$\partial_m{}^n = \partial^n / \partial x_m{}^n$ the nth derivative with respect to the mth component, $\partial^k T = \partial_1{}^{k_1} \ldots \partial_3{}^{k_3} T$, ϕ a test function,

$$< \partial^k T , \phi > = (-1)^{|k|} < T, \partial^k \phi > .$$

Example 25. Point charges, electric field

The Coulomb potential for a point charge q $\delta(x)$, x=(x_1,x_2,x_3), in the origin "disappearing at infinity" is given by the locally integrable function

$$q/(4\pi\epsilon_0 |x|) = q\,(x_1{}^2 + x_2{}^2 + x_3{}^2)^{-1/2}/(4\pi\epsilon_0) \text{ for } x=(x_1,x_2,x_3)$$

More general, the electrostatic potential of point charges q[[i]] at points p[[i]], Norm[x]= $(x_1{}^2 + x_2{}^2 + x_3{}^2)^{-1/2}$, in Heaviside-Lorentz-Units with ϵ_0 set to one, is given by

```
electroStaticPotential[q_, p_, x_] :=
 Sum[ q[[i]] / Norm[x - p[[i]]], {i, Length[q]}] / (4 π)
```

The singular distribution $\rho(x) = \sum_{k=1}^{n} q_k \delta(x - x_k)$ is a generalized charge density for n electric point charges q_k at x_k, k=1,...,n, x and x_k in $\mathbb{R}^3$. The Coulomb potential arises from the convolution of $-\rho(x)/\epsilon_0$ with the fundamental solution $-1/(4\pi|x|)$ of the potential equation

$$\Delta u = -\rho/\epsilon_0$$

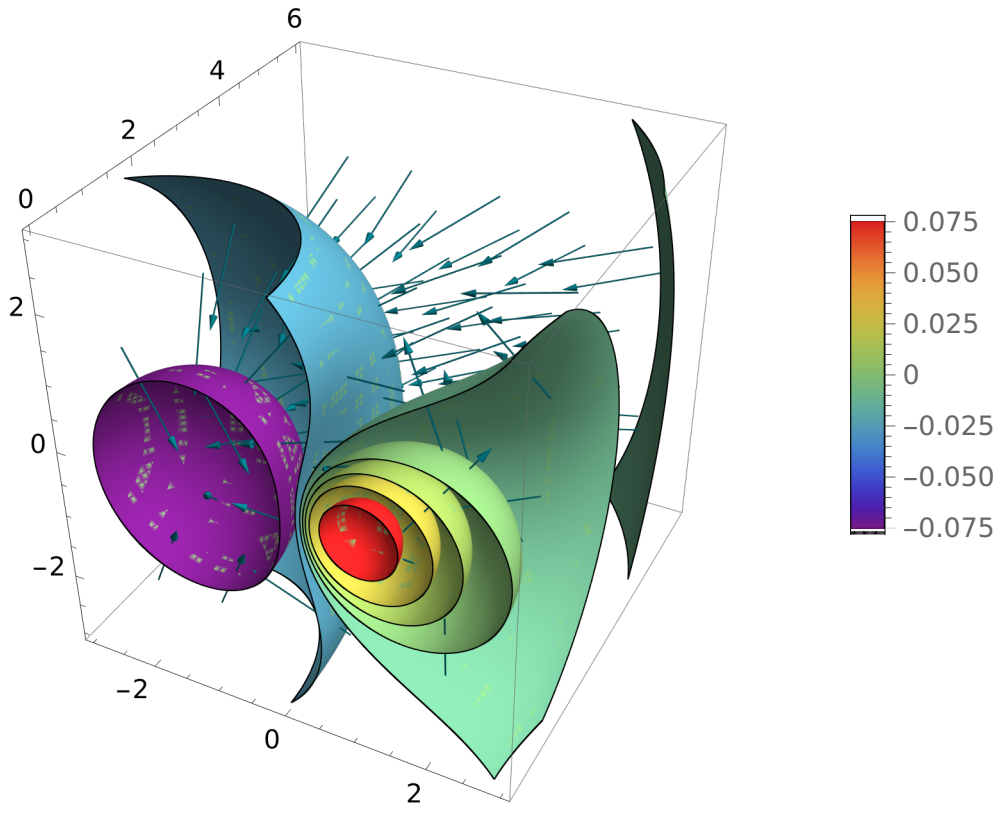
(see section 2.3 and chapter 3). The term |x| is the norm of a vector x.

For illustration, we plot some equipotential surfaces and illustrate the electric field for q={−2, +1} in p={{−1, 0, 0}, {1, 0, 0}} in the half-space y>0:

```
In[•]:= electricField[{q1_, q2_}, {{x1_, y1_, z1_}, {x2_, y2_, z2_}}] =
  -D[electroStaticPotential[{q1, q2},
      {{x1, y1, z1}, {x2, y2, z2}}, {x, y, z}], {{x, y, z}}] /.
   Abs'[x_] :> x / Sqrt[x^2];
```

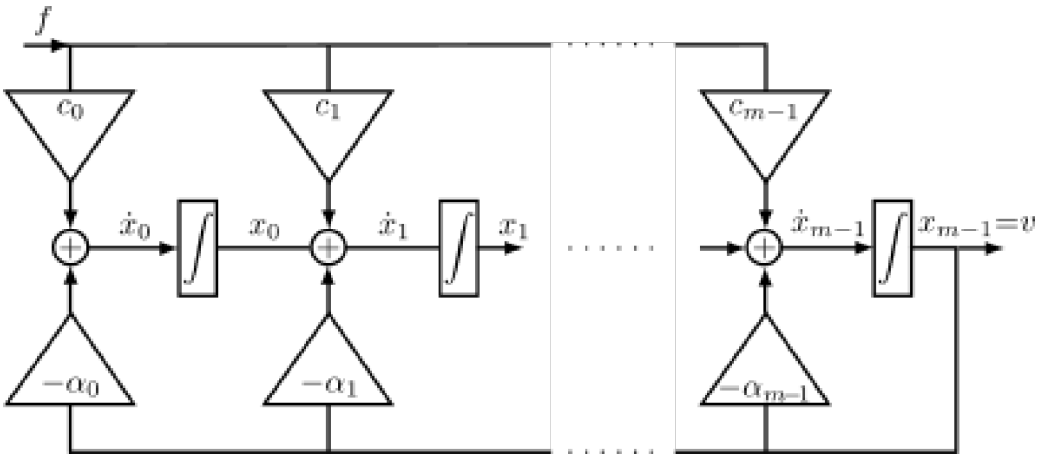
```
In[•]:= p1 = ContourPlot3D[
           Evaluate[electroStaticPotential[{-2, +1},
             {{-1, 0, 0}, {1, 0, 0}}, {x, y, z}]], {x, -3, 3},
           {y, 0, 6}, {z, -3, 3},
           Contours → {-0.075, -0.025, -0.01, 0, 0.01, 0.025,
             0.075},  Mesh → None, PlotLegends → Automatic,
           ColorFunction → "Rainbow"];
         v = VectorPlot3D[electricField[{-2, 1}, {{-1, 0, 0}, {1, 0, 0}}],
           {x, -2, 2}, {y, 0, 6}, {z, -2, 2}, VectorStyle → "Arrow3D",
           VectorPoints → 5, VectorScale → {.2, Scaled[0.1]},
           Axes → True, Boxed → True, AxesStyle → Black];
         Show[p1, v]
```

Out[•]=



We will come back to this example, when we treat later (ch. 3) fundamental solutions of partial differential equations.

Out[•]=



We compute e^{At} s(t) and give out the response $\boldsymbol{v}$ of the filter for the input k δ. That is the convolution of G[[3,1]] below with F[[1]] shown below.

In[•]:=
```
G = FullSimplify[MatrixExp[A s] HeavisideTheta[s]];
(* long output suppressed *)
G[[3, 1]]
```

Out[•]=

$$\frac{1}{3\,\Omega^2}\,e^{-s\,\Omega}\,\text{HeavisideTheta[s]}$$

$$\left(3+e^{\frac{s\,\Omega}{2}}\left(-3\,\text{Cos}\left[\frac{1}{2}\,\sqrt{3}\;s\,\Omega\right]+\sqrt{3}\;\text{Sin}\left[\frac{1}{2}\,\sqrt{3}\;s\,\Omega\right]\right)\right)$$

Now, the response of the filter. It is the convolution of G[[3,1]] with F[[1]]:

In[•]:=
```
response[t_] = FullSimplify[Convolve[G[[3, 1]], F[[1]], s, t]]
 (* Unit is [V] *)
```

Out[•]=

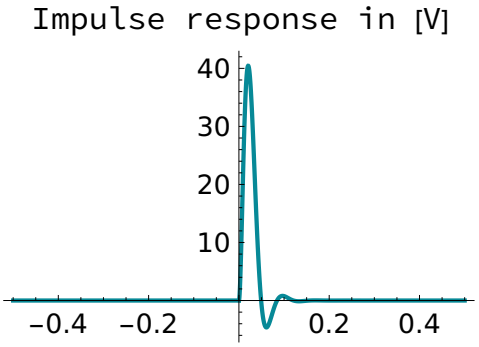
$$\frac{1}{3}\,e^{-t\,\Omega}\,k\,\Omega\,\text{HeavisideTheta[t]}$$

$$\left(3+e^{\frac{t\,\Omega}{2}}\left(-3\,\text{Cos}\left[\frac{1}{2}\,\sqrt{3}\;t\,\Omega\right]+\sqrt{3}\;\text{Sin}\left[\frac{1}{2}\,\sqrt{3}\;t\,\Omega\right]\right)\right)$$

We recognize from that impulse response that the system is stable. It is a regular and rapidly decreasing distribution, Next a plot of the impulse response for an angular cutoff frequency Ω=100 rad/s

```
In[•]:= Ω = 100; k = 1; (* Ω cutoff angular frequency, k = 1 [Vs] *)
        Plot[response[t], {t, -0.5, 0.5}, PlotRange → All,
         PlotLegends → Placed["Impulse response in [V]", Above]]
```

Out[•]=



Here, we compute the frequency response for the angular cutoff frequency Ω=10.000 rad/s. It is the Fourier transform of the impulse response (see later ch. 6). This Fourier transform is called the frequency response. We plot its absolute values.

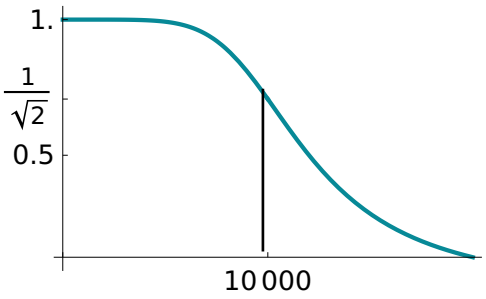
```
In[•]:=
        Ω = 10 000; k = 1;(* cutoff angular frequency Ω,
        impulse strength k *)
        freqresponse[ω_] =
         FullSimplify[FourierTransform[response[t], t, ω,
           FourierParameters → {1, -1}]]
```

Out[•]=

$$\frac{1\,000\,000\,000\,000\,i}{(-10\,000\,i+\omega)\left(-100\,000\,000+\omega\left(-10\,000\,i+\omega\right)\right)}$$

```
In[•]:= p1 = Plot[Abs[freqresponse[ω]], {ω, 0, 20 000},
           PlotRange → All,
           Ticks → {{0, 10 000}, {0, 0.5, 1 / Sqrt[2], 1.0}},
           ImageSize → Small]
```

In the following graphics we see the magnitude response of the 3rd order Butterworth low-pass, cutoff angular frequency 10.000 rad/s, 3 dB damping at 10.000 rad/s



If we are not interested in the underlying mathematics, we can compute the impulse response directly with Mathematica, provided we know that with k=1 it is the third component of the homogeneous system's solution. In both cases shown we use essentially 5 Mathematica commands:

```
In[•]:=
X[t_] = {x[t], y[t], z[t]};
system = X'[t] == A.X[t];
sol = DSolve[system, {x, y, z}, t];
particularsol =
  Flatten[
   Table[{x[t], y[t], z[t]} /. sol /.
     {C[1] → Ω^3, C[2] → 0, C[3] → 0}, 3]];
impulseresponse2[t_] = particularsol[[3]] HeavisideTheta[t]
```

$$\text{Out[•]= } -\frac{10\,000}{3}\, e^{-10\,000\,t}\,\text{HeavisideTheta}[t]\left(-3+3\,e^{5000\,t}\,\text{Cos}\left[5000\,\sqrt{3}\,t\right]-\sqrt{3}\,e^{5000\,t}\,\text{Sin}\left[5000\,\sqrt{3}\,t\right]\right)$$

Thus, we have obtained the same impulse response as before:

```
In[•]:= FullSimplify[response[t] - impulseresponse2[t]]
Out[•]= 0
```

Partial Differential Equations with Constant Coefficients, Fundamental Solutions

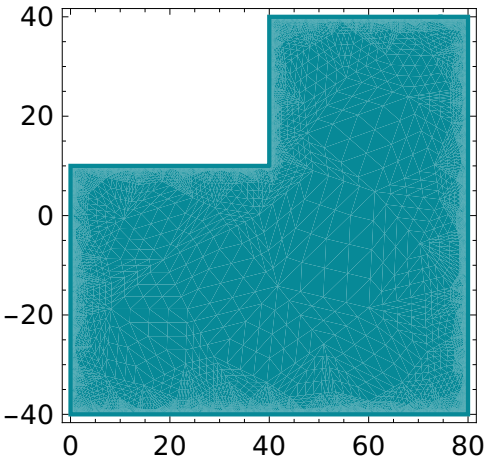
In its most general form a linear partial differential equation with constant coefficients for n real variables $x_1,\ldots, x_n$ is written as $P(\partial)T=F$ with $P(\partial)=\sum_{|k|\le m} a_k\partial^k$, $a_k\in\mathbb{C}$, and multi-indexes $k=(k_1,\ldots,k_n)$. It has order

Example 38. Wave propagation, Huygens's principle

This example illustrates the propagation of a periodic wave packet at the origin in an L-shaped planar region. The wave propagates along an absorbing wall with a corner, at which the wave is diffracted. We observe the decay of the amplitudes with increasing distance from the source, diffraction behind the corner according to the Huygens principle, a latency in the "shadow space" behind the vertex and interferences of the diffraction part with the source part. With FEM we compute a numerically approximate **weak solution**. At first we define the region, in which the wave propagates. This is the L-shaped region in gray shown below.

```
In[•]:= c = 2; (* Propagation speed m/s*)
        ω = Pi / 2; (* Angular frequency of the wave packet *)
        tmax = 44; (* Observation time in s*)
        region = ImplicitRegion[
          Or[0 < x < 80 && -40 < y < 10, 40 < x < 80 && 10 < y < 40], {x, y}]
        (* x,y in m*)
        RegionPlot[region]

Out[•]= ImplicitRegion[
         (0 < x < 80 && -40 < y < 10) || (40 < x < 80 && 10 < y < 40), {x, y}]
```

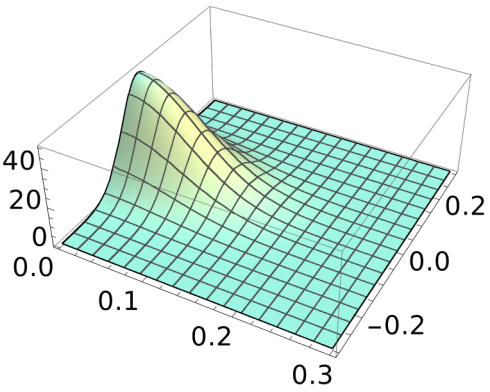
Out[•]=



The **source** is the following periodic wave packet located at the origin. For the numerical solution we need a regular source instead of a point

source like Sin[ω t] DiracDelta[x,y].

```
In[•]:= f[x_, y_] = 50 Exp[-(x^2 + y^2)/0.1^2];
        source[x_, y_, t_] := Sin[ω t] UnitStep[t] f[x, y];
        Plot3D[f[x, y], {x, 0, 0.3}, {y, -0.3, 0.3}, PlotRange → All]
```



Now the PDE of the inhomogeneous 2D-wave equation and initial conditions:

```
pde = D[u[x, y, t], {t, 2}] ==
    c^2 Laplacian[u[x, y, t], {x, y}] + source[x, y, t] +
     NeumannValue[-D[u[x, y, t], t]/c, True];
ics = {u[x, y, 0] == 0, Derivative[0, 0, 1][u][x, y, 0] == 0};
```

Computation of a weak solution with the method of finite elements with the command **NDSolveValue** in Mathematica and specification of options. The meshing and choice of the interpolation functions is left to Mathematica. The used options for NDSolveValue

Method->{"MethodOfLines","SpatialDiscretization"->{"FiniteElement","MeshOptions"->{"MaxCellMeasure"->0.6}}}]

cause in the spatial discretization a finite element mesh, while the method of lines is a technique for solving partial differential equations by discretizing in all but one dimension and then integrating the semi-discrete problem as a system of ODEs or DAEs.

```
In[•]:= sol = NDSolveValue[{pde, ics}, u, {x, y} ∈ region,
          {t, 0, tmax},
          Method → {"MethodOfLines",
            "SpatialDiscretization" →
             {"FiniteElement",
              "MeshOptions" → {"MaxCellMeasure" → 0.6}}}]
```

```
Out[•]= InterpolatingFunction[
```

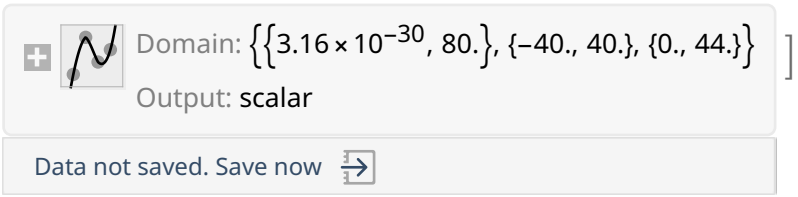


Illustration of the Result

We prepare a graphical representation of the numerical result. We define a wall as [0,40]x[10,40], and plot it jointly with two snapshots of the solution, one at t=25s, the other at t=40s. The wall is assumed to be completely absorbing (Neumann condition in the PDE). The wave propagates along the wall. By the Huygens principle a secondary wave starts at the corner and we get a total of a bended wave after the corner and interference of the waves coming directly from the source and that coming from the corner.

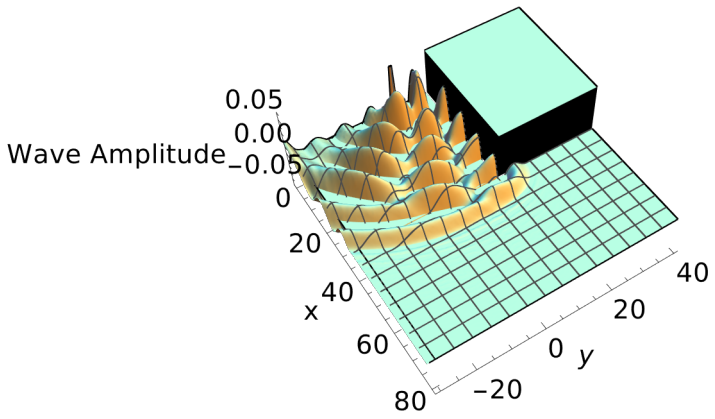
```
wall[x_, y_] = 0.06 (HeavisideTheta[x] - HeavisideTheta[x - 40])
    (HeavisideTheta[y - 10] - HeavisideTheta[y - 40]);
p0 = Plot3D[wall[x, y], {x, 0, 80}, {y, -30, 40},
    Boxed → False, Axes → {True, True, True},
    PlotPoints → 200, PlotRange → {-0.05, 0.06},
    Filling → Axis, FillingStyle → Opacity[1, Black],
    Mesh → None, AxesLabel → {"x", y, "  Wave Amplitude"}];

p1 = Plot3D[sol[x, y, 25], {x, 0.01, 80}, {y, -40, 40},
    Boxed → False, PlotPoints → 200,
    PlotRange → {-0.05, 0.06}, Axes → {True, True, True},
    AxesLabel → {"x", y, " Wave Amplitude"}];
```

```
p2 = Plot3D[sol[x, y, 40], {x, 0.01, 80}, {y, -40, 40},
    Boxed → False, PlotPoints → 200,
    PlotRange → {-0.05, 0.06}, Axes → {True, True, True},
    AxesLabel → {"x", y, " Wave Amplitude"}];
```

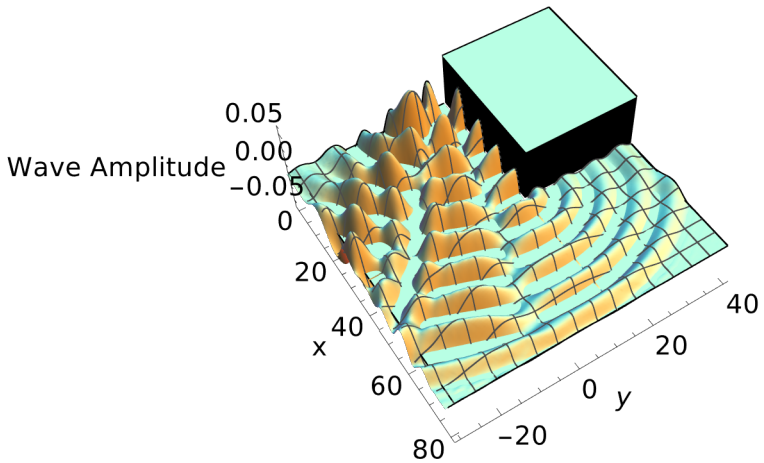
We see two snapshots of the wave, the first after 25s, the second after 40s. This gives an impression of the wave's dynamics (see the area around the corner). The snapshots show also the decrease in amplitude with increasing distance from the source, the Huygens principle in the shadow space behind the corner, and thus the wave's diffraction.

In[•]:=
```
p3 = Show[p0, p1]
p4 = Show[p0, p2]
```

Out[•]=



Out[•]=



Example 40. Deflection of a beam under load

As our last example we consider a linearly elastic rectangular titanium beam that is fixed at both ends (x = 0 and x=10). We apply a surface load of 15 kN (about the force by the weight of 20 persons). For linearly elastic material models in solid mechanics, the equilibrium, continuity and output stress measures are Cauchy stresses.

The stress tensor σ, the strain tensor ε, displacement vector u [m] and body load vectors b [N/m^3] or g [m/s^2] , ρ [kg/m^3] the material density, are based on the differential operator

$$\nabla\sigma\ (\varepsilon\ (\mathrm{u}\ (\mathrm{t},\ \mathrm{x}))) - \mathrm{b} - \rho\ \mathrm{g}$$

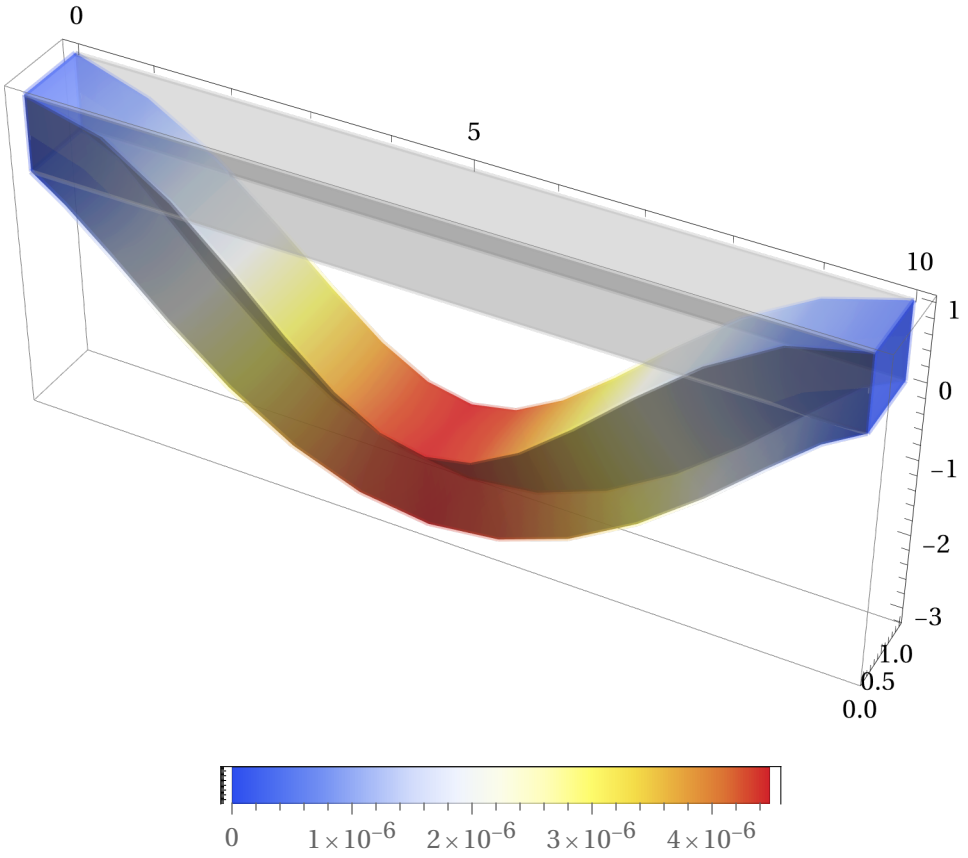
Here, the Nabla operator ∇ means divergence.

We use the material constants available in Mathematica as well as the implemented computation of the according stress tensor $\sigma\ (\varepsilon\ (\mathrm{u}\ (\mathrm{t},\ \mathrm{x})))$.
Now the Mathematica code, the FEM computation and a graphical representation:

```
In[25]:= Needs["NDSolve`FEM`"]
         $HistoryLength = 0;

In[27]:= Ω = Cuboid[{0, 0, 0}, {10, 1, 1}];
         vars = {{u[x, y, z], v[x, y, z], w[x, y, z]}, {x, y, z}};
         pars = <|"Material" → Entity["Element", "Titanium"]|>

Out[29]= ⟨|Material → titanium|⟩

In[30]:= A = SolidBoundaryLoadValue[z == 1, vars,
           pars, <|"Force" → {0, 0, Quantity[-15 000, "Newtons"]}|>]

Out[30]= {NeumannValue[0, z == 1], NeumannValue[0, z == 1], NeumannValue[
                                    15 000
          -------------------------------------------------------------------
          FEMNBoundaryIntegrate[1, {x, y, z}, SpatialRegion, z == 1]
          , z == 1]}
```

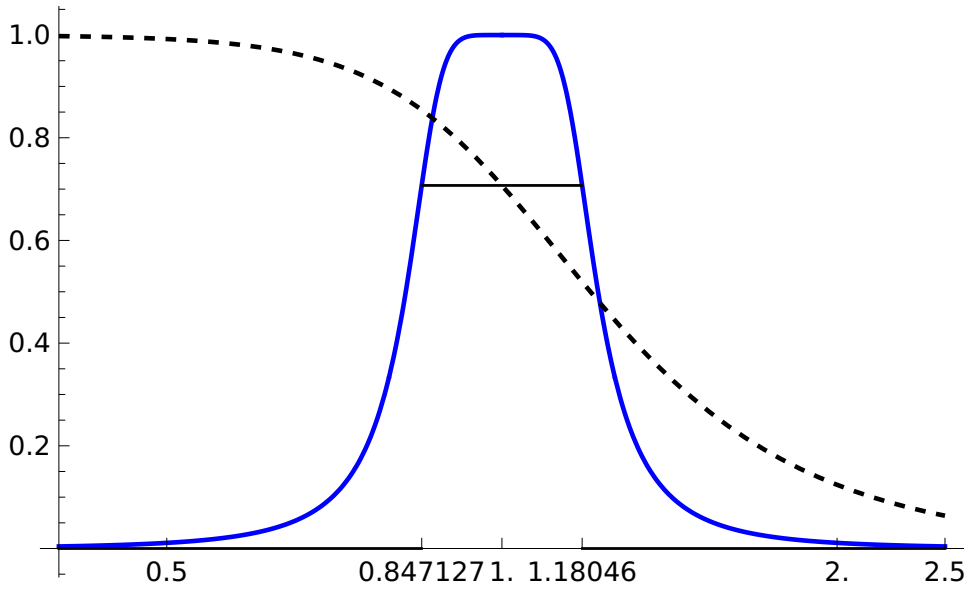
In[35]:=
```
VectorDisplacementPlot3D[beamDisplacement, {x, y, z} ∈ Ω,
 PlotLegends → Automatic,
 ColorFunction → "TemperatureMap"]
```

Out[35]=



Finally, we plot the von Mises stress. The unit is [Pa].

Out[•]=



Example 57. The Hilbert transform

We consider the Hilbert transform

$$\mathbf{H(f)(t)= \frac{1}{\pi}\ vp(\ t^{-1})\ *\ f(t)\ =\ \frac{1}{\pi}\ \int_{-\infty}^{\infty} f(s)\,/\,(t-s)\,ds.}$$

The singular integral is a principle value integral. It is a non-causal filter. We consider functions f, for which the convolution with vp(t^{-1}) is possible and the Fourier exchange theorem holds (p. 12, p. 95). For details on the theory it is referred to [6] and [9]. The Hilbert transform shifts every frequency component of a function by a phase of $-\frac{\pi}{2}$ radians.

A common application of the Hilbert transform in signal processing is extracting the envelope (instantaneous amplitude) of a signal. By creating an analytic signal - where the original signal is the real part and its Hilbert transform is the imaginary part - the magnitude of this complex signal provides the smooth outline (envelope) of a rapidly oscillating wave. We give some simple examples with Mathematica 14.3, beginning with the cosine function.

First, we compute the singular integral, then we use the command HilbertTransform, which is available starting in Mathematica 14.3:

```
Integrate[Cos[s]/(Pi (t - s)), {s, -Infinity, Infinity},
 PrincipalValue → True]
(* using the principle value integral *)
HilbertTransform[Cos[t], t, t]
(* using the implemented Mathematica command *)
```

Out[•]= Sin[t] if t ∈ ℝ

Out[•]= Sin[t]

Next we check the Fourier exchange theorem with the cosine:

In[•]:=
```
FourierTransform[Cos[s], s, w, FourierParameters → {1, -1}]
FourierTransform[1/s/Pi, s, w, FourierParameters → {1, -1}]
```

Out[•]= π DiracDelta[-1 + w] + π DiracDelta[1 + w]

Out[•]= -i Sign[w]

In[•]:=
```
FullSimplify[InverseFourierTransform[% × %%, w, t,
  FourierParameters → {1, -1}]]
```

Out[•]= Sin[t]

In a second example we illustrate a trigonometric function f (black), its Hilbert transform htf (blue) and the **magnitude of the analytic signal F = f +i htf as envelope (red)**. The envelope shows the 50 Hz oscillation around the constant one.

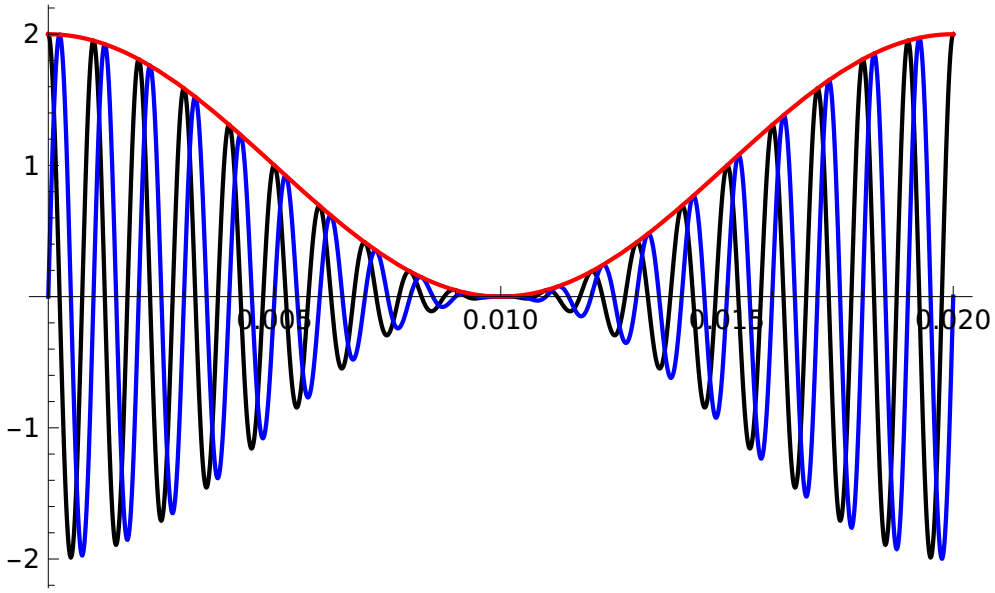
```
f[t_] = (1 + Cos[2 π 50 t]) Cos[2 π 1000 t];
(* a 50 Hz cosine with an amplitude modulation by
 1 kHz *)
htf[t_] = HilbertTransform[f[s], s, t];
(* HilbertTransform is available starting with
 version 14.3 in Mathematica *)
```

The analytic signal is

```
F[t_] = f[t] + i htf[t];
```

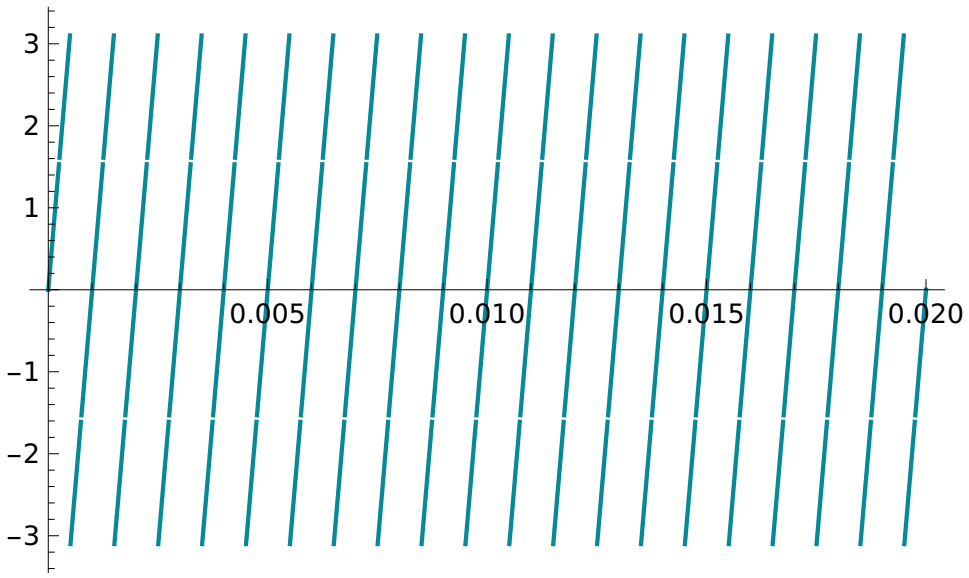
```
In[•]:= Plot[{f[t], htf[t], Abs[F[t]]}, {t, 0, 0.02},
         PlotStyle → {Black, Blue, Red}, PlotRange → All]
```

Out[•]=




```
In[•]:= Plot[Arg[F[t]], {t, 0, 0.02}]
```

Out[•]=



Below, we see the spectrum of the signal f, followed by the spectrum of the analytic signal, which has twice the pulse strength and deletes the spectrum on the negative half-axis:

```
In[•]:= FourierTransform[f[t], t, ω, FourierParameters → {1, -1}]
Out[•]= 1/2 π DiracDelta[-2100 π + ω] + π DiracDelta[-2000 π + ω] +
        1/2 π DiracDelta[-1900 π + ω] + 1/2 π DiracDelta[1900 π + ω] +
        π DiracDelta[2000 π + ω] + 1/2 π DiracDelta[2100 π + ω]

In[•]:= FourierTransform[F[t], t, ω, FourierParameters → {1, -1}]
Out[•]= π DiracDelta[-2100 π + ω] +
        2 π DiracDelta[-2000 π + ω] + π DiracDelta[-1900 π + ω]
```

Unlike conventional smoothing filters, Hilbert transform envelope extraction does not lose information, resulting in higher-fidelity images and data. It is used in vibration analysis and structural monitoring to find the slowly varying amplitude of a rapidly vibrating signal (see above), helping to identify faults. Applications in real time discrete signal processing usually approximate the Hilbert transform by causal finite impulse response filters (FIR filter) as considered in the following section below. The Hilbert transform enables, for example, the calculation of the precise pulse energy, reducing noise and improving imaging of ultrasonic echoes. For specific examples of applications, please search the Internet or consult literature, such as F.W. King [5].

6.3 Discrete Linear Filters

For the treatment of discrete filters we use impulse sequences, generalized Fourier series as their Fourier transforms and the z-transform. A detailed presentation can be found in the authors textbook [1]. For a discrete signal $\mathbf{x}=\sum_{n=-\infty}^{\infty} x_n \delta_n$ in $\boldsymbol{S}'$ we have its Fourier transform $\mathrm{F}(\mathbf{x})(\omega) = \sum_{n=-\infty}^{\infty} x_n e^{-i\, n\, a\, \omega}$ as a generalized Fourier series and its **z-transform** $\mathrm{X}(z)=\sum_{n=-\infty}^{\infty} x_n z^{-n}$. Thus, $\mathrm{F}(\mathbf{x})=\mathrm{X}(e^{i\ a\,\omega})$. The constant a in $\delta_n = \delta(t - n\ a)$ is a fixed time constant and $1/a$ corresponds to a sampling frequency, if the coefficients x_n are sampling values of a continuous signal. We observe that the spectrum $\mathrm{F}(\mathbf{x})$ is periodic.

There are also other possible realizations of the same filter (characterized by different parentheses in the difference representation or variations in the representation of H). For this, please refer to, for example, U. Tietze, Ch. Schenk [8] and others.
If all coefficients b_m, m=1,...,N, are zero, the filter has a finite impulse response (is a **FIR** filter). Otherwise it has an infinite impulse response (is a **IIR** filter).

We now show how one can design a discrete IIR filter using the frequency response of an according analog filter with the help of the so-called Bilinear Transformation.

Example 59. Derivation of a discrete IIR Butterworth lowpass filter from a corresponding analog filter

Starting point of the example is a rational transfer function R(s)=1/P(s), frequency response R($i\omega$) of a Butterworth lowpass filter. The polynomial is the characteristic polynomial of the differential equation P(δ)u=δ, which determines the lowpass filter (see example 35). To obtain the **2π/a-periodic** frequency response of an according discrete filter we map the frequency axis one-to one to the interval]-π/a , π/a[. A map, which accomplishes this, is

$$\mathbf{T(\omega) = \frac{2}{A}\,\arctan\left(\frac{\omega\, a}{2\,V}\right)\ with\ inverse\ T^{-1}(\Omega) = \frac{2}{a}\,V\,\tan\left(\frac{\Omega\, a}{2}\right) = \omega.}$$

Ω is the angular frequency in the discrete case, 1/a the sampling frequency, V is a factor such that prior bias can be chosen, for example, that the required cutoff frequency ω_c is a fixed point of the mapping. Then, one can define the frequency response $H(e^{i\Omega a})$ of the proposed discrete filter by

$$\mathbf{H(e^{i\Omega a}) = R(\,i\,T^{-1}(\Omega))\ \ for\ \Omega \in\,]\,-\pi/a\,,\,\pi/a\,[.}$$

From addition theorems for the tangent function it follows with $\mathbf{z = e^{i\,\Omega\, a}}$

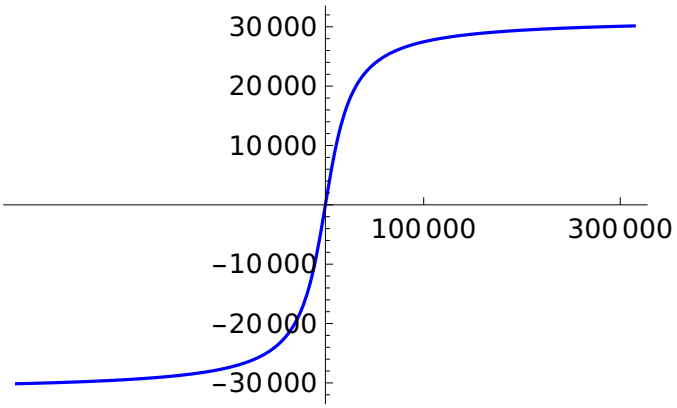
$$\mathbf{H(e^{i\Omega a}) = H(z) = R(\,i\,\frac{2}{a}\,V\,\tan\left(\frac{\Omega a}{2}\right)) = R\left(\frac{2\,V}{a}\,\frac{1-z^{-1}}{1+z^{-1}}\right) = R(B(z)).}$$

The transformation $B(z)=\frac{2V}{a}\frac{1-z^{-1}}{1+z^{-1}}=s$ for $z\in\mathbb{C}$, $B(-1)=\infty$, $B(\infty)=\frac{2V}{a}$ has the inverse $B^{-1}(s)=\frac{2V/a+s}{2V/a-s}=z$.

This **Möbius transformation** is called **Bilinear transformation.** The last equation shows that **H(z) is rational in z**. Thus, the obtained discrete filter can immediately realized as in the block diagram of the preceding example 58. The left half-plane of $\mathbb{C}$ is mapped by B^{-1} to the interior of the unit circle, the imaginary axis to the unit circle. Let's illustrate what was described: First a Plot of the mapping T, afterwards

```
In[•]:= a := 10^-4; ωc = 200 π; V = ωc a / 2 Cot[ωc a / 2];
        T[ω_] = 2 / a ArcTan[ω a / (2 V)]; (* mapping ℝ to ]-π/a ,
        π/a [ *)
        T[ωc] (* show te fixed point of T *)
        Plot[ T[ω], {ω, -100 000 π, 100 000 π}, PlotRange → All,
         PlotStyle → Directive[Blue, Thickness[0.005]],
         Ticks → {{10^5, 3 × 10^5}, Automatic}]
        (* Graph of T *)
Out[•]= 200 π
```

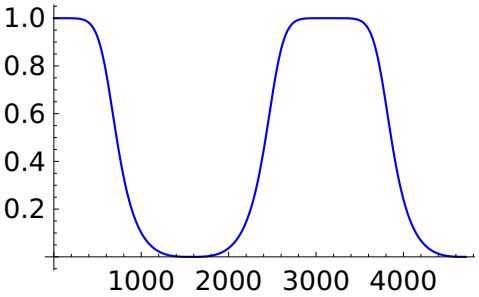
Out[•]=



```
-20 Log10[N[Abs[HH[Exp[-I wc a]]]]]
(* Dämpfung des diskreten Filters bei der
 Grenzfrequenz *)
plot1 = Plot[Abs[HH[Exp[-I Ω a]]], {Ω, 0, 1500 π},
  PlotStyle → Directive[Blue, Thickness[0.005]]]
```

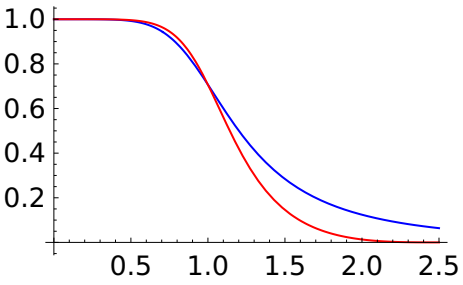
Out[•]= 3.0103

Out[•]=



In[•]:=
```
plot2 := Plot[Abs[R[ s wc]], {s, 0, 2.5},
  PlotStyle → Directive[Blue, Thickness[0.005]]]
plot3 := Plot[Abs[HH[Exp[-I s wc a]]], {s, 0, 2.5},
  PlotStyle → Directive[Red, Thickness[0.005]]]
Show[{plot2, plot3}]
```

Out[•]=



Of course, Mathematica has implemented methods to compute discrete filters of various types from given analog transfer models and vice versa. Let's compare what Mathematica computes with the ButterworthFilterModel and BilinearTransform as method. The difference comes from the fact that we have set the cutoff frequency as a fixed point of the transformation so that the magnitude of the discrete filter equals that of the analog filter at this point. The frequency response of the Mathematica computation is plotted blue, the one computed by us is red: