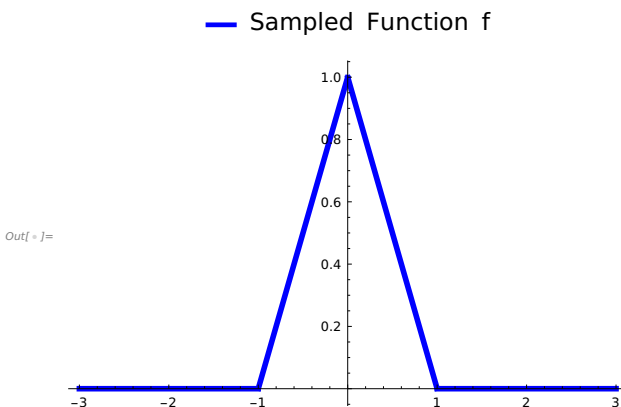
The DFT is often used to obtain approximations for Fourier transforms  $\mathcal{F}f$  of generally non-periodic functions  $f$ . It makes sense to consider time-limited functions, i.e., functions with bounded support (see below). For such functions  $f$ , assumed to be piecewise continuously differentiable with support, for example, in  $[-T, T]$  it holds for samples  $F_k$  of their Fourier transform at points  $2\pi k/T$  and the Fourier coefficients  $c_k$  of the  $T$ -periodic extension of  $f$  the relation  $F_k = T c_k$  (see [1], 11.5). Therefore, we can use the DFT to approximately compute samples of the Fourier transform of  $f$ , by which we can approximately illustrate the Fourier transform through interpolation.

Enlarging the sampling duration  $T$  by additionally appending zeros to the samples (with equal sampling frequency) one can improve with higher frequency resolution the approximation for  $\mathcal{F}f$ . Analogously, zero-padding in the frequency range yields an upsampling of a time-signal by an IDFT. This is widely used in the algorithms of OFDM transmissions in digital communications (for details see [1], 12.3).

For this, the interpolation points are then  $(2\pi k/T, T c_k)$  with the DFT coefficients  $c_k$  and updated sampling time. We test an example with a triangle function and its known Fourier transform.

```
In[ ]:= T = 1;
f[t_] = (t + T) (HeavisideTheta[t + T] - HeavisideTheta[t]) +
        (-t + T) (HeavisideTheta[t] - HeavisideTheta[t - T]);
F[w_] = FourierTransform[f[t], t, w, FourierParameters -> {1, -1}]
pf = Plot[f[t], {t, -3, 3}, PlotLegends -> Placed[{"Sampled Function f"}, Above],
        PlotStyle -> {Blue, Thickness[0.01]}
```

$$\text{Out[ ]} = \frac{2 - 2 \cos[w]}{w^2}$$

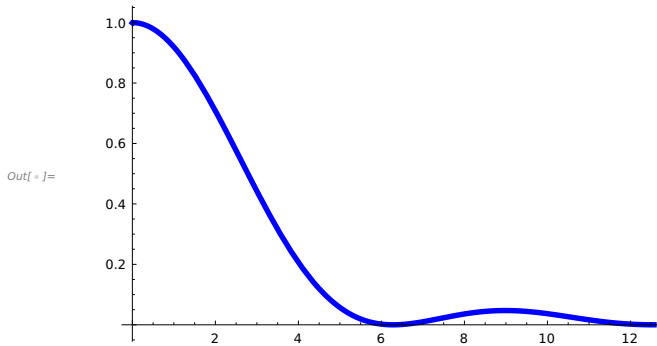


```

In[ ]:= pF = Plot[Abs[F[w]], {w, 0, 4 Pi}, PlotStyle -> {Blue, Thickness [0.01]},
    PlotLegends -> Placed[{"Fourier Transform of the Sampled Function f,
    right sided"}, Above]]

```

— Fourier Transform of the Sampled Function f,  
right sided



### Zero-Padding

With 9 samples we obtain a first rather rough approximation for  $\mathcal{F}f$  covering the angular frequency range up  $4\pi$ . It is shown in the left image below.

Then, we can improve the approximation by zero-padding. *We append 2039 zeros to the samples of  $f$ .* Equivalently, we sample  $f$  with the same sample rate over the time of  $T=2048/4s=512s$ . The errors come from the numerical approximation of Fourier coefficients by the DFT and from alias effects, since  $\mathcal{F}f$  has unbounded support.

```

In[ ]:= data = Table[ f[-1 + 2 n/9], {n, 0, 8}];
phase = Table[(-1)^k, {k, 0, 8}];
dft = phase Chop[Fourier[data, FourierParameters -> {-1, -1}]]
pF1 = ListLinePlot[2 dft[[1 ;; 5]],
    DataRange -> {0, 4 Pi}, PlotStyle -> {Blue, Thickness [0.01]},
    PlotLegends -> Placed[{"Approximation of the Fourier Transform of f,
    9 Samples of f"}, Above]];
Out[ ]:= {0.493827, 0.204713, -0.00699058, 0.0246914,
    -0.0105191, 0.0105191, -0.0246914, 0.00699058, -0.204713}

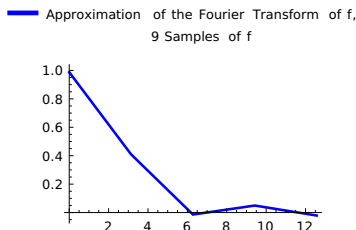
```



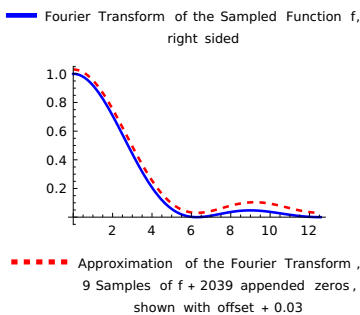
```

In[ ]:= datanew = Table[f[-1 + n/4], {n, 0, 2047}]; (* equal sampling rate *)
phase = Table[(-1)^k, {k, 0, 2047}];
dftnew = phase Chop[Fourier[datanew, FourierParameters -> {-1, -1}]];
pz = ListLinePlot[Abs[512 dftnew[[1 ;; 1024]]] + 0.03, DataRange -> {0, 4 Pi},
  PlotRange -> All, PlotStyle -> {Dashed, Red, Thickness [0.01]},
  PlotLegends -> Placed[{"Approximation of the Fourier Transform ,
9 Samples of f + 2039 appended zeros,
shown with offset + 0.03"}, Below]];
pF2 = Show[pF, pz]; (* updated sampling time T according
to the number of samples included the appended zeros *)
GraphicsColumn [{pF1, pF2}]

```



Out[ ]:=



### 3.5 DFT, IDFT and Time Windows in Digital Signal Processing

#### Example 13. WLAN transmission with windowed OFDM

In digital communication with multi-carrier transmissions, usually **OFDM** (orthogonal frequency division multiplexing) is used in **DMT, DSL, DVB, WLAN, LTE or 5G**.

The information for transmission is fed to the transmitter as an encoded bit stream.

The information of  $N$  bit groups is transformed to *complex amplitudes  $c_k$  of a trigonometric polynomial*  $S(t) = \sum_{k=-N/2}^{N/2} c_k e^{ik2\pi t/T}$  on a time interval  $[0, T]$  with equally spaced frequencies  $k/T$ ,  $k=-N/2, \dots, N/2$  ( $N$  even).

The parts of a single frequency are called *carriers*. The carriers are pairwise orthogonal in  $L^2([0, T])$ , which motivates the name OFDM for the method. The *trigonometric polynomial  $S$*  has the bandwidth of 20 MHz in **WLAN**. Control of this 20 MHz transmission bandwidth in WLAN is the task of the respective hardware in a WLAN device. Physically, the signals are voltage curves across time.

The **real-valued transmission pulse  $S_R$**  is obtained from  $S$  by *quadrature amplitude modulation (QAM) with an angular intermediate center frequency  $\omega_c$*  and multiplied with a time-window  $w_T$ , i.e., defined by

$S_R(t) = \text{Re}[e^{i\omega_c t} \sum_{k=-N/2}^{N/2} c_k e^{ik2\pi t/T}] w_T(t)$ . The Fourier coefficient  $c_0$  is set to zero.

The rectangle time window  $w_T$  is simply the function  $1_{[0, T]}$ . Different modulations for transmitting this signal are possible and in use. We focus on the **16QAM** modulation in WLAN at 2.4 GHz with bandwidth 20 MHz. With 16QAM, a pulse is transmitting **48 data carriers**, each mapping a 4-bit-group out of the encoded bitstream to a complex amplitude of a carrier. **16QAM maps each possible 4-bit-group one-to-one to a complex amplitude out of 16 possibilities** (see the constellation diagram on p. 85). The quadrature amplitude modulation preserves orthogonality of the carriers and the signal bandwidth. By QAM, the spectrum is shifted to  $\omega_c$  as angular center frequency. The samples obtained by an IDFT of the amplitudes  $c_k$  are interpolated to eventually obtain the real-valued transmission signal  $S_R$  (see also above p.79, 3.4, Trigonometric Interpolation). Thus, almost all can be achieved with discrete signal processing. **The receiver can invert this QAM modulation**, thus get back the complex signal  $S$  in the baseband (center frequency = 0), **detect its complex amplitudes by a DFT of its samples** and can thus **reconstruct the transmitted bits from the 16QAM mapping**.

**Signal Processing:** The **transmitter** generates with the amplitudes of  $S$  a *discrete time signal by an IDFT* to obtain a number of interpolation points. An also discrete quadrature amplitude modulation of that complex samples yields the samples of the real-valued signal  $S_R$  that can be *fed to a lowpass filter (like a Butterworth lowpass filter) to obtain a continuous real-valued signal for transmission* (compare the Shannon Sampling Theorem). The transmit power of a WLAN device (e.g. 100 mW) is the actual radiated RF power — obtained by scaling and amplifying the signal.

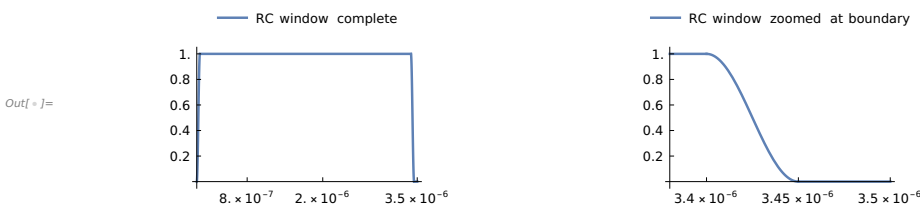
The **receiver** afterwards can invert the QAM modulation and compute from samples of the obtained signal the sought complex amplitudes **simply by a DFT** of that samples, provided that various disturbances in the transmission channel are mastered by suitable *channel equalization* (see for example [4] and [7] on channel estimation).

Finally, from the **16QAM** mapping the transmitted bit sequence can be reconstructed.

The signal contains additionally 4 pilot carriers (-21, -7, 7, 21) with known frequencies and amplitudes at the receiver, which can be detected at the receiver and used for synchronization and channel estimation for denoising. See example 7 in 3.1, p. 68.

**Windowing, Spectral Efficiency:** Using a rectangle time-window  $1_{[0,T]}$  for an information package in a sent pulse however causes much unacceptable *out-of-band emissions* due to the low decreasing of the window's spectrum (like sinc-functions). Therefore smoother time-windows are used in practice as, for example, *RC windows* (raised cosine windows), which have faster spectral decay. We plot an RC window.

```
T = 3.2 × 10−6; (* duration of a rectangle window across seconds *)
tr = T / 16;
alpha = −1 + (T + 5 tr / 4) / T;
T (1 + alpha);
(* slightly longer duration 3.45 × 10−6 s of the RC window *)
rcwindow[t_] = (0.5 × (1 − Cos[Pi t / (tr / 4)]) (UnitStep[t] − UnitStep[t − tr / 4]) +
  (UnitStep[t − tr / 4] − UnitStep[t − (T + tr)]) +
  0.5 × (1 + Cos[Pi (t − (T + tr)) / (tr / 4)])
  (UnitStep[t − (T + tr)] − UnitStep[t − (T + 5 tr / 4)]));
p1w = Plot[rcwindow[t], {t, 0, 3.5 × 10−6},
  PlotLegends → Placed[{"RC window complete"}, Above],
  Ticks → {{0.8 × 10−6, 2. × 10−6, 3.5 × 10−6}, {0.2, 0.4, 0.6, 0.8, 1.0}}];
p2w = Plot[rcwindow[t], {t, 3.38 × 10−6, 3.5 × 10−6},
  PlotLegends → Placed[{"RC window zoomed at boundary"}, Above],
  Ticks → {{3.4 × 10−6, 3.45 × 10−6, 3.5 × 10−6}, {0.2, 0.4, 0.6, 0.8, 1.0}}];
GraphicsRow[{p1w, p2w}]
```



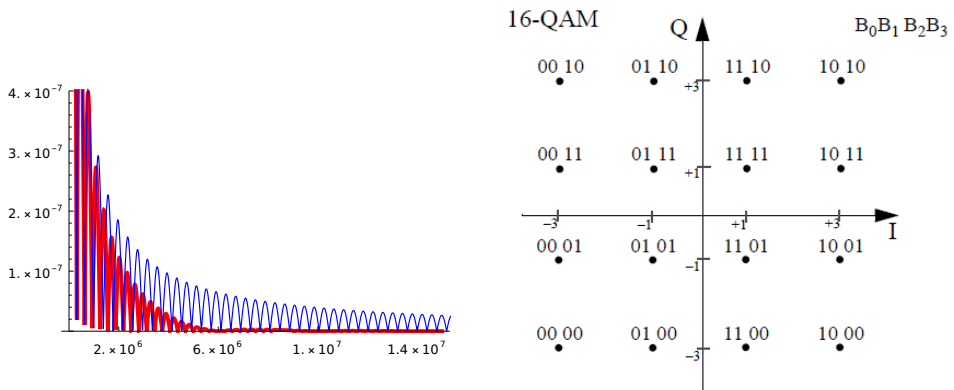
Since the duration of the RC window - shown across time in seconds - is slightly longer than that of the rectangle window, orthogonality of the carriers is lost and a low inter-carrier-interference occurs (ICI), which is mastered however by the 16QAM encoding. Below for comparison, in the left figure you see the decay of the half-sided spectral magnitude of the rectangle window with duration  $T$  and that of the RC window. They are shown across frequencies in Hz from the center frequency of channel 1 in the WLAN example below until the start of WLAN channel 6 with distance  $1.5 \times 10^7$  Hz from the center frequency.

The right figure below shows the 16QAM constellation diagram, IEEE 802.11-2020.

```

In[ ]:= ftwindow[w_] = T Sin[w T / 2] / (w T / 2) Cos[w alpha T / 2] / (1 - (alpha w T / Pi) ^ 2);
(*Fourier transform of the window *)
ftrechteckwindow[w_] =
  FourierTransform[UnitStep[t] - UnitStep[t - T], t, w, FourierParameters -> {1, -1}];
p1 = Plot[Abs[ftwindow[2 Pi s]], {s, 0, 15 * 10^6},
  PlotStyle -> Directive[Red, Thickness[0.01]], PlotRange -> {0, 4 * 10^(-7)},
  Ticks -> {{2. * 10^6, 6. * 10^6, 10. * 10^6, 14. * 10^6}, Automatic}];
p2 = Plot[Abs[ftrechteckwindow[2 Pi s]], {s, 0, 15 * 10^6},
  PlotStyle -> Directive[Blue, Thickness[0.003]], PlotRange -> {0, 4 * 10^(-7)},
  Ticks -> {{2. * 10^6, 6. * 10^6, 10. * 10^6, 14. * 10^6}, Automatic}];
p3 =
  Show[
    p1,
    p2];
p4 = Import["/home/rolf/Desktop/16qam2.webp"];
GraphicsRow[{p3, p4}]

```



Absolute values of the **spectral magnitude** of the **rectangle window** (blue) and of the **Raised Cosine window** (red), showing the far better damping of the RC window for increasing frequencies.

#### 16QAM Constellation Diagram

The I-Axis means the real parts, the Q-Axis the imaginary parts of complex numbers. For example, the bit group

**1001 is mapped to 3 - 1j.**

Note that **each single carrier**  $c_k e^{i k \omega_0 t}$  in a sent pulse (see below) with amplitude  $c_k$  **has the spectrum**  $c_k \mathcal{F}[\text{window}](\omega - k \omega_0)$  with  $\mathcal{F}$  the **Fourier transform**,  $\omega_0 = 2\pi/T$ . With regard to radiation and spectral efficiency, this demonstrates the relevance of pulse shaping with the time window.

Now, we consider two **typical spectral shapes of a single WLAN pulse**, which I generated with Mathematica according to the 16QAM mapping of groups of 4 bits each to a complex amplitude. I omit the somewhat lengthy Mathematica commands for this here. This pulse consists of **a single OFDM symbol**, which transmits with 16QAM an information package. The 802.11a/g standard *in the 2.4 GHz frequency band* with 16QAM uses 48 data subcarriers and 4 pilot subcarriers with carrier numbers  $(-21, -7, 7, 21)$ , with known complex amplitudes at the receiver. Thus, for the pulse a trigonometric polynomial is used with 52 equally spaced frequencies to transmit the data (192 bits of an **encoded** bit stream) in complex amplitudes  $c_k$  as described shortly above. The actual useful information bits are less than 192 in a pulse, because they are transmitted with a **code rate** for error correction redundancy. For example, a code rate of  $3/4$  means that out of four transmitted data bits, three actually contain useful data and the fourth bit is an error correction redundancy bit. The bitstream is usually encoded with convolutional coding as forward error correction (FEC). **Physically, the signals are voltage curves across time.**

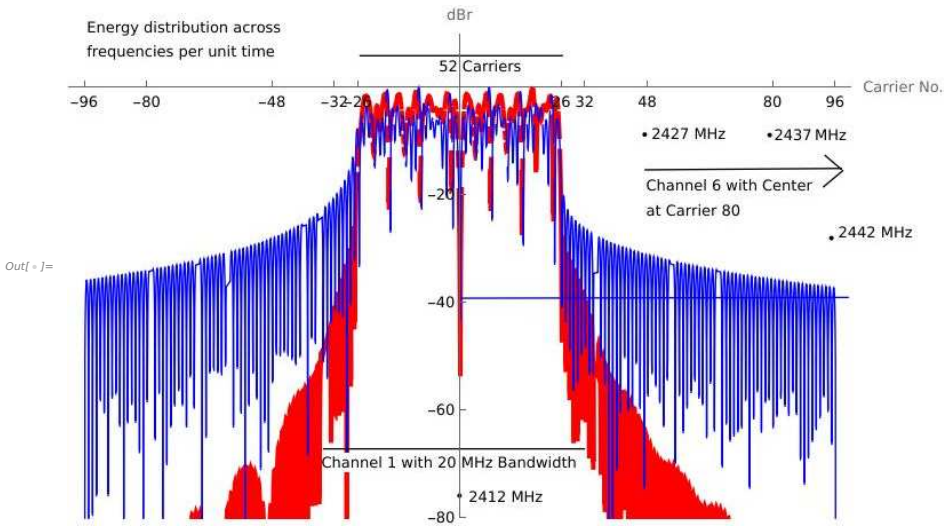
The total channel bandwidth is 20 MHz with an occupied bandwidth of 16.6 MHz for the overall data transmission. The subcarrier spacing is 312.5 kHz. The channel spacing is 25 MHz. Below you clearly see that the blue spectral energy distribution of a transmission with a rectangle time window has much more out-of-band emission than the red spectral energy distribution, where an **RC pulse shaping** is used.

**Thus, pulse shaping is a relevant topic in communication engineering to mitigate interferences.**

The channels 1, 6 or 11 are often used and suitably chosen by WLAN routers at home operating at 2.4 GHz with 20 MHz bandwidth, if several networks operate in a near neighborhood, because these channels hardly show mutual interferences. You can check this with your router at home. Some channels may be restricted in certain countries (e.g. the US does not permit channel 12 to prevent interference with other devices in the adjacent frequency band like satellite phones and other low-speed data communications).

The **allowed out-of-band radiations** are regulated in **IEEE specifications** with **spectral masks** for various transmission methods. The RC pulse shape, for example, fulfills the requirements for WLAN 802.11a/g, the pulse with a rectangle time window does not meet the requirements.

This is demonstrated in the following illustration. Compare the corresponding radiations of channel 1 at 2427 MHz, where the carriers of channel 6 start. The pulse shape with a rectangular time window in **blue**, that of an RC pulse shape in **red**. *The RC pulse shaping as shown here is **used in real systems***, for example, by Broadcom according to "L. Montreuil et al. (2013), Broadcom Recommendations for Tx Symbol Shaping". Broadcom is a supplier for digital communication devices of various providers and offers also PCI WiFi Cards.



### Supplementary explanations

1) The WLAN signal is an *energy signal*, it has *no power density spectrum* in the classical sense. In practice for transmission tests, a great number of OFDM signal pulses transmit a random 01-sequence. Their spectra are averaged to obtain an *approximation for a power spectral density (PSD)*. Such PSD diagrams are used by manufacturers to demonstrate compliance with regulations for out-of-band radiations in WLAN transmissions. A power spectral density can theoretically be introduced, if the *OFDM transmission is mathematically modeled as a stationary stochastic process* (Wiener-Chintchin-Theorem).

In PSD figures, the results are mostly shown in units **dBm/Hz relative to a total power**. Alternatively, as above with my single OFDM symbol spectrum, it can be shown in **dBr, which is normed so that the peak is at 0 dB**. The spectral masks in the **IEEE regulations are given in dBr**, because only the shape of the power density spectrum matters, when the out-of-band radiation is relevant. For example, in 802.11g for WLAN with OFDM the radiation must fulfill -28 dBr at 20 MHz offset and -40 dBr at 30 MHz offset from the center frequency. A pulse with the rectangle time window does not fulfill this requirement.

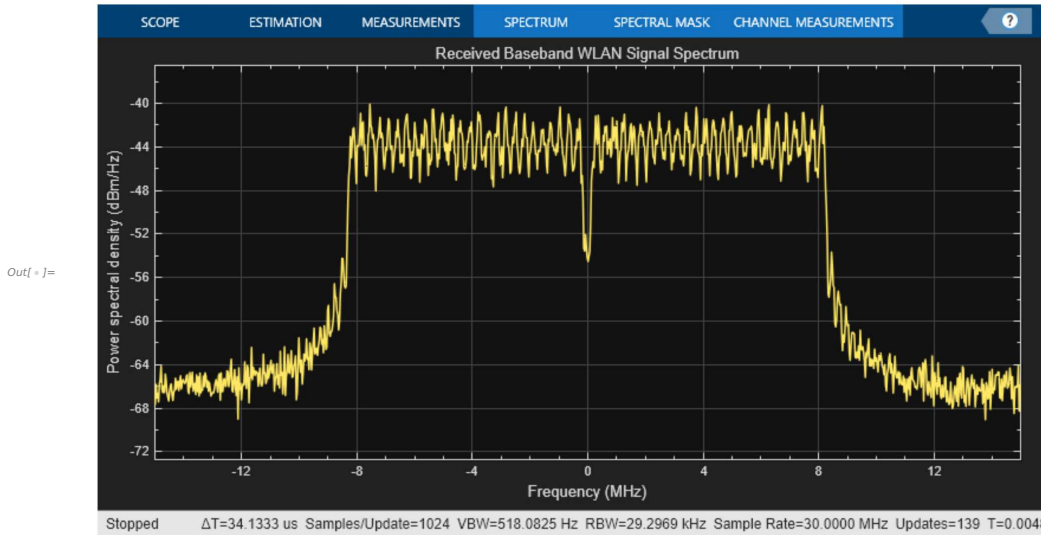
To give an example, how a PSD picture of such a measured average spectral power density comes about, we must know the total power to which the measurement is related. **The example below** shows dBm/Hz across frequencies with a peak level of about -40 dBm/Hz. The **total power P**, to which this measurement is related, is not shown. The unit is dBm/Hz for the bandwidth 2 MHz. The physical unit is  $V^2/Hz$ .

$$dBm = dBm/Hz + 10 \log_{10}[\text{bandwidth in Hz}], \quad dBm = 10 \log_{10}[P/1 \text{ mW}]$$

$$P = 10^{dBm/10} \cdot 1 \text{ mW},$$

i.e., for -40 dBm/Hz,  $dBm = 33.0103$ . We find  $P \approx 2000 \text{ mW}$ . A WLAN transmission device typically has a transmission power of 100 mW (20 dBm) in the 2.4 GHz band for the signal.

Here the example of such a spectral power density, presumably made with a rectangular time window. The used time window is not specified on the screen of this spectral analyzer, but we can compare with the previous graphic above to obtain this presumption.



2) Some relevant data on WLAN with OFDM and 16QAM in the 2.4 MHz band:

FFT length	64
Guard Interval (GI)	1/4
Data Carriers	48
Pilot Carriers	4

The **guard interval** defines a duration, which is available for a **cyclical extension of the signal** to the left as a so-called **cyclic prefix**. The cyclic prefix is realized as a copy of the last samples of the OFDM symbol that is prepended to the actual time signal samples.

The primary purpose of the cyclic prefix is to mitigate the effects of multipath propagation as delay spread that can cause inter-symbol interference (ISI) in wireless communication. By inserting the cyclic prefix, the receiver can tolerate a certain amount of delay in the received signal without introducing ISI (see [1], ch. 12 for a detailed example).

An important *advantage of a cyclic prefix* is that the convolution with *the impulse response*  $h$  of a time-invariant transmission channel can mathematically be represented as a **cyclic convolution**, if this impulse response does not last longer than the prefix. This allows for interference suppression *using the samples of the estimated channel frequency response*  $\mathcal{F}(h)$ , because the received signal  $r$  is the convolution

$$r = S * h + \text{additive noise}.$$

Thus without noise, division of the spectral values of  $r$ , obtained from the DFT of the received signal samples, by the corresponding values of  $\mathcal{F}(h)$  would directly give the sought signal amplitudes  $c_k$  of  $S$  (cf. [1], ch.12). However, a convolution equation of that type is an *illposed problem* (think of very small values of  $\mathcal{F}(h)$ ), and in real practice various modified estimation algorithms are used. For details on *spectral estimation* see, for example, [7] K. D. Kammeyer, K. Kroschel.

A **cyclic postfix** is analogously a copy of the first carriers of the OFDM symbol that is appended to the end of the symbol. While less common than the cyclic prefix, it can be used in conjunction with windowing techniques in some OFDM implementations, such as in some 5G waveforms. For more on this, see [1], ch. 12.

### 16QAM

Duration of an OFDM symbol  $4 \mu\text{s}$

Guard Interval Duration  $0.8 \mu\text{s}$

IDFT Period  $3.2 \mu\text{s}$

Data Carriers 48

802.11a with 20 MHz channel bandwidth uses 64 carriers, 48 of which are reserved for data, and 4 for pilot tones.

Code Rate  $1/2$  or  $3/4$

Maximal Bit Rate 24 Mbps with Code Rate  $1/2$

$[(48 \text{ number of carriers} \times 4 \text{ bits per carrier} \times 1/2 \text{ code rate}) / (4 \mu\text{s symbol duration})] = 24 \text{ Mbps}$

Maximal Bit Rate 36 Mbps with Code Rate  $3/4$

$[(48 \times 4 \times 3/4) / 4] = 36$

With 64QAM und code rate  $3/4$ , the maximal bit rate is accordingly 54 Mbps, often offered in DSL contracts, that are also based on OFDM transmission technology.

Code rates  $< 1$  come from coding the bitstream of useful data with error-correcting codes (convolution codes). A code rate of  $3/4$  means that out of four transmitted data bits, three actually contain useful data and the fourth bit is an error correction redundancy bit.

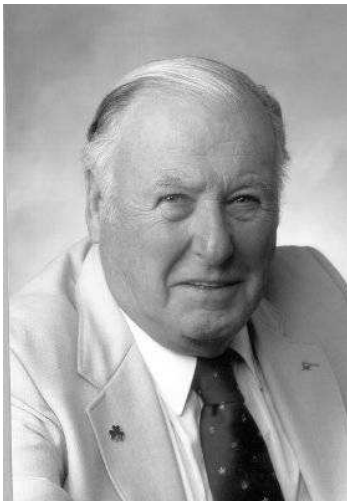
**Summary.** In that application example, widely used in digital devices of our everyday live, I have tried to explain only the basic principle of the physical layer of an OFDM transmission. There are many topics that need to be mastered for practical real-time transmission within a few  $\mu\text{s}$  per symbol with OFDM or modifications of the procedure (OFDMA, COFDM, FBMC, GFDM, etc.). These include, in particular, *peak reduction* (with many equal amplitudes in the OFDM symbols), *peak-to-average power ratio reduction* (PAPR), *channel equalization* with multiple frequency-selective channels (Doppler effects with moving transmitters or receivers causing frequency dispersion), *denoising*, and *many others*. Despite mathematically simple principles, it is a long way to a robust technology, demanding high skill from engineers and computer scientists. For a detailed treatment of communication engineering, [4] L. W. Couch (2012), “Digital and Analog Communication Systems” can be a helpful reference. Since almost everything in the presented transmission can be done with discrete signal processing, algorithms can be developed once and then distributed millions of times without significant additional costs. This allows for low acquisition costs.



In implementations for computing a DFT usually the fast FFT algorithm is used. It reduces the number of computations for an  $N = 2^n$  point DFT from  $N^2$  to  $N \log_2(N)$ . Thus it yields a significant reduction of computation time. The algorithm, along with its recursive application, was invented by Carl Friedrich Gauss around 1805. Cooley and Tukey independently rediscovered and popularized it 160 years later. For details see [1], ch. 6, or other references on the topic. For a DFT and IDFT, *in Mathematica the FFT is used as default*.

The FFT was one of the Top 10 “*Algorithms of the Century*”.

Finally you see two images of the authors of the Fast Fourier Transform algorithm (FFT), one of the top 10 “Algorithms of the Century” as listed by the IEEE Computer Society Journal.



James W. Cooley (1926 - 2016)



John Tukey (1915 - 2000)

### Example 14. Windowed Fourier transform with a DFT, Spectrograms

In its classical form, the Fourier transform  $\mathcal{F}$  does not allow for simultaneous time-frequency analysis. For example, speech or a piece of music in our everyday experience has a specific “time pattern” and at the same time a specific “frequency pattern”. However, the spectral function of a signal does not show at what times and with what respective amplitudes a specific angular frequency  $\omega$  occurs in a signal  $f$ , but rather accumulates contributions of the same angular frequency  $\omega$  over the entire time course of  $f$  in  $\mathcal{F}f(\omega)$ . Dennis Gabor (1900-1979) already noticed these disadvantages for signal processing purposes, and in 1946 in his work “Theory of Communication”, he proposed time-frequency localization through Fourier transforms with window functions.

To obtain information about the “time-frequency pattern” of a signal, one determines not the spectral function of the entire signal, but the spectral functions for time segments of  $f$ . Time segments of a signal  $f$  are obtained by multiplying  $f$  with functions of finite effective duration. Such functions are referred to as window functions or time windows as considered above. We consider the following example.

A short-term model for a siren is approximately the function or chirp  $f(t) = A \sin(g(t))$  with  $g(t) = 2\pi t(\alpha + \beta t^2)$  for  $0 \leq t \leq 10$  s and constants  $A, \alpha, \beta$ . The derivative of the argument  $g'(t) = 2\pi t(2\alpha + 3\beta t)$  can be considered as the instantaneous angular frequency at time  $t$ . The magnitude spectrum, approximately calculated with a DFT for parameters  $A=1, \alpha=4[1/s^2], \beta=-4/15 [1/s^3]$  over  $T=10$  s, shows a multitude of frequencies up to the maximum frequency 20 Hz, but not the parabolic frequency modulation and not the instantaneous frequencies at different times (left image below). The graph of an approximation for the windowed Fourier transform of  $f$  with the “Hann window”  $w(t) = 0.5 - 0.5 \cos(2\pi t/T)$  for  $0 \leq t \leq T=1$  s, on the other hand, clearly shows the rise and fall of the instantaneous frequencies and corresponds to our usual impression of the variable frequency of the siren tone (right image). The calculations used a 512-point DFT over a total of  $T=10$  s, with the DFT coefficients  $C_k T$  plotted as approximations for  $\mathcal{F}f(2\pi k/T)$  in the first image. In the second case, 50 Hann windows of duration 1 s were used at intervals of 0.2 s each. Per time segment, a 128-point DFT was performed and the resulting (single-sided) DFT magnitude spectra were combined to form the second image. Neither representation shows the constant amplitude  $A=1$ . One reason is the strong aliasing effects due to the frequency modulation. The sum of the  $|C_k|^2$  of the first image agrees numerically very well with the quadratic mean of  $f$  in  $[0, T]$  (in both cases, the value is about 0.5). Numerical integration to calculate the windowed Fourier transform for 20 Hz at  $t_0 = 5$  s results in approximately 0.24, as shown in the following spectrogram on the right. The signal values (and thus  $A$ ) can only be approximately recovered from the DFT using an interpolation polynomial or the formula for discrete reconstruction from the data (for more details please see [1], 12.5). Now to the images:

In[ ]:=

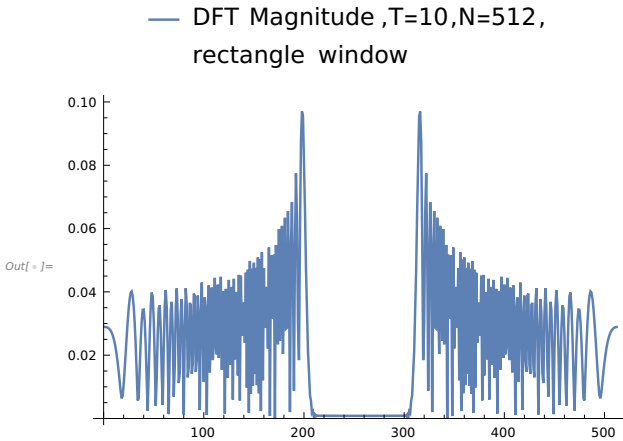
```
ClearAll["Global`*"]
```

```

In[ ]:= B = 1;
M = 128;
NN = 50;
h[x_] = UnitStep[x];
w[x_] = (0.5 - 0.5 Cos[2 Pi B x]) * (h[x] - h[x - 1/B]);
f[x_] = Sin[2 Pi 20 x (2/10 x^2 - 1/75 x^3)]
data0 = Table[f[10 n/512], {n, 0, 511}];
dft = Chop[Fourier[data0, FourierParameters -> {-1, -1}]];
pdft = ListLinePlot[Abs[dft], PlotRange -> All,
  PlotLegends -> Placed[{"DFT Magnitude ,T=10,N=512,
rectangle window"}, Above]]
data1[k_, j_] = N[w[j/(B M)] x f[k*0.2 + j/(B M)]];
FT1[k_, n_] := N[1/M Sum[data1[k, j] Exp[-2 Pi I n j/M], {j, 0, M-1}]];
z[k_, n_] := N[Abs[FT1[k, n]]];
data2 = Table[z[k, n], {n, 1, 25}, {k, 0, NN-1}];
pwindowed =
  ListPlot3D[data2, PlotRange -> All, Mesh -> 100, Axes -> {True, True, True},
    Boxed -> False, AxesLabel -> {"Window No.,Time in s =
Window No. x 0.2s", "Hz", "Magnitude"},
    AxesStyle -> Directive[Black, Plain, 10],
    PlotStyle -> Directive[PlotPoints -> 100], ViewPoint -> {30, -40, 50},
    AxesEdge -> {{-1, -1}, {1, -1}, {-1, -1}},
    Ticks -> {{10, 40}, {10, 20, 25}, {0.0, 0.2}},
    PlotLegends -> Placed[{"3D Spectrogram , Windowed Fourier Transform ,
50 Hann Windows in T=10s"}, Above]];

```

$$\text{Out[ ]} = \sin\left[40\pi\left(\frac{x^2}{5} - \frac{x^3}{75}\right)\right]$$

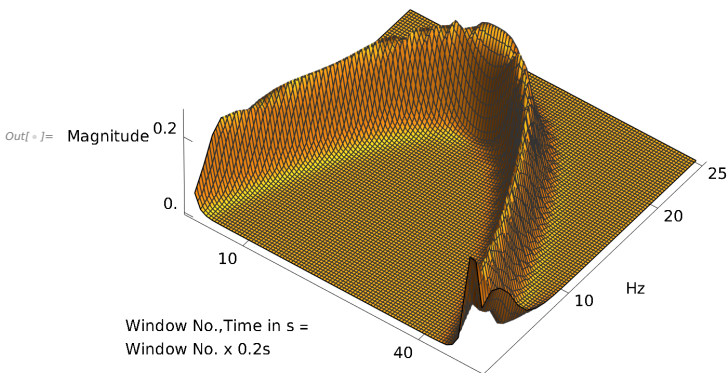


Now the 3D illustration of the windowed Fourier transform showing the time-frequency pattern of the signal.

**The illustration is also called a spectrogram.**

`In[ ]:= Show[pwindowed]`

■ 3D Spectrogram , Windowed Fourier Transform ,  
50 Hann Windows in T=10s



In Mathematica you can illustrate spectrograms in a 2D image with the **command Spectrogram for a list of sampled data**. For tests this is left to the reader. Instead we make a 2D representation ourselves with MatrixPlot for our list data2:

```

In[ ]:= mplot = MatrixPlot[data2, PlotLegends → True,
    Axes → True, FrameLabel → {"Hz", "Window No."},
    DataReversed → {True, False}, ColorFunction → "CMYKColors"];

```

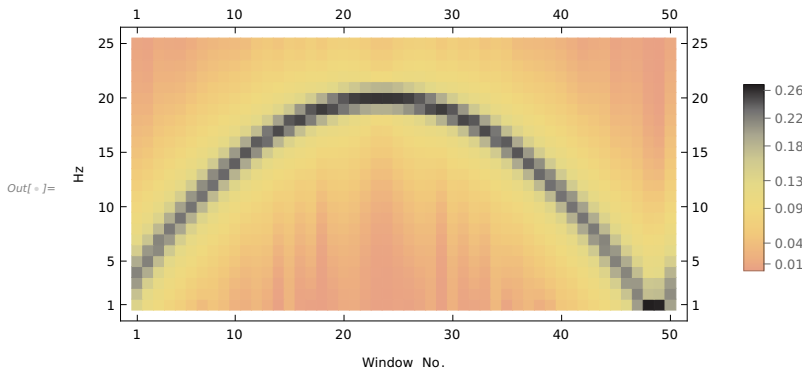
2 D Spectrogram of the siren signal, time  $t = \text{window number} \cdot 0.2\text{s}$ .

It could be sharpened with more sampling points.

```

In[ ]:= Show[mplot]

```



### 3.6 The Discrete Cosine Transforms DCT I and DCT II

Widely used variants of the DFT for real - valued functions are the discrete Cosine transforms DCT I and DCT II.

#### 1. The discrete Cosine Transforms DCT I

We assume a continuous, piecewise continuously differentiable real-valued function  $f$  on  $[0, T]$ , which we think of as being extended to an even  $2T$ -periodic function  $f_p$  on the line, and consider samples  $y_n$  of  $f_p$  with the symmetry  $y_n = y_{-n}$ .

With  $N=2m$  samples  $y_n = f_p(nT/m)$  for  $n = -m+1, \dots, m$ , we obtain for the DFT coefficients  $a_k = C_k + C_{-k} = 2 C_k$  of  $f_p$  and  $0 \leq k \leq m$ , due to the symmetry

$$y_n = y_{n \pm 2m} = f_p(nT/m \pm 2T) \quad \text{and the relation} \\ \text{Exp}[-i \pi k n / m] = \text{Exp}[i \pi k (n \pm 2m) / m]$$

the DCT I and its inverse (see also [1], chapter 6 for more details). The IDCT can directly be seen from the interpolation polynomial  $P_2$  above, because

$$y_n = P_2(nT/m) \quad \text{with } \omega_0 = \pi/T \text{ for the } 2T\text{-periodic } f_p.$$

$$\text{DCT I} \quad a_k = \frac{2}{m} \left( \frac{y_0}{2} + \sum_{n=1}^{m-1} y_n \cos(\pi k n / m) + \frac{y_m}{2} \cos(k\pi) \right), \quad k=0, \dots, m$$

$$\text{IDCT I} \quad y_n = \frac{a_0}{2} + \sum_{k=1}^{m-1} a_k \cos(\pi k n / m) + \frac{a_m}{2} \cos(n\pi), \quad n=0, \dots, m$$

Before showing applications of the DCT, we turn to another option and consider interpolation with a shifted set of nodes in comparison. This case results in the variant known as DCT II, which is particularly widespread in DCT applications. Here we describe simply the result and refer to [1], chapter 6 for more details and proofs.

## 2. The discrete Cosine Transforms DCT II

As before, we assume a given continuous, piecewise continuously differentiable real-valued even  $2T$ -periodic function  $f_p$ . However, we now choose a shifted set of nodes  $t_n$ , at which the samples are taken:

$$t_n = (2n+1)T/(2m), \quad 0 \leq n \leq 2m-1.$$

Using the given symmetry properties we achieve after some calculation (see [1], 6.4) the corresponding real-valued trigonometric interpolation polynomial  $P_3$  from the formula for  $P_2$  above as

$$P_3(t) = \frac{a_0}{2} + \sum_{k=1}^{m-1} a_k \cos(\omega_0 k t) \quad \text{with } \omega_0 = \pi/T, \quad y_n = f_p(t_n) \quad \text{and the DCT II is}$$

$$\text{DCT II:} \quad a_k = \frac{2}{m} \sum_{n=0}^{m-1} y_n \cos(\pi k (2n+1)/(2m)), \quad k=0, \dots, m-1.$$

**IDCT II:** We can immediately recognize the inverse IDCT II from  $P_3$ .

$$y_n = \frac{a_0}{2} + \sum_{k=1}^{m-1} a_k \cos(\pi k (2n+1)/(2m)) \quad n=0, \dots, m-1.$$

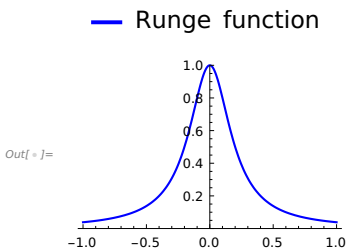
The DCT II in its 2D variant is widely used in image processing as for JPEG compression algorithms. It is also closely connected with interpolation by Chebyshev polynomials as we will see in the subsequent section 3.7.

### 3.7 Interpolation with the DCT I and DCT II

As an example we consider the so-called Runge function  $\text{runge}[t] = 1/(1+25 t^2)$  in the interval  $[-1,1]$ .

It is infinitely often differentiable on the entire reals, but its power series has only convergence radius  $1/\sqrt{5}$  due to the poles at  $\pm 1/\sqrt{5} i$ . We make first a trigonometric interpolation with the DCT I and then an interpolation with the DCT II with shifted nodes. This is a remarkable example, because polynomial interpolation with equidistant nodes give very bad approximations. We will see this in the subsequent section 3.9, where Chebyshev polynomials with Chebyshev nodes are used for interpolation. There the coefficients for the interpolation polynomials are also obtained with a DCT.

```
In[ ]:= runge[t_] = 1/(1 + 25 t^2);
plot20 = Plot[runge[t], {t, -1, 1}, PlotRange -> {0, 1},
  PlotLegends -> Placed[{"Runge function"}, Above],
  PlotStyle -> {Blue, Thickness[0.008]}, ImageSize -> Small]
```



1) Now a DFT and equivalently a DCT I with an even number  $NN$  of samples. As already above, we have to correct the phases in the DFT spectrum, because we take the samples in  $[-1,1]$ . The resulting spectrum must be real and even (see [1], 4.1.)

```
In[ ]:= T = 2; NN = 16;
data20 = Table[runge[-1 + T k/NN], {k, 0, NN - 1}];
dataphase20 = Table[(-1)^k, {k, 0, NN - 1}];
dft20 = dataphase20 * Chop[Fourier[data20, FourierParameters -> {-1, -1}]];
```

Here the uniquely determined real trigonometric interpolation polynomial with the maximal frequency in the above defined vector space  $V_{NN/2}$ . Above, this approximation was called  $P_2(t)$  with  $NN/2=m$ .

In the example we have the maximal frequency 4 Hz. We plot the Runge function and this approximation in Red with a small offset +0.03 for a better visibility.

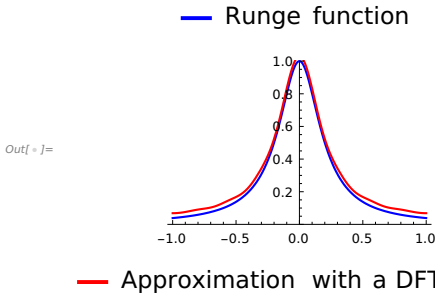
```

In[ ]:= rungeapprox1 [t_] =
  FullSimplify [Sum[dft20[[k + 1]] Exp[I * 2 Pi / T k t], {k, 0, NN / 2 - 1}] +
    Sum[dft20[[k + 1]] Exp[I * 2 Pi / T * (k - NN) t], {k, NN / 2 + 1, NN - 1}]] +
    dft20[[NN / 2 + 1]] Cos[NN / 2 * 2 Pi / T * t]
offset =
  0.03;
Out[ ]:= 0.274611 + 0.344365 Cos[π t] + 0.175654 Cos[2 π t] +
  0.0972947 Cos[3 π t] + 0.050132 Cos[4 π t] + 0.0285903 Cos[5 π t] +
  0.0149957 Cos[6 π t] + 0.0105194 Cos[7 π t] + 0.00383778 Cos[8 π t]

In[ ]:= plot21 = Plot[rungeapprox1 [t] + offset, {t, -1, 1},
  PlotRange → {0, 1.1}, PlotStyle → {Red, Thickness [0.008]},
  PlotLegends → Placed[{"Approximation with a DFT/DCT I"}, Below]];

In[ ]:= Show[{plot20, plot21}]

```



This DFT approximation is the same as with a DCT I with  $NN/2+1=9$  samples in the interval  $[0,1]$  of the form  $t_k = k/m$  for  $k=0, \dots, m$  with  $m=NN/2$ .

**2) Now an interpolation by a DCT II** with an even number of samples. The nodes are  $t_k = (2n+1)/NN$  for  $0 \leq n \leq NN/2-1$ . We use here the DCT II, which is implemented in Mathematica with the command `FourierDCT[list, 2]` and choose  $NN2=NN+2$ ,  $m=NN2/2$ ,  $NN=16$  from before to obtain the same maximal frequency as above. The resulting interpolation function is  $P_3(t)$  in the notation from above and in [1], 6.6.

```

In[ ]:= NN2 = NN + 2; m = NN2 / 2; (* now take the samples in [0,1] *)
list = Table[runge[(2 n + 1) / (2 m)], {n, 0, m - 1}];
dct2 = 1 / Sqrt[m] FourierDCT [list, 2];
rungeapprox2 [t_] = dct2[[1]] + Sum[2 dct2[[k]] Cos[(k - 1) π t], {k, 2, m}]

```

```

Out[ ]:= 0.27471 + 0.344011 Cos[π t] + 0.175799 Cos[2 π t] +
  0.0967712 Cos[3 π t] + 0.050008 Cos[4 π t] + 0.0274507 Cos[5 π t] +
  0.0138577 Cos[6 π t] + 0.00727322 Cos[7 π t] + 0.00286974 Cos[8 π t]

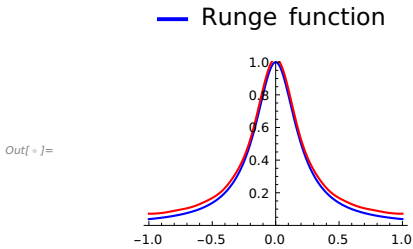
```



```

In[ ]:= offset = 0.03;
plot22 = Plot[rungeapprox2 [t] + offset, {t, -1, 1},
  PlotRange → {0, 1.1}, PlotStyle → {Red, Thickness[0.008]},
  PlotLegends → Placed[{"Approximation with a DCT II"}, Below]];
Show[{plot20, plot22}]

```



— Approximation with a DCT II

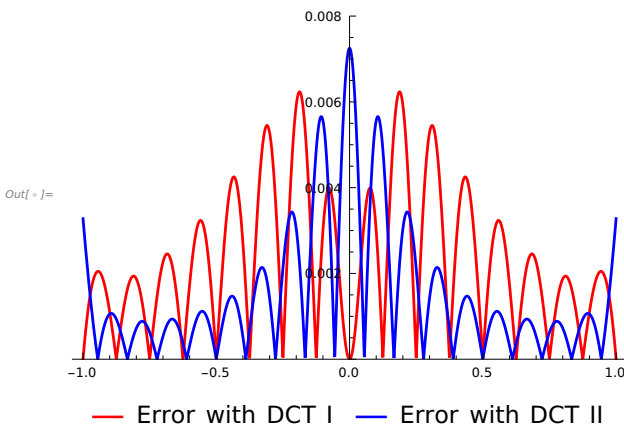
Subsequently the absolute **approximation errors** of both trigonometric polynomials for a comparison. The zeros of the error curves are the positions, where the Runge function is interpolated. In both cases we have 9 those nodes in  $[0, 1]$ .

```

In[ ]:= errordct1 [t_] = Abs[runge[t] - rungeapprox1 [t]];
errordct2 [t_] = Abs[runge[t] - rungeapprox2 [t]];

In[ ]:= plot23 = Plot[errordct1 [t], {t, -1, 1},
  PlotRange → {0, 0.008}, PlotStyle → {Red, Thickness[0.005]},
  PlotLegends → Placed[{"Error with DCT I"}, Below]];
plot24 = Plot[errordct2 [t], {t, -1, 1}, PlotRange → {0, 0.008},
  PlotStyle → {Blue, Thickness[0.005]},
  PlotLegends → Placed[{"Error with DCT II"}, Below]];
Show[{plot23, plot24}]

```



### 3.8 Application of the DCT I in Numerical Integration, Clenshaw-Curtis Quadrature

In [1], 6.5 the Clenshaw-Curtis quadrature is derived. We will consider it here as an application of a DCT I with the example

$$f[t_] = e^t + 3 \cos[24 t] - t^6 \text{ on } [-1, 1].$$

The following representation shall give readers already a first impression of the use of vectors and matrices in Mathematica.

We choose as example  $m=12$  for  $2m+1$  used samples of  $f$  for the algorithm.

As shown in [1], with the Clenshaw-Curtis quadrature instead of  $f(\cos(\phi))\sin(\phi)$  on  $[0, \pi]$  a trigonometric approximating polynomial for the factor  $f(\cos(\phi))$  on  $[0, \pi]$  is integrated, which is obtained by a DCT I and has the maximal angular frequency  $2m$ . With that, an approximation  $SN(f)$  for the desired integral of  $f$  over  $[-1, 1]$  is computed.

In [1] it is shown that one can use a DCT I with only  $m+1$  samples for the computation of the weights  $w_n$  in the quadrature formula

$$SN(f) = \sum_{n=0}^m w_n (f(x_n) + f(-x_n))$$

with  $x_n = \cos[n\pi/(2m)]$ ,  $0 \leq n \leq m$ .

For the computation in the example we define as in [1], 6.5 the  $(m+1) \times (m+1)$  DCT I matrix and proceed as described there.

With the subsequently defined vector  $b$  for the computation of the weights we need the transpose of the DCT I matrix, which is called `dctmatrixtransposed`.

```
In[ ]:= m = 12;
dctpart1 = Table[1 / m Cos[π (k - 1) (n - 1) / m], {k, m + 1}, {n, 2, m}];
dctpart2 = Map[Prepend[#, 1 / (2 m)] &, dctpart1];
vector = Table[1 / (2 m) Cos[π (k - 1)], {k, m + 1}];
dctmatrixtransposed = Append[Transpose[dctpart2], vector];
(* with dctmatrixtransposed //MatrixForm you can see the usual matrix form as
   output. But dont do that here in the definition of dctmatrixtransposed ,
   because otherwise matrix multiplication with b below does not work *)

In[ ]:= dctmatrixtransposed // MatrixForm
```

Out[ ]:= //MatrixForm=

$$\begin{pmatrix} \frac{1}{24} & \frac{1}{24} & \frac{1}{24} & \frac{1}{24} & \frac{1}{24} & \frac{1}{24} & \frac{1}{24} & \frac{1}{24} & \frac{1}{24} & \frac{1}{24} & \frac{1}{24} & \frac{1}{24} & \frac{1}{24} \\ \frac{1}{12} & \frac{\sqrt{3}+1}{24\sqrt{2}} & \frac{1}{8\sqrt{3}} & \frac{1}{12\sqrt{2}} & \frac{1}{24} & \frac{\sqrt{3}-1}{24\sqrt{2}} & 0 & -\frac{\sqrt{3}-1}{24\sqrt{2}} & -\frac{1}{24} & -\frac{1}{12\sqrt{2}} & -\frac{1}{8\sqrt{3}} & -\frac{\sqrt{3}+1}{24\sqrt{2}} & -\frac{1}{12} \\ \frac{1}{12} & \frac{1}{8\sqrt{3}} & \frac{1}{24} & 0 & -\frac{1}{24} & -\frac{1}{8\sqrt{3}} & -\frac{1}{12} & -\frac{1}{8\sqrt{3}} & -\frac{1}{24} & 0 & \frac{1}{24} & \frac{1}{8\sqrt{3}} & \frac{1}{12} \\ \frac{1}{12} & \frac{1}{12\sqrt{2}} & 0 & -\frac{1}{12\sqrt{2}} & -\frac{1}{12} & -\frac{1}{12\sqrt{2}} & 0 & \frac{1}{12\sqrt{2}} & \frac{1}{12} & \frac{1}{12\sqrt{2}} & 0 & -\frac{1}{12\sqrt{2}} & -\frac{1}{12} \\ \frac{1}{12} & \frac{1}{24} & -\frac{1}{24} & -\frac{1}{12} & -\frac{1}{24} & \frac{1}{24} & \frac{1}{12} & \frac{1}{24} & -\frac{1}{24} & -\frac{1}{12} & -\frac{1}{24} & \frac{1}{24} & \frac{1}{12} \\ \frac{1}{12} & \frac{\sqrt{3}-1}{24\sqrt{2}} & -\frac{1}{8\sqrt{3}} & -\frac{1}{12\sqrt{2}} & \frac{1}{24} & \frac{\sqrt{3}+1}{24\sqrt{2}} & 0 & -\frac{\sqrt{3}+1}{24\sqrt{2}} & -\frac{1}{24} & \frac{1}{12\sqrt{2}} & \frac{1}{8\sqrt{3}} & -\frac{\sqrt{3}-1}{24\sqrt{2}} & -\frac{1}{12} \\ \frac{1}{12} & 0 & -\frac{1}{12} & 0 & \frac{1}{12} & 0 & -\frac{1}{12} & 0 & \frac{1}{12} & 0 & -\frac{1}{12} & 0 & \frac{1}{12} \\ \frac{1}{12} & -\frac{\sqrt{3}-1}{24\sqrt{2}} & -\frac{1}{8\sqrt{3}} & \frac{1}{12\sqrt{2}} & \frac{1}{24} & -\frac{\sqrt{3}+1}{24\sqrt{2}} & 0 & \frac{\sqrt{3}+1}{24\sqrt{2}} & -\frac{1}{24} & -\frac{1}{12\sqrt{2}} & \frac{1}{8\sqrt{3}} & \frac{\sqrt{3}-1}{24\sqrt{2}} & -\frac{1}{12} \\ \frac{1}{12} & -\frac{1}{24} & -\frac{1}{24} & \frac{1}{12} & -\frac{1}{24} & -\frac{1}{24} & \frac{1}{12} & -\frac{1}{24} & -\frac{1}{24} & \frac{1}{12} & -\frac{1}{24} & -\frac{1}{24} & \frac{1}{12} \\ \frac{1}{12} & -\frac{1}{12\sqrt{2}} & 0 & \frac{1}{12\sqrt{2}} & -\frac{1}{12} & \frac{1}{12\sqrt{2}} & 0 & -\frac{1}{12\sqrt{2}} & \frac{1}{12} & -\frac{1}{12\sqrt{2}} & 0 & \frac{1}{12\sqrt{2}} & -\frac{1}{12} \\ \frac{1}{12} & -\frac{1}{8\sqrt{3}} & \frac{1}{24} & 0 & -\frac{1}{24} & \frac{1}{8\sqrt{3}} & -\frac{1}{12} & \frac{1}{8\sqrt{3}} & -\frac{1}{24} & 0 & \frac{1}{24} & -\frac{1}{8\sqrt{3}} & \frac{1}{12} \\ \frac{1}{12} & -\frac{\sqrt{3}+1}{24\sqrt{2}} & \frac{1}{8\sqrt{3}} & -\frac{1}{12\sqrt{2}} & \frac{1}{24} & -\frac{\sqrt{3}-1}{24\sqrt{2}} & 0 & \frac{\sqrt{3}-1}{24\sqrt{2}} & -\frac{1}{24} & \frac{1}{12\sqrt{2}} & -\frac{1}{8\sqrt{3}} & \frac{\sqrt{3}+1}{24\sqrt{2}} & -\frac{1}{12} \\ \frac{1}{24} & -\frac{1}{24} & \frac{1}{24} & -\frac{1}{24} & \frac{1}{24} & -\frac{1}{24} & \frac{1}{24} & -\frac{1}{24} & \frac{1}{24} & -\frac{1}{24} & \frac{1}{24} & -\frac{1}{24} & \frac{1}{24} \end{pmatrix}$$

Now the needed vector **b**, and with that the computation of the vector **w** of weights for the quadrature. They all are positive with the sum equal to 1.

In[ ]:= beta = Table[2, {k, m + 1}]; beta[[1]] = 1; beta[[m + 1]] = 1;

b = Table[beta[[k]] / (1 - 4 (k - 1)^2), {k, m + 1}];

w = N[dctmatrixtransposed . b]

Out[ ]:= {0.00173913, 0.0166755, 0.0340258, 0.0500188, 0.0654954, 0.0796553, 0.0925836, 0.103831, 0.113378, 0.120922, 0.126452, 0.129768, 0.0654559}

In[ ]:= Sum[w[[k]], {k, 1, m + 1}]

Out[ ]:= 1.

Finally the necessary samples of **f** and the numerical integration :

In[ ]:= f[t\_] = Exp[t] + 3 Cos[24 t] - t^6;

y = Table[N[f[Cos[(k - 1) π / (2 m)]] + f[-Cos[(k - 1) π / (2 m)]]], {k, 1, m + 1}]

Out[ ]:= {3.63124, 2.55048, -0.84054, -4.22864, -0.183221, 8.05541, 0.451138, -0.453541, 7.28713, -3.68553, 8.05142, -3.98271, 8.}

**Result: The value SN by the Clenshaw-Curtis quadrature is**

In[ ]:= SN = w.y

Out[ ]:= 1.83855

**For comparison: You can exactly solve the integral**

```
In[ * ]:= integral = Integrate[f[t], {t, -1, 1}]
```

```
Out[ * ]= - $\frac{2}{7}$  -  $\frac{1}{e}$  +  $e$  +  $\frac{\text{Sin}[24]}{4}$ 
```

```
In[ * ]:= N[%, 16]
```

```
Out[ * ]= 1.838293511071661
```

The reason for the difference of the integral values is the fixed number  $2m+1$  of samples we worked with, while Mathematica iterates to get a certain precision. The quadrature with  $N+1=2m+1$  nodes is exact for polynomials up to the degree  $N$ . For the computation of the weights a DCT I of length  $m+1$  is sufficient.

#### An example for an little program with Mathematica with the Clenshaw-Curtis quadrature

A program in Mathematica is a **Module**, which we here call `ccq[g,{a,b},m,opt]` for integration of  $g:[a,b] \rightarrow \mathbb{R}$  with  $m$  as above, i.e., polynomials up to degree  $2m$  are exactly integrated. The parameter `opt` is the number of decimals in the result.

```
In[ * ]:= ccq[g_, {a_, b_}, m_, opt_] :=
Module[{f, x, dctpart1, dctpart2, dctmatrixtransposed, beta, be, w, y, dez},
  f[x_] = g[(b - a)/2 x + (a + b)/2] (b - a)/2;
  (* hier wird Substitution benutzt *)
  dctpart1 = Table[1/m Cos[π (k - 1) (n - 1)/m], {k, m + 1}, {n, 2, m}];
  dctpart2 = Map[Prepend[#, 1/(2 m)] &, dctpart1];
  vector = Table[1/(2 m) Cos[π (k - 1)], {k, m + 1}];
  dctmatrixtransposed = Append[Transpose[dctpart2], vector];
  beta = Table[2, {k, m + 1}]; beta[[1]] = 1; beta[[m + 1]] = 1;
  be = Table[beta[[k]]/(1 - 4 (k - 1)^2), {k, m + 1}];
  dez = opt; w = N[dctmatrixtransposed.be, dez];
  y = Table[N[f[Cos[(k - 1) π/(2 m)]] +
    f[-Cos[(k - 1) π/(2 m)]]], dez], {k, 1, m + 1}];
  w.y]
```

**Example 15.** We choose  $g$  as the above integrated function on  $[-1,1]$ , but this time with  $m=24$ , i.e., 49 samples for the numerical integration. thus we achieve - except the last decimal - the same result as in the computation above from the exact solution with 16 decimals.

```
In[ * ]:= g = Function[{t}, Exp[t] + 3 Cos[24 t] - t^6];
```

```
ccq[g, {-1, 1}, 24, 16]
```

```
Out[ * ]= 1.838293511071663
```

**Example 16.** We integrate the polynomial  $p[x] = x^3 - 3x^{12} + 10x^{15}$  on  $[1,3]$  with  $m=8$  and  $\text{opt}=20$ .

Since  $p$  is a polynomial of degree less than 16, this quadrature is exact.

```
In[ ]:= p[x_] = x^3 - 3 x^12 + 10 x^15;
ccq[p, {1, 3}, 8, 20]
```

```
Out[ ]:= 2.6536299538461538462 × 107
```

We test the result by exact integration and let Mathematica show a decimal approximation with again 20 decimals.

```
In[ ]:= Integrate[p[x], {x, 1, 3}]
N[%, 20]
```

```
Out[ ]:= 
$$\frac{344971894}{13}$$

```

```
Out[ ]:= 2.6536299538461538462 × 107
```

### 3.9 Application of the DCT in Interpolation with Chebyshev Polynomials

The Chebyshev polynomial  $T[n,x]$  of the first kind with degree on  $[-1,1]$  for  $n \geq 0$  is defined by

$$T[n,x] = \cos[n \arccos[x]].$$

With the trigonometric addition theorems for the cosine you quickly find the recursion equation ( $n=1,2,\dots$ )

$$T[n+1,x] = 2x T[n,x] - T[n-1,x].$$

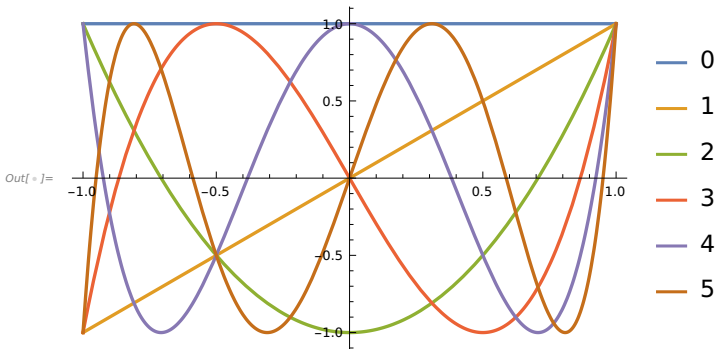
This shows that  $T[n,x]$  is a polynomial of degree  $n$  and as that is defined on entire  $\mathbb{R}$  and  $\mathbb{C}$ . For even  $n$  the  $T[n,x]$  are even, for odd  $n$  they are odd functions and always it holds  $|T[n,x]| \leq 1$  auf  $[-1,1]$ . The coefficient  $a_n$  of  $a_n x^n$  in  $T[n,x]$  is  $2^{n-1}$ . Since  $\cos[nx]$  in  $[0,\pi]$  has exactly  $n$  zeros, the polynomial  $T[n,x]$  has exactly  $n$  real zeros in  $[-1,1]$ . These zeros  $x[k]$  are  $x[k] = \cos[(2k-1)\pi/(2n)]$  für  $k=1,\dots,n$ .

Mathematica knows the Chebyshev polynomials as  $\text{ChebyshevT}[n,x]$ . We plot the first 5 of them and observe their symmetry properties and zeros.

```

In[ ]:= Plot[Evaluate[Table[ChebyshevT[n, x], {n, 0, 5}]], {x, -1, 1},
  PlotStyle → Directive[Hue, Thickness[0.006]],
  PlotRange → All, PlotLegends → {0, 1, 2, 3, 4, 5}]

```



### Chebyshev Polynomials as an orthogonal system

The considered polynomials build an orthogonal system with respect to the inner product

$$\langle f, g \rangle_w = \int_{-1}^{+1} f(x) g(x) / (\sqrt{1-x^2}) dx$$

with the weight function  $w(x) = 1 / (\sqrt{1-x^2})$ , considered for all real-valued functions  $f$  and  $g$  on  $[-1, 1]$  for which  $\langle f, f \rangle_w = \|f\|_w^2$  and  $\langle g, g \rangle_w = \|g\|_w^2$  exist. As usual we identify functions  $f$  and  $g$  with  $\|f - g\|_w = 0$  and denote the vector space of the according equivalence classes as Hilbert space  $H = L_w^2([-1, 1])$ . The Chebyshev polynomials  $T[n, x]$  build a complete orthogonal system in that space. We show the first 3 of them in unnormalized form.

```

In[ ]:= T0 = ChebyshevT[0, x]
        T1 = ChebyshevT[1, x]
        T2 = ChebyshevT[2, x]

```

```
Out[ ]:= 1
```

```
Out[ ]:= x
```

```
Out[ ]:= -1 + 2 x^2
```

Now with according norming so that  $\|T[n, x]\|_w = 1$ , here for  $n=0, 1, 2$ .

```
In[ ]:= T0n[x_] = ChebyshevT [0, x] / Sqrt[Pi]
T1n[x_] = Sqrt[2 / Pi] ChebyshevT [1, x]
T2n[x_] = Sqrt[2 / Pi] ChebyshevT [2, x]
```

$$\text{Out}[ ] = \frac{1}{\sqrt{\pi}}$$

$$\text{Out}[ ] = \sqrt{\frac{2}{\pi}} x$$

$$\text{Out}[ ] = \sqrt{\frac{2}{\pi}} (-1 + 2 x^2)$$

By definition of the Chebyshev polynomials, a series expansion of an element  $f$  in  $H = L_w^2([-1, 1])$  is just the Fourier series expansion of the  $2\pi$ -periodic even function  $f(\cos(\phi))$ . The corresponding series converges in the norm of  $H$  and even uniformly for continuously differentiable  $f$  as we know from the theory of Fourier series. Thus, the coefficients of the expansions

$$(1) \quad f(x) = \frac{a_0}{2} T[0, x] + \sum_{k=1}^{\infty} a_k T[k, x]$$

are just the Fourier coefficients  $a_0/2$  and  $a_k$  ( $k = 1, 2, \dots$ ) of  $f(\cos(\phi))$ .

**Example 17.** We consider some approximations of the sign function on  $[-1, 1]$  by Chebyshev polynomials and observe the Gibbs phenomenon as known from Fourier series expansions near a jump. We plot the sign function and partial sums up to  $T[5, x]$ ,  $T[9, x]$  and  $T[19, x]$ . We only have odd powers in the polynomials, because sign is an odd function. We compute the coefficients with the inner product  $\langle \cdot, \cdot \rangle_w$  and as demonstration also as Fourier coefficients of  $\text{Sign}[\text{Cos}[\phi]]$ . We obtain the same coefficients.

```
In[ ]:= f[x_] = Sign[x]
a0half := Integrate[f[x] ChebyshevT [0, x] / Sqrt[1 - x^2], {x, -1, 1}] / Pi;
(* of course zero since f is odd *)
Fca0half := 1 / Pi Integrate[f[Cos[phi]], {phi, 0, 2 Pi}];
(*computed as Fourier coefficient *)
ak[k_] := 2 / Pi Integrate[f[x] ChebyshevT [k, x] / Sqrt[1 - x^2], {x, -1, 1}];
(*by inner product *)
Fcak[k_] := 2 / Pi Integrate[f[Cos[phi]] Cos[k phi], {phi, 0, Pi}];
(* as Fourier coefficient *)
```

$$\text{Out}[ ] = \text{Sign}[x]$$

A short test that the same coefficients are computed .

```

In[ * ]:= a0half
          ak[3]      (* Coeff. zu T[3,x] *)
          Fcak[3]     (* same computed as Fourier coefficient of f(cos(φ) *)

```

```
Out[ * ]:= 0
```

$$\text{Out}[ * ] = -\frac{4}{3\pi}$$

$$\text{Out}[ * ] = -\frac{4}{3\pi}$$

Now the indicated approximations with 5, 9 and 19 Chebyshev polynomials. The coefficients computed differently, in the last case by numerical integration.

```

In[ * ]:= fct2[t_] = Sum[Fcak[k] ChebyshevT[k, t], {k, 1, 5}]
          (* Näherung mit Polynomgrad 5 *)
          Simplify[N[%]] (* hier numerisch *)
          fct3[t_] = Sum[ak[k] ChebyshevT[k, t], {k, 1, 9}]
          (* Näherung mit Polynomgrad 9 *)
          Simplify[N[%]]
          fct4[t_] = Simplify[N[2/Pi] × Sum[
                        NIntegrate[f[Cos[x]] Cos[n x], {x, 0, Pi}] ChebyshevT[n, t], {n, 1, 19, 2}]]];

```

$$\text{Out}[ * ] = \frac{4t}{\pi} - \frac{4 \times (-3t + 4t^3)}{3\pi} + \frac{4 \times (5t - 20t^3 + 16t^5)}{5\pi}$$

$$\text{Out}[ * ] = 3.81972 t - 6.79061 t^3 + 4.07437 t^5$$

$$\begin{aligned} \text{Out}[ * ] = & \frac{4t}{\pi} - \frac{4 \times (-3t + 4t^3)}{3\pi} + \frac{4 \times (5t - 20t^3 + 16t^5)}{5\pi} - \\ & \frac{4 \times (-7t + 56t^3 - 112t^5 + 64t^7)}{7\pi} + \frac{4 \times (9t - 120t^3 + 432t^5 - 576t^7 + 256t^9)}{9\pi} \end{aligned}$$

$$\text{Out}[ * ] = 6.3662 t - 33.9531 t^3 + 85.5617 t^5 - 93.1284 t^7 + 36.2166 t^9$$

Below the plot of the approximations showing the Gibbs phenomenon

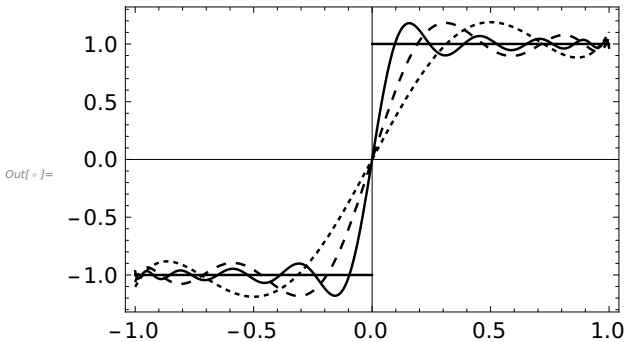


```

In[ ]:= g1 := Plot[f[t], {t, -1, 1}, Frame → True,
      FrameStyle → Directive[Black, FontSize → 15,
      FontWeight → Plain],
      PlotStyle → {GrayLevel[0.0], Thickness[0.005]}}
g2 := Plot[fct2[t], {t, -1, 1}, Frame → True,
      FrameStyle → Directive[Black, FontSize → 15, FontWeight → Plain],
      PlotStyle → {GrayLevel[0.0], Dashing[0.01], Thickness[0.005]}}
g3 := Plot[fct3[t], {t, -1, 1}, Frame → True,
      FrameStyle → Directive[Black, FontSize → 15, FontWeight → Plain],
      PlotStyle → {GrayLevel[0.0], Dashing[0.02], Thickness[0.005]}}
g4 := Plot[fct4[t], {t, -1, 1}, Frame → True,
      FrameStyle → Directive[Black, FontSize → 15, FontWeight → Plain],
      PlotStyle → {GrayLevel[0.0], Thickness[0.005]}}

In[ ]:= Show[g1, g2, g3, g4, PlotRange → All ]

```



### Comparison with interpolation by Legendre polynomials for $\sin(3x)$

We had already shown before interpolation by Legendre polynomials. We compare that with interpolation by Chebyshev polynomials and show the respective error curves. We see that the error by Chebyshev interpolation near to the boundary points -1,1 is much less than by Legendre interpolation, which is due to the weight function in the space  $H$ .

```

In[ ]:= coeff1[n_] := NIntegrate[LegendreP[n, x] Sin[3 x], {x, -1, 1}];
n2[x_] = Expand[Sum[coeff1[n] LegendreP[n, x]/(2/(2 n + 1)), {n, 0, 5}]]
(* Legendre interpolation polynomial *)

Out[ ]:= 0. + 2.97177 x - 4.23916 x^3 + 1.42043 x^5

In[ ]:= f[x_] := Sin[3 x]
n3[x_] = Simplify[
  N[2/Pi Sum[Integrate[f[Cos[phi]] Cos[n phi], {phi, 0, Pi}] ChebyshevT[n, x],
    {n, 1, 5}]] ]

Out[ ]:= 2.96278 x - 4.19364 x^3 + 1.37691 x^5

```

In[ ]:=

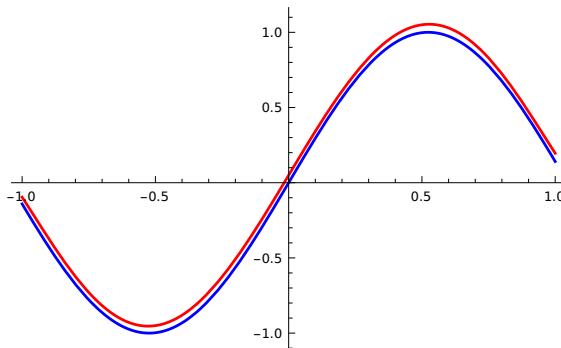
```
p1 = Plot[f[x], {x, -1, 1},
  PlotStyle → Directive [Blue, Thickness [0.005]], PlotRange → All,
  PlotStyle → Directive [Blue, Thickness [0.005]], PlotRange → All];
```

In[ ]:=

```
p3 = Plot[n3[x] + 0.05, {x, -1, 1},
  PlotStyle → Directive [Red, Thickness [0.005]], PlotRange → All, PlotLegends →
  Placed[{"Approximation by a Chebyshev Polynomial of Degree 5
  shown in Red with offset +0.005 for better visibility"}, Above]];
Show[p1, p3]
```

— Approximation by a Chebyshev Polynomial of Degree 5  
shown in Red with offset +0.005 for better visibility

Out[ ]:=



In[ ]:=

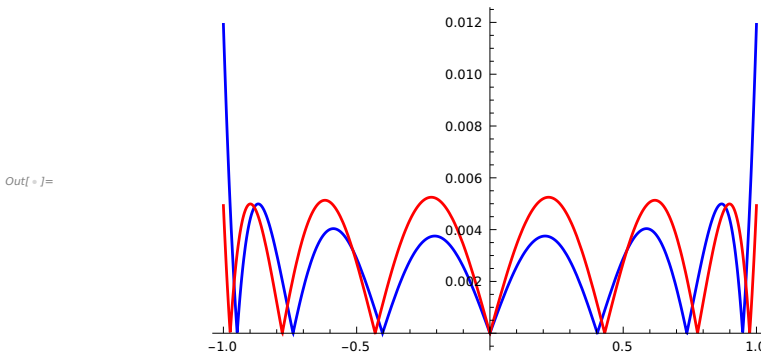
```
p4 := Plot[Abs[Sin[3 x] - n2[x]], {x, -1, 1},
  PlotStyle → Directive [Blue, Thickness [0.005]], PlotRange → All]
p5 := Plot[Abs[Sin[3 x] - n3[x]], {x, -1, 1}, PlotLegends →
  Placed[{"Approximation Errors by a Chebyshev Polynomial of Degree 5
  shown in Red versus Legendre Interpolation in Blue"}, Above],
  PlotStyle → Directive [Red, Thickness [0.005]], PlotRange → All]
```

### Below the errors of Legendre versus Chebyshev interpolation

The Chebyshev interpolation in red has greater errors in the interior of  $[-1, 1]$ , but is much better at the boundary. This is an effect of the used weight function in the inner product, which punishes errors at the boundary with greater norm.

In[ ]:= Show[p4, p5]

- Approximation Errors by a Chebyshev Polynomial of Degree 5 shown in Red versus Legendre Interpolation in Blue



### Connection with DCT I and DCT II, Example of C. Runge

In technical signal processing one often has the task to compute trigonometric or polynomial approximations for a function from samples of a continuous signal. Then, with the coefficients of an interpolation function the approximation is given by only a few numbers, which can be processed. (Computers can only process numbers and not continuous functions.)

Since Chebyshev polynomials are closely connected with Fourier series expansions, we have the possibility to find polynomial interpolations with the help of a DCT I or a DCT II. Then, the Fourier coefficients from above are replaced through their DCT coefficients computed from samples of a function. From our knowledge of the DFT it is obvious that alias effects must be observed. We treat this somewhat later below. In the following we consider continuous, piecewise continuously differentiable functions  $f$  on  $[-1,1]$ . By a parameter transformation, the results can also be used on other intervals. This is left to the reader. One obtains the following interpolation formulas (cf. [1], 6.6):

### Interpolation with a DCT I:

*With the  $m+1$  ( $m \in \mathbb{N}$ ) interpolation nodes  $x_n = \cos(n\pi/m)$ ,  $n=0, \dots, m$  in  $[-1,1]$  the polynomial*

$$P_{2,T}(x) = C_0 + 2 \sum_{k=1}^{m-1} C_k T[k,x] + C_m T[m,x]$$

*is a real-valued interpolation polynomial for  $f$ . The coefficients  $C_k$ ,  $k=0, \dots, m$ , are obtained by a DCT I of the samples  $f(x_n)$ ,  $n=0, \dots, m$ .*

### Interpolation with a DCT II and the so-called Chebyshev abscissa $x_n$ :

With the  $m+1$  ( $m \in \mathbb{N}$ ) Chebyshev interpolation nodes  $x_n = \cos((2n+1)\pi / (2m+2))$ ,  $n=0, \dots, m$  in  $[-1, 1]$ , i.e., with the zeros of  $T[m+1, x]$ ,

$$P_{3,T}(x) = A_0 + \sum_{k=1}^m A_k T[k, x]$$

is a real-valued interpolation polynomial for  $f$ . The coefficients  $A_k$ ,  $k=0, \dots, m$ , are obtained by a DCT II of the samples  $f(x_n)$ ,  $n=0, \dots, m$ .

We consider again the famous example of the Runge function  $f(x) = 1 / (1 + 25x^2)$  on the interval  $[-1, 1]$ .

Polynomial interpolations for this example with equidistant nodes yield with increasing number of nodes always worse approximations for  $f$ , while interpolation with Chebyshev nodes yields with increasing numbers of nodes a sequence of interpolation polynomials that converges uniformly to  $f$ .

At first the example with 7 and 17 equidistant nodes as illustration that these are bad choices for polynomial approximations to  $f$ .

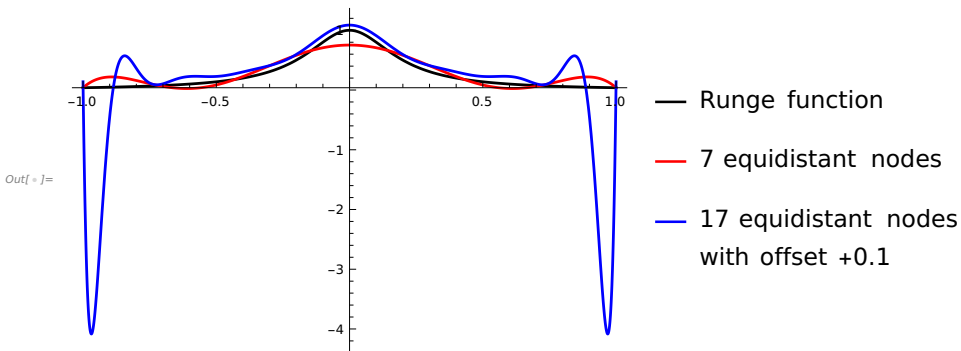
```
in[*]:= f[x_] := 1 / (1 + 25 x^2) (* The Runge example *)
a := -1; b := 1; (* Interval [a,b] *)
X[m_, n_] := a + (b - a) m / n; Y[m_, n_] := f[X[m, n]];
(* n+1 nodes X and samples Y of f *)

Lagr[n_, k_, x_] :=  $\left( \prod_{j=0}^{k-1} \frac{x - X[j, n]}{X[k, n] - X[j, n]} \right) \left( \prod_{j=k+1}^n \frac{x - X[j, n]}{X[k, n] - X[j, n]} \right);$ 

InterpolyLagrange [n_, x_] :=  $\sum_{k=0}^n Y[k, n] \times \text{Lagr}[n, k, x];$ 

(* The Lagrange Interpolation Polynomial *)
plot1 := Plot[f[x], {x, a, b}, PlotStyle → Directive[
  Black, Thickness [0.005]],
  PlotRange → All, PlotLegends → {"Runge function"}]
plot2 := Plot[InterpolyLagrange [7, x], {x, a, b}, PlotStyle → Directive [
  Red, Thickness [0.005]], PlotRange → All, PlotLegends → {"7 equidistant nodes"}]
plot3 := Plot[InterpolyLagrange [17, x] + 0.1, {x, a, b}, PlotStyle → Directive [
  Blue, Thickness [0.005]],
  PlotRange → All, PlotLegends → {"17 equidistant nodes
with offset +0.1"}]
```

```
In[ ]:= Show[plot1, plot2, plot3]
```



### Instead, Interpolation with Chebyshev Polynomials

First with a DCT I. We use the Mathematica version of the DCT I.

You must observe that for the Mathematica-DCT compared to my notation here and in [1] the scaling factor has to be adjusted. Here as prefactor  $1/\sqrt{2m}$ , if we use  $m+1$  samples of  $f$  in  $[-1,1]$ , or in other words  $m+1$  samples of  $f(\cos(\phi))$  for  $\phi \in [0,\pi]$ . I have preferred the scaling factor so that the DCT coefficients correspond to the amplitudes of the oscillations in the approximations.

(Hint: If you use implemented routines for a DFT (FFT), DCT in a program, test the norming factors by simply treating a cosine with the routines. This can easily prevent unpleasant surprises.)

We choose for the example 17 equidistant nodes in  $[0,\pi]$ , starting at zero, and the according samples of  $f(\cos(\phi))$ :

```
In[ ]:= m := 16; (* because numbering starts with zero *)
list1 := Table[f[Cos[ n π / m]], {n, 0, m}]
```

Now the DCT I of Mathematica with the right factor. Each second coefficient is zero according to the symmetry of  $f$ . We see the graphics of  $f$  (black) and its approximation (red)

```
In[ ]:= coeff = 1 / Sqrt[2 m] FourierDCT [list1, 1];
```

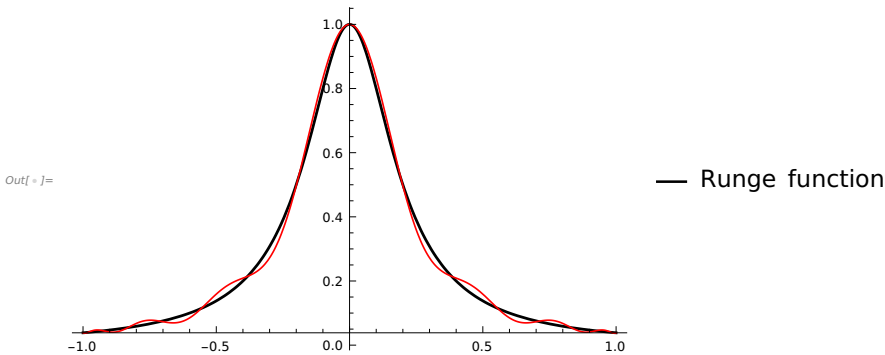
```
InterpolyTscheb1 [x_] = Simplify [coeff [[1]] +
```

$$2 \sum_{k=2}^m \text{coeff} [[k]] \text{ChebyshevT} [k - 1, x] + \text{coeff} [[m + 1]] \text{ChebyshevT} [m, x]$$

```
plot4 := Plot[InterpolyTscheb1 [x], {x, a, b}, PlotStyle → Directive [
  Red, Thickness [0.003]], PlotRange → All]
```

```
Show[plot1, plot4]
```

```
Out[ ]:= 1. - 18.4579 x2 + 180.138 x4 - 931.478 x6 + 2718.63 x8 -
4638.33 x10 + 4585.72 x12 - 2433.11 x14 + 535.928 x16
```



### The same game with a DCT II. We use the Mathematica version of the DCT II.

Again we have to adjust the norming factor in the DCT II of Mathematica, here the factor  $1/\sqrt{m+1}$ . We compute the interpolation polynomial, again with  $m+1=17$  nodes and plot it together with the function von C. Runge:

```
In[ ]:= list2 := Table[N[f[Cos[(2 n + 1) π / (2 m + 2)]]], {n, 0, m}]
```

```
(* Samples at the Chebyshev abscissa *)
```

```
coeff2 = N[1 / Sqrt[m + 1]] × Chop[FourierDCT [list2, 2] ]; (* DCT II of that list *)
```

```
In[ ]:= InterpolyTscheb2 [x_] = Simplify [coeff2 [[1]] + 2 ∑_{k=2}^{m+1} coeff2 [[k]] ChebyshevT [k - 1, x]]
```

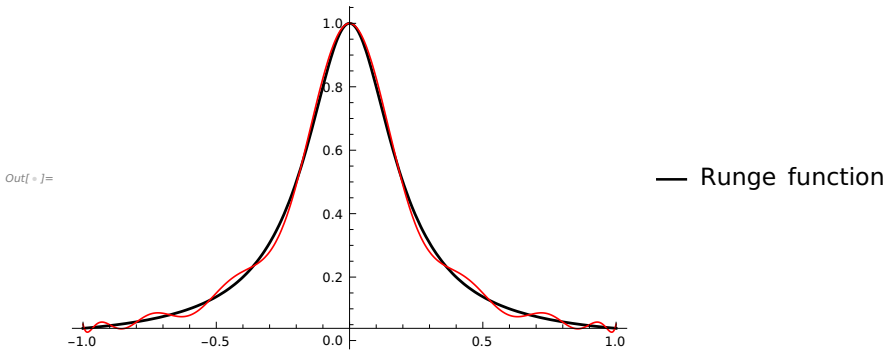
```
(* resulting polynomial *)
```

```
Out[ ]:= 1. - 19.192 x2 + 201.018 x4 - 1122.49 x6 + 3529.36 x8 -
6457.85 x10 + 6814.73 x12 - 3842.14 x14 + 895.603 x16
```

```

In[ ]:= plot5 := Plot[InterpolyTscheb2 [x], {x, a, b}, PlotStyle → Directive [
      Red, Thickness [0.003]], PlotRange → All]
Show[plot1, plot5]

```



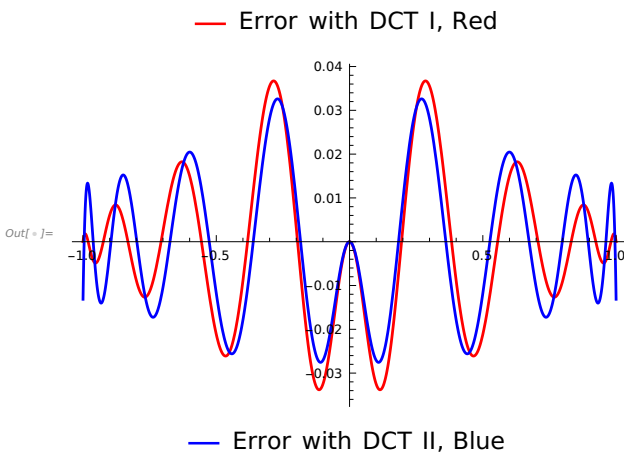
As expected one can hardly distinguish both versions by looking at them. Thus, we show the corresponding error functions, the error by the DCT I in red, by the DCT II in blue. With increasing numbers  $m$  of nodes both interpolations yield also good approximations considered on the entire interval  $[-1, 1]$ , which finally converge uniformly to  $f$  for  $m \rightarrow \infty$ .

### The errors of the both interpolations

```

In[ ]:= plot6 := Plot[f[x] - InterpolyTscheb1 [x], {x, a, b}, PlotStyle → Directive [
      Red, Thickness [0.005]], PlotRange → All,
      PlotLegends → Placed[{"Error with DCT I, Red"}, Above]]
plot7 := Plot[f[x] - InterpolyTscheb2 [x], {x, a, b}, PlotStyle → Directive [
      Blue, Thickness [0.005]], PlotRange → All,
      PlotLegends → Placed[{"Error with DCT II, Blue"}, Below]]
Show[plot6, plot7]

```



### The Alias Effect with Chebyshev Polynomials

Since the considered interpolations by Chebyshev polynomials are closely associated with the DCT, it is obvious that alias effects are involved in the computation of the coefficients as in the DFT and DCT. As in a DFT with  $N$  samples over  $[0, T]$  all values of the functions  $\exp(i(k+mN)2\pi t/T)$  coincide at the sampling points you cannot distinguish values of certain Chebyshev polynomials at the interpolation nodes. From that point of view one can see the approximation error of the interpolation polynomials as a consequence of alias effects in the polynomial coefficients and obtain error estimates from that (see [8] for details).

For an illustration we consider the last case of an interpolation with the Chebyshev abscissa  $x_n = \cos((\pi(2n+1)/(2m+2)))$ ,  $n = 0, \dots, m$ .

From  $T[k, \cos(x_n)] = \cos[k x_n]$  one gets with some computation work by the trigonometric addition theorems that

$T[k, x]$  and  $(-1)^l T[l(2m+2) \pm k, x]$  for  $l \in \mathbb{N}$  coincide on all nodes  $x_n$ . For a continuous function  $f$  on  $[-1, 1]$  and its interpolation polynomial with  $m+1$  Chebyshev abscissa as nodes we obtain with the interpolation polynomial  $P_{3,T}(x) = A_0 + \sum_{k=1}^m A_k T[k, x]$ :

For the coefficients  $A_k$  in  $P_{3,T}$  above we have the following alias relation:

*For  $l \in \mathbb{N}$ ,  $k \in \mathbb{N}_0$ , the coefficients  $a_k$  of  $f(x) = \frac{a_0}{2} T[0, x] + \sum_{k=1}^{\infty} a_k T[k, x]$  and the  $m+1$  Chebyshev abscissa as nodes it holds for the coefficients  $A_k$  the alias relation*

$$A_k = C_k(a_k + \sum_{l=1}^{\infty} (-1)^l (a_{l(2m+2)+k} + a_{l(2m+2)-k}))$$

**We demonstrate this effect with a simple example.**

**Example 18.** We interpolate  $f(x) = T[9, x] + 2 T[10, x] + 2 T[11, x] + T[20, x] + T[21, x]$  with 5 Chebyshev abscissa as nodes in  $[-1, 1]$ . The coefficients

$a_{10} = a_{11} = 2$  und  $a_{20} = a_{21} = 1$  yield by the alias effect with  $m=4$  the interpolation polynomial  $P_{3,T}[x] = -T[0, x] - 2T[1, x] = -1 - 2x$ , because

$A_0 = \frac{1}{2} \times (-2 a_{10} + 2 a_{20}) = -1$  und  $A_1 = -a_{11} - a_9 + a_{21} = -2$ . We show  $P_{3,T}$  and plot the polynomial (red) and  $f$  (blue).

Of course it is obvious in advance that the number of nodes and thus the degree of the according interpolation polynomial is by far not sufficient to approximate  $f$  with that polynomial. It is simply demonstrated what finds its way into the coefficients  $A_k$ .



```

f[x_] := ChebyshevT [9, x] + 2 ChebyshevT [10, x] +
        2 ChebyshevT [11, x] + ChebyshevT [20, x] + ChebyshevT [21, x]
m := 4
list4 := Table[N[f[Cos[(2 n + 1)  $\pi$  / (2 m + 2)]]], {n, 0, m}]
(* samples at the Chebyshev abscissa *)
coeff3 := N[1 / Sqrt[m + 1]]  $\times$  Chop[FourierDCT [list4, 2]]
(* DCT II of the samples *)

InterpolyTscheb3 [x_] = Simplify  $\left[ \text{coeff3}[[1]] + 2 \sum_{k=2}^{m+1} \text{coeff3}[[k]] \text{ChebyshevT}[k-1, x] \right]$ 

(* resulting polynomial *)
plot6 := Plot[f[x], {x, a, b}, PlotStyle  $\rightarrow$  Directive [
        Blue, Thickness [0.006]], PlotRange  $\rightarrow$  All,
        PlotLegends  $\rightarrow$  Placed[{"The function f(x), Blue"}, Above]]
plot7 := Plot[InterpolyTscheb3 [x], {x, a, b}, PlotStyle  $\rightarrow$  Directive [
        Red, Thickness [0.005]], PlotRange  $\rightarrow$  All,
        PlotLegends  $\rightarrow$  Placed[{"The interpolation with 5 Chebyshev nodes
        and Alias Effects, Red"}, Below]]

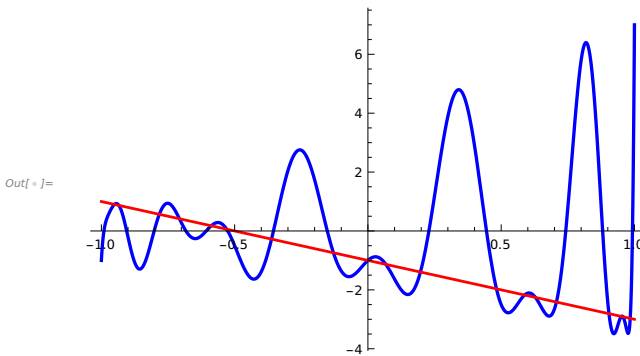
```

Out[ ]:=  $-1. - 2. x - 2.52051 \times 10^{-9} x^3$

The coefficient for  $x^3$  comes from errors in the numerical computation of the DCT.

In[ ]:= Show[plot6, plot7]

— The function f(x), Blue



— The interpolation with 5 Chebyshev nodes  
and Alias Effects, Red

One must observe such effects, when one works with samples and interpolation polynomials for example in non-linear equations and wants maybe approximate terms like a function  $f^3$  by a polynomial. Thus, you need a sufficient number of samples for a good approximation of a function  $f$ . If interested, you can find error estimates in [2].

**Exercise.** Derive a corresponding alias relation for the interpolation with the nodes  $x_n = \cos(n\pi/m)$ ,  $n = 0, \dots, m$  in  $[-1, 1]$  and test your formula.

### An Extremal Property of the Chebyshev Polynomials

Finally in that section we will illustrate the following extremal property of Chebyshev polynomials, which plays a role in circuit design in electrical engineering, where Chebyshev filters are widely used due to their damping properties outside of the passband of lowpass filters. It is proven in [1], 6.7.

*It holds the following theorem:*

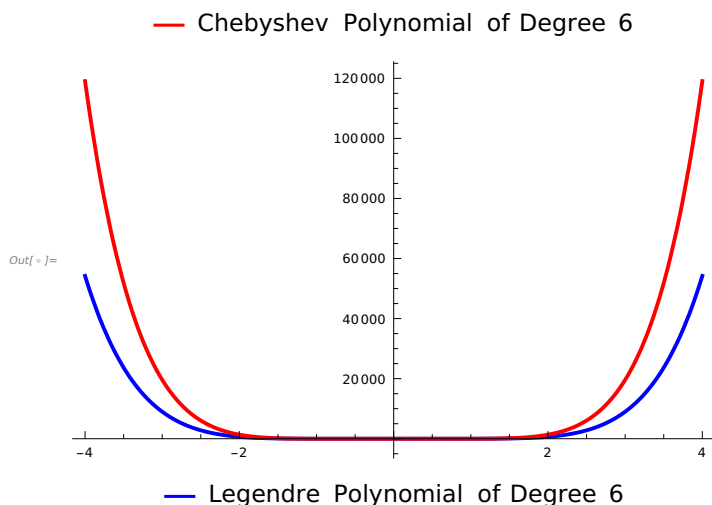
1. For every  $x_0$  outside of  $[-1, 1]$  the polynomial  $T[n, x]/T[n, x_0]$  has minimal supremum norm compared with all polynomials  $P$  with degree  $n$  and  $P(x_0) = 1$ .
2. Compared with all polynomials  $P$  of degree  $n$  with  $|P(x)| \leq 1$  on  $[-1, 1]$  the Chebyshev polynomial  $T[n, x]$  increases fastest outside of  $[-1, 1]$ , i.e., there holds  $|T[n, x]| \geq |P(x)|$ .

**Example 19.** We consider as last example in that section the graphs of a Legendre polynomial and a Chebyshev polynomial of the first kind with equal degrees in  $[-4,4]$  and see the different growth in the complement of  $[-1,1]$ .

```

In[ ]:= plot8 := Plot[LegendreP[6, x], {x, -4, 4}, PlotStyle -> Directive[
    Blue, Thickness[0.006]], PlotRange -> All,
    PlotLegends -> Placed[{"Legendre Polynomial of Degree 6"}, Below]]
plot9 := Plot[ChebyshevT[6, x], {x, -4, 4}, PlotStyle -> Directive[
    Red, Thickness[0.006]], PlotRange -> All,
    PlotLegends -> Placed[{"Chebyshev Polynomial of Degree 6"}, Above]]
Show[plot8, plot9]

```



## 4 The DCT 2D, JPEG, Huffman Code

### 4.1 DCT-2D, JPEG

We see how the DCT II is defined for 2D signals and apply it to an example from image processing.

For the definition I have simply copied the following section from [1], 6.9

As announced in the example before, we first specify the 2-dimensional variant of DCT II, which we use below. For an  $(M \times N)$ -matrix  $A$  with components  $A_{mn}$ , the DCT-2D of  $A$  is the  $(M \times N)$ -matrix  $B$  with components  $B_{pq}$  for  $0 \leq p \leq M - 1$ ,  $0 \leq q \leq N - 1$ , defined by

$$\text{DCT-2D:} \quad B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos\left(\frac{\pi(2m+1)p}{2M}\right) \cos\left(\frac{\pi(2n+1)q}{2N}\right),$$

$$\text{IDCT-2D:} \quad A_{mn} = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \alpha_p \alpha_q B_{pq} \cos\left(\frac{\pi(2m+1)p}{2M}\right) \cos\left(\frac{\pi(2n+1)q}{2N}\right)$$

Out[ \* ] =

Thereby, the normalization factors are those of the numerics software Matlab given by

$$\alpha_p = \begin{cases} 1/\sqrt{M} & \text{if } p = 0, \\ \sqrt{2/M} & \text{if } 1 \leq p \leq M - 1, \end{cases} \quad \alpha_q = \begin{cases} 1/\sqrt{N} & \text{if } q = 0, \\ \sqrt{2/N} & \text{if } 1 \leq q \leq N - 1. \end{cases}$$

The DCT-2D is the concatenation of a DCT over the rows of the matrix  $A$ , followed by a DCT over the columns of the preceding transformation result. The larger  $p + q$  the higher frequency components in the signal the coefficient  $B_{pq}$  is assigned to

The DCT-2D is widely used in image processing, for example in image compression. We consider the case of the well-know JPEG compression as example and explore some of its properties.

In JPEG compression an image is divided into 8x8 or 16x16 pixel blocks, which are transformed with a DCT-2D. The results are quantized per block. This is done so that the according DCT coefficients of such blocks of a grayscale matrix, depending on their position in the coefficient matrix, are divided by accordingly positioned values of a so-called **luminance table** and rounded to integers. The values of the luminance table **depend on the desired compression ratio**. In the following example such a luminance table is shown and used.

## JPEG

### An 8x8 Luminance Table for JPEG

```
In[ ]:= qLum = {{16, 11, 10, 16, 24, 40, 51, 61},
  {12, 12, 14, 19, 26, 58, 60, 55}, {14, 13, 16, 24, 40, 57, 69, 56},
  {14, 17, 22, 29, 51, 87, 80, 62}, {18, 22, 37, 56, 68, 109, 103, 77},
  {24, 35, 55, 64, 81, 104, 113, 92}, {49, 64, 78, 87, 103, 121, 120, 101},
  {72, 92, 95, 98, 112, 100, 103, 99}}; qLum // MatrixForm
```

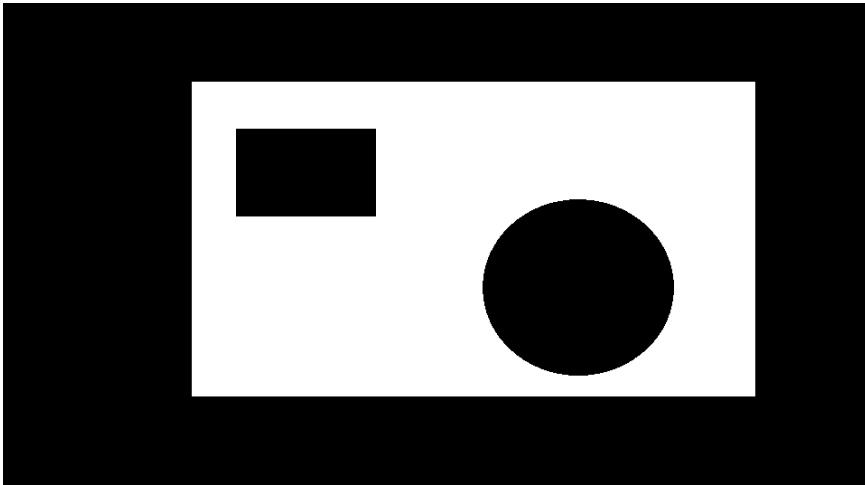
```
Out[ ]:= %//MatrixForm=
```

$$\begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}$$

Since the DCT coefficients for higher frequency components usually decrease rapidly and the divisors of the table for such coefficients increase, one mostly gets many zeros in the high frequencies as a result of quantization. These quantized spectral data can be stored or transmitted in compressed form by entropy encoding. When transmitting a JPEG image, the used encoding method (e.g. Huffman table, not uniquely determined) is specified in the file header as necessary information for decoding. At the viewer, the data stream is decoded back into the DCT matrix and subjected to IDCT-2D block by block. As a rule, the IDCT data for the image must also be rendered again if there are values, which do not belong to integers in [0,255]. In color images, the color information is quantized analogously with chrominance tables. The quantization can lead to undesired artifacts in the neighborhood of edges in combination with the Gibbs phenomenon, since the IDCT after compression usually yields a trigonometric interpolation polynomial different from that of the original DCT data. This can be quickly verified by zooming in on the edges in a JPEG image as we will see in the following example.

**Example 20.** We load a test image and use the Mathematica JPEG algorithm to analyze some effects.

```
In[ ]:= testimage = Import["/home/rolf/Desktop/MATHEMATICA -NEW/Testimage.bmp"]
```



Out[ ]:=

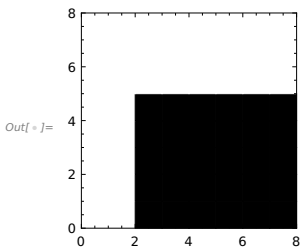
```
In[ ]:= ImageDimensions[testimage] (* Check Pixel numbers *)
imagedata = ImageData[ImageTake[testimage, {137, 144}, {257, 264}], "Byte"];
imagedata // MatrixForm
(* 8x8 Pixel Block at the edge of the rectangle upper left in byte form *)
```

Out[ ]:= {960, 534}

Out[ ] // MatrixForm =

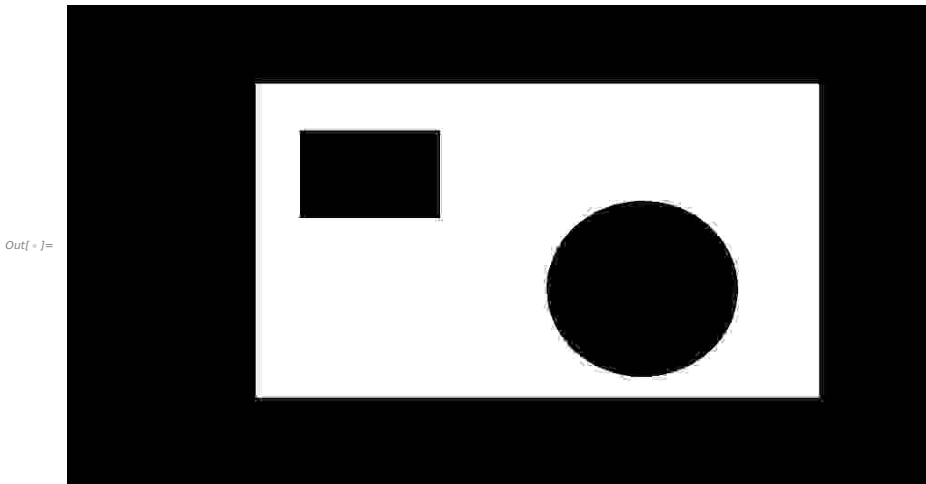
$$\begin{pmatrix} 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 0 & 0 & 0 & 0 & 0 & 0 \\ 255 & 255 & 0 & 0 & 0 & 0 & 0 & 0 \\ 255 & 255 & 0 & 0 & 0 & 0 & 0 & 0 \\ 255 & 255 & 0 & 0 & 0 & 0 & 0 & 0 \\ 255 & 255 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

```
In[ ]:= image2 = Image[imagedata , Magnification -> 20, ImageSize -> 150];
Show[image2, Frame -> True, FrameStyle -> Black]
(* Edge upper left of the rectangle . Shown are 8x8 Pixel *)
```



Below the jpeg compressed image . On closer inspection one recognizes **artefacts** at the boundaries of the figures. They are caused by aliasing and the Gibbs phenomenon in the DCT approximation. Here we use the jpeg compression implemented in Mathematica. Below we do this ourselves to see better what's going on.

```
In[ ]:= Export["testimage.jpeg", testimage ,
  ColorSpace -> "Grayscale ", "CompressionLevel " -> 1.0];
testimagejpeg = Import["testimage.jpeg"]
(* Size of originally 512 KB compressed to 9 KB *)
```



We look again at the left upper edge of the rectangle

```

In[ * ]:= image3data = ImageData [ImageTake [testimagejpeg , {137, 144}, {257, 264}], "Byte"];
image3data // MatrixForm (*The same section as above,
now from the new image data matrix. The changes are obvious.*)
image3 = Image[image3data , "Byte", ImageSize -> 150];
Show[image3, Frame -> True, FrameStyle -> Black]

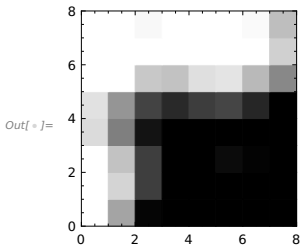
```

Out[ \* ]:=

```

( 255 255 248 255 255 255 250 190 )
( 255 255 255 255 255 255 255 208 )
( 255 255 200 194 223 228 185 136 )
( 225 149 67 41 61 69 38 0 )
( 219 127 19 0 0 0 0 0 )
( 255 194 62 0 0 11 3 0 )
( 255 212 62 0 0 0 0 0 )
( 255 164 5 0 0 0 0 0 )

```



Analogously music is distorted, if it is saved compressed as MP3 files. The edges of an image correspond in acoustics major changes in the dynamic of a music piece. Deleting of small DCT coefficients - mostly corresponding to high frequencies - diminishes the bandwidth and cancels overtones. Let's do the compression ourselves for the considered 8x8 block.



```

In[ * ]:= dctblock = FourierDCT [imagedata ]; dctblock // MatrixForm (* DCT of the block*)
jpegblock =
  Table[Sign[dctblock[[i, j]] Floor[Abs[dctblock[[i, j]] / qLum[[i, j]]], {i, 1, 8}, {j, 1, 8}];
jpegblock // MatrixForm (* The resulting data for the JPEG format –
  now with many zeros. Rounded to zero direction *)

```

Out[ \* ]//MatrixForm=

$$\begin{pmatrix} 1083.75 & 288.828 & 208.233 & 101.423 & 0. & -67.7686 & -86.2531 & -57.4515 \\ 452.847 & -136.779 & -98.6121 & -48.0304 & 0. & 32.0929 & 40.8465 & 27.207 \\ 176.692 & -53.3685 & -38.4765 & -18.7405 & 0. & 12.522 & 15.9375 & 10.6157 \\ -65.8676 & 19.8948 & 14.3434 & 6.98613 & 0. & -4.66799 & -5.94122 & -3.95733 \\ -135.234 & 40.8465 & 29.4487 & 14.3434 & 0. & -9.58393 & -12.198 & -8.12487 \\ -44.0114 & 13.2933 & 9.58393 & 4.66799 & 0. & -3.11905 & -3.9698 & -2.6442 \\ 73.1882 & -22.1059 & -15.9375 & -7.76258 & 0. & 5.18679 & 6.60153 & 4.39715 \\ 90.0768 & -27.207 & -19.6152 & -9.55383 & 0. & 6.38367 & 8.12487 & 5.41181 \end{pmatrix}$$

Out[ \* ]//MatrixForm=

$$\begin{pmatrix} 67 & 26 & 20 & 6 & 0 & -1 & -1 & 0 \\ 37 & -11 & -7 & -2 & 0 & 0 & 0 & 0 \\ 12 & -4 & -2 & 0 & 0 & 0 & 0 & 0 \\ -4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The **data** in a segment of a JPEG file **are thus DCT coefficients** . Thus, the viewer always needs a program (browser, image viewer) that generates by a IDCT again an image from the data - hopefully in realtime. The decoder program therefore must know for reconstruction the used luminance table, which is transmitted in a JPEG file (there in segment FF DB).

The DCT data are usually stored and transmitted with a **Huffman code**. Only with that coding of the data, which now as rule contain many zeros, a data compression is achieved. A Huffman code is not uniquely determined as we will see below. In the header of each JPEG file - in segment FF C4 (DHT "Definition of Huffman Table") - is defined, which Huffman code was used for the data. Each viewer program must then generate per image block the corresponding code table to correctly decode it.

## Image reconstruction

For image reconstruction it is requantized with the same luminance table, i.e., the elements of the last matrix above are multiplied per element with the coefficients of the luminance matrix. We proceed with the data block above.

```
In[ * ]:= blockrequant = Table[jpegblock qLum];
blockrequant // MatrixForm (* compare with the matrix dctblock above *)
imagesection = FourierDCT[blockrequant, 3];
imagesection // MatrixForm (* Result after backwards transform *)
```

Out[ \* ]//MatrixForm=

$$\begin{pmatrix} 1072 & 286 & 200 & 96 & 0 & -40 & -51 & 0 \\ 444 & -132 & -98 & -38 & 0 & 0 & 0 & 0 \\ 168 & -52 & -32 & 0 & 0 & 0 & 0 & 0 \\ -56 & 17 & 0 & 0 & 0 & 0 & 0 & 0 \\ -126 & 22 & 0 & 0 & 0 & 0 & 0 & 0 \\ -24 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 49 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 72 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Out[ \* ]//MatrixForm=

$$\begin{pmatrix} 232.424 & 266.247 & 237.206 & 257.599 & 281.129 & 254.306 & 252.728 & 239.335 \\ 248.509 & 270.827 & 227.428 & 240.51 & 267.331 & 250.489 & 259.007 & 251.401 \\ 296.274 & 294.574 & 218.665 & 208.702 & 229.084 & 217.085 & 232.532 & 228.796 \\ 216.537 & 185.044 & 67.8716 & 26.0159 & 31.7552 & 16.3546 & 30.8313 & 25.7937 \\ 261.305 & 206.599 & 59.2583 & -2.47993 & -2.23629 & -16.8252 & -3.25102 & -10.8073 \\ 274.018 & 207.74 & 48.739 & -14.1592 & -5.43848 & -11.4411 & 4.25628 & -5.12619 \\ 279.142 & 208.043 & 46.2578 & -12.0838 & 5.72693 & 5.65908 & 20.5211 & 7.69597 \\ 270.516 & 196.841 & 33.1335 & -24.2175 & -4.27453 & -4.80511 & 6.07901 & -10.7755 \end{pmatrix}$$

We see the mentioned alias and Gibbs effect with real values outside of the greyscale with integers from zero to 255.

Thus, **the image must be rastered again**. We do this ourselves by replacing negative values by zero, those greater than 255 by 255, and otherwise we round off. Afterwards we look at the result.

```

In[ ]:= imageraster1 = Table[Max[{imagesection[[i, j]], 0}], {i, 1, 8}, {j, 1, 8}];
imageraster = Table[Floor[Min[{imageraster1[[i, j]], 255}]], {i, 1, 8}, {j, 1, 8}];
imageraster // MatrixForm

```

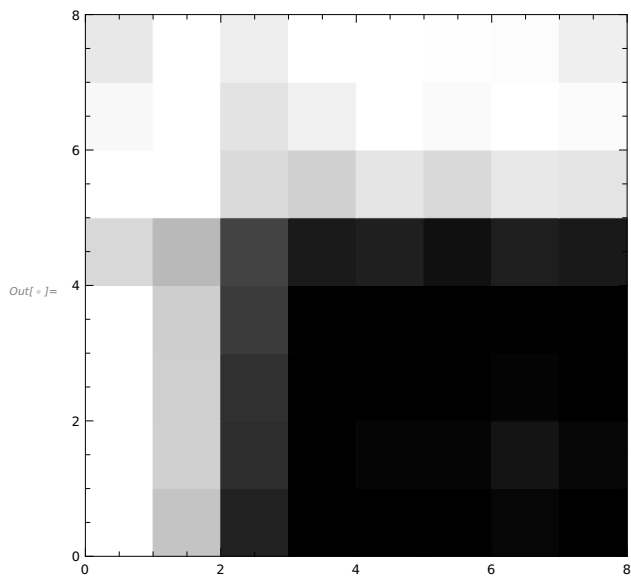
Out[ ]:= MatrixForm=

$$\begin{pmatrix} 232 & 255 & 237 & 255 & 255 & 254 & 252 & 239 \\ 248 & 255 & 227 & 240 & 255 & 250 & 255 & 251 \\ 255 & 255 & 218 & 208 & 229 & 217 & 232 & 228 \\ 216 & 185 & 67 & 26 & 31 & 16 & 30 & 25 \\ 255 & 206 & 59 & 0 & 0 & 0 & 0 & 0 \\ 255 & 207 & 48 & 0 & 0 & 0 & 4 & 0 \\ 255 & 208 & 46 & 0 & 5 & 5 & 20 & 7 \\ 255 & 196 & 33 & 0 & 0 & 0 & 6 & 0 \end{pmatrix}$$

```

In[ ]:= jpegimageresult = Image[imageraster, "Byte"];
Show[jpegimageresult, Frame -> True, FrameStyle -> Black]

```



## 4.2 Huffman Coding

Even if it is not actual Fourier analysis, but closely connected with its everyday use in image processing, we finally explain for our example how a possible Huffman code for the elements in our 8x8-Matrix jpegblock above is generated. For simplicity, we do not use rearrangement or run lengths. Only the principle of Huffman coding is explained.

At first we flatten our 8x8-Matrix jpegblock above. The resulting list has 64 elements.

```
In[ * ]:= data = Flatten[jpegblock]
```

```
Out[ * ]:= {67, 26, 20, 6, 0, -1, -1, 0, 37, -11, -7, -2, 0, 0, 0, 0, 12,
           -4, -2, 0, 0, 0, 0, 0, -4, 1, 0, 0, 0, 0, 0, 0, -7, 1, 0, 0, 0, 0, 0, 0,
           -1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0}
```

**Huffman coding is done in 3 steps, the decoding in one step.**

1) At first a character frequency table of the coefficients in the list is generated. The result contains the coefficients with increasing frequency, in the example for instance {2,-7} as list element, because -7 appears two times in the coefficient matrix. Here the used function and the result for that:

```
In[ * ]:= getCharFreqTable [data_List] := Sort[{Count[Flatten[#, & /@ data], #][1]], #][1]] & /@
           Transpose [{Union[Flatten[#, & /@ data]]}];
```

```
getCharFreqTable [
  data]
```

```
Out[ * ]:= {{1, -11}, {1, 6}, {1, 12}, {1, 20}, {1, 26}, {1, 37},
           {1, 67}, {2, -7}, {2, -4}, {2, -2}, {3, -1}, {4, 1}, {44, 0}}
```

2) From that list a Huffman tree is generated in which nodes and edges are represented by a binary string. There are always exactly 2 paths that we can choose from a node to a leaf or another node. These are represented by a 1 or a 0. The optimal tree structure is achieved by recursion (reduction), which processes the coefficients based on their respective frequency.

```
In[ * ]:= getHuffmanTree [data_List] :=
           Nest[Sort[Delete[ReplacePart[#, {Plus @@ (Transpose[Take[#, {1, 2}]]][1]],
           Take[#, {1, 2}]], {1}], {2}]] &,
           getCharFreqTable [data], Length[getCharFreqTable [data]] - 1][1, 2]
```

This function works as follows:

The process starts by placing the first two coefficients of the ordered frequency table above into a character set under a node. The frequency of this character set is then defined as the sum of the frequencies of the characters it contains. In the example with {1,-11} and {1,6}, 2 is the sum of the two frequencies in these elements.

**Example:**

```
In[ * ]:= {Plus @@ (Transpose [Take[#, {1, 2}]] [1]), Take[#, {1, 2}]} &@getCharFreqTable [data]
Out[ * ]:= {2, {{1, -11}, {1, 6}}}
```

The result then replaces the first two entries in the frequency table. In the example:

```
In[ * ]:= (Delete[ReplacePart [#, {Plus @@ (Transpose [Take[#, {1, 2}]] [1]), Take[#, {1, 2}]],
{1}], {2}] &@getCharFreqTable [data])
Out[ * ]:= {{2, {{1, -11}, {1, 6}}}, {1, 12}, {1, 20}, {1, 26}, {1, 37},
{1, 67}, {2, -7}, {2, -4}, {2, -2}, {3, -1}, {4, 1}, {44, 0}}
```

Now it is again sorted with increasing frequencies :

```
In[ * ]:= (Sort[Delete[ReplacePart [#, {Plus @@ (Transpose [Take[#, {1, 2}]] [1]),
Take[#, {1, 2}]], {1}], {2}] &@getCharFreqTable [data])
Out[ * ]:= {{1, 12}, {1, 20}, {1, 26}, {1, 37}, {1, 67}, {2, -7},
{2, -4}, {2, -2}, {2, {{1, -11}, {1, 6}}}, {3, -1}, {4, 1}, {44, 0}}
```

This process is continued by placing the first two resulting entries, whether a single coefficient or a group of coefficients, under a node.

The recursion ends with the full Huffman tree, here in the form of a “nested list”, which we call “hTree”.

```
In[ * ]:= hTree = getHuffmanTree [data]
Out[ * ]:= {{20, {{8, {{4, {{2, -4}, {2, -2}}}, {4, {{2, {{1, -11}, {1, 6}}}, {2, {{1, 12}, {1, 20}}}}}},
{12, {{5, {{2, {{1, 26}, {1, 37}}}, {3, -1}}},
{7, {{3, {{1, 67}, {2, -7}}}, {4, 1}}}}}}, {44, 0}}
```

**3)** Below is how to generate a Huffman code from this:

The code of a character in our data is uniquely determined by its position in hTree.

From the positions of the characters we generate code words and thus a “code list” analogous to “data”. We then output the bit stream that encodes the entire list. You

can see that data elements receive shorter code words the greater their frequency in the list to be list to be encoded. (Task: Analyze the two following functions). Below the resulting bitstream that encodes the example.

```

In[ * ]:= encodeChar [c_, tree_List] := (list = Flatten [Position [tree, {_, c}]];
      b = Table [ToString [list[[i]] - 1], {i, 1, Length [list], 2}];
      StringJoin [b])
encode [charlist_, tree_] :=
  Table [encodeChar [charlist[[n]], tree], {n, Length [charlist]}]
codelist = encode [data, hTree]
bitstream = {StringJoin [codelist]}
StringLength [bitstream[[1]]]

Out[ * ]:= {01100, 01000, 00111, 00101, 1, 0101, 0101, 1, 01001, 00100, 01101, 0001, 1, 1, 1, 1,
      00110, 0000, 0001, 1, 1, 1, 1, 1, 0000, 0111, 1, 1, 1, 1, 1, 01101, 0111, 1, 1, 1,
      1, 1, 1, 0101, 1, 1, 1, 1, 1, 1, 0111, 1, 1, 1, 1, 1, 0111, 1, 1, 1, 1, 1, 1}

Out[ * ]:= {0110001000001110010110101010101001001000110100011111001100000000111111,
      000001111111110110101111111101011111110111111110111111110111111111 }

Out[ * ]:= 133

```

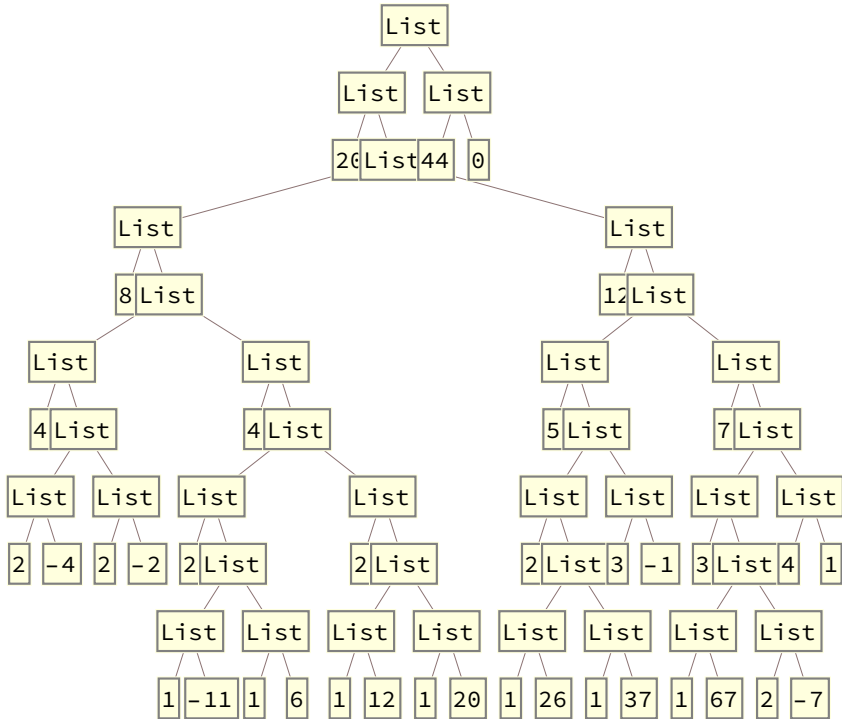
A quick recalculation shows : Encoding the 64 numbers as 8 - bit words would require 512 bits . Coding as above would require a bit stream with a length of only 133 bits, i. e., **would only use about 1/4 of the previous memory space**. A more clever arrangement according to a zigzag pattern as mentioned above and representation of the resulting terminating sequence with 35 zeros by a single code word would save another 26 % .

The CCITT coding uses de facto Difference Coding, one zigzag pattern per block and run-length code .

Mathematica offers an implemented command `TreeForm` for a plot of the corresponding tree.

In[ ]:= TreeForm [hTree]

Out[ ]:= TreeForm=



### Now the Decoding

We first generate a list that shows the assignment of the matrix coefficients and their code words in pairs. We call the list "translationTable".

Then - programmed as a loop - we process the bitstream by combining characters of the 0-1 sequence with the StringTake command until we find the first code word in translationTable. It is in the second component of a hit. Then we assign the first component of this hit - the encoded character - to the result component "result[[1]]" and delete the decoded bit sequence from 'stream' using StringDrop. The procedure is then repeated in a loop until all entries "result[[n]]" are available for all n,  $1 \leq n \leq 64$ .

```

In[ * ]:= huffmanTable [data_List, code_List] := Table[{data[[i]], code[[i]], {i, Length[data]}}]
translationTable = Union[huffmanTable [data, codelist]]
result = Table[NULL, {n, 64}];
stream = bitstream;

Out[ * ]:= {{-11, 00100}, {-7, 01101}, {-4, 0000}, {-2, 0001}, {-1, 0101}, {0, 1}, {1, 0111},
           {6, 00101}, {12, 00110}, {20, 00111}, {26, 01000}, {37, 01001}, {67, 01100}}

In[ * ]:= For[n = 1, n ≤ 64, n++,
           For[m = 1, m ≤ 64 && result[[n]] == NULL, m++,
             ( a := StringTake [stream, m];
               If[MemberQ [translationTable, {_, a[[1]]}],
                 (b := Position [translationTable, {_, a[[1]]}];
                   result[[n]] = translationTable [[b[[1]]][[1]][[1]];
                   stream = StringDrop [stream, m];)] )]]
ArrayReshape [result, {8, 8}] // MatrixForm

```

Out[ \* ]//MatrixForm=

$$\begin{pmatrix} 67 & 26 & 20 & 6 & 0 & -1 & -1 & 0 \\ 37 & -11 & -7 & -2 & 0 & 0 & 0 & 0 \\ 12 & -4 & -2 & 0 & 0 & 0 & 0 & 0 \\ -4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

In[ \* ]:=

**Above, the encoded and again re-decoded matrix, and below for comparison the matrix jpegblock from p. 122 to check that everything worked well.**

In[ \* ]:= jpegblock // MatrixForm

Out[ \* ]//MatrixForm=

$$\begin{pmatrix} 67 & 26 & 20 & 6 & 0 & -1 & -1 & 0 \\ 37 & -11 & -7 & -2 & 0 & 0 & 0 & 0 \\ 12 & -4 & -2 & 0 & 0 & 0 & 0 & 0 \\ -4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$



## References

- [1] Brigola, R. Fourier Analysis and Distributions, A First Course with Applications, Springer, 2025
- [2] Briggs, W. L., Van Emden Henson The DFT, An Owners Manual for the Discrete Fourier Transform, SIAM, 1995
- [3] Brown, J. W., Churchill, R. V. Fourier Series and Boundary Value Problems, McGraw Hill, 2011
- [4] Couch, L. W. Digital and Analog Communication Systems, Pearson, Prentice Hall, 2012
- [5] Courant, R., Hilbert, D. Methods of Mathematical Physics, Wiley-VCH, New York, 1993
- [6] Dautray, R., Lions, J. L. Mathematical Analysis and Numerical Methods for Science and Technology, 6 Vol., Springer, 1992
- [7] Kammeyer, K.-D., Kroschel, K. Digitale Signalverarbeitung, Springer, 2002
- [8] Mason, J. C., Handscomb, D. Chebyshev Polynomials, CRC Press, 2002
- [9] Mint-U, T., Debnath, L. Linear Partial Differential Equations for Scientists and Engineers, Birkhäuser, 2006
- [10] Tolstov, G. P. Fourier Series, Dover, New York, 1976
- [11] Zygmund, A. Trigonometric Series, Cambridge University Press, 2003