Betrachtung eines Butterworth-TP --- 6.Mai 2007 Ingenieurbüro Baumann, Dorsten

```
reset():ta:=time():DIGITS:=32:w:=2*PI*f:
```

die Eingangsdaten

```
n:=4:fg:=10e3:ue2:=1:wg:=2*PI*fg:
```

die Berechnungen

```
if frac(n/2)=0 then
     k:=n/2:
     b:=[1 $ i=1..k]:
     a:=[2*cos((2*i-1)*PI/2/n) $ i=1..k]:
else
     k:=(n+1)/2:
     b:=[0]:a:=[1]:
     Liste:=[1 $ i=2..k]:b:=b.Liste:delete Liste:
     Liste:=[2*cos((i-1)*PI/n) $ i=2..k]:a:=a.Liste:delete Liste:
end_if:
```

die Koeffizienten für quadratische Glieder ( T(p) = A0 / [ product( 1 + ai*p/wg + bi*(p/wg)^2 ) ] ) bei Normierung auf fg bei Amin

```
delete aQuad,bQuad:for i from 1 to k do
     aQuad[i]:=float(op(a,i)):
     bQuad[i]:=float(op(b,i)):
end_for:
aQuad;bQuad;
```

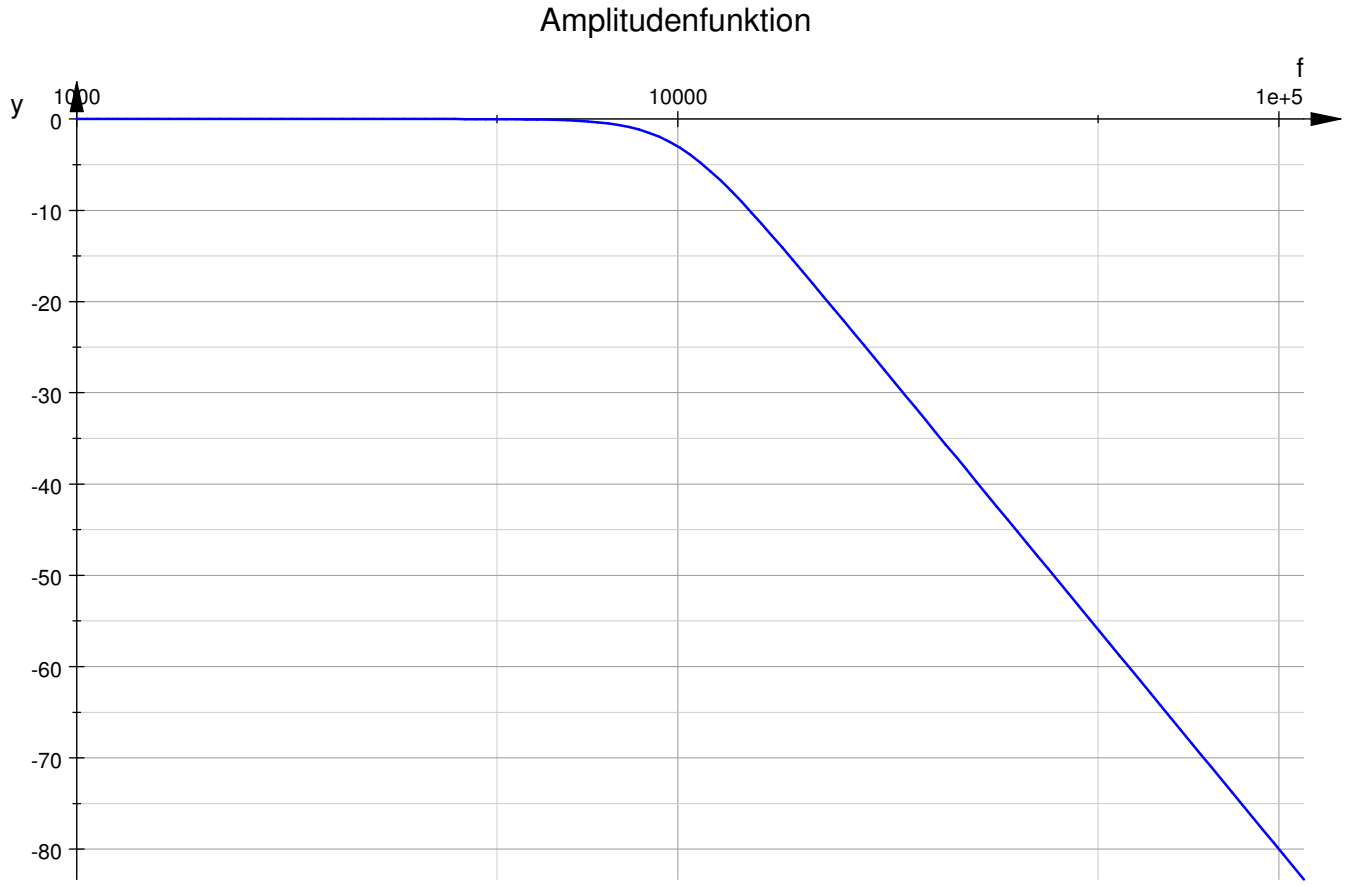
$$\begin{bmatrix} 1 & = 1.8477590650225735122563663787936 \\ 2 & = 0.76536686473017954345691996806 08 \end{bmatrix}$$

$$\begin{bmatrix} 1 & = 1.0 \\ 2 & = 1.0 \end{bmatrix}$$

```
delete i:prod:=(f)->product(1+a[i]*I*w/wg+b[i]*(I*w/wg)^2, i=1..k):
U2U0:=(f)->1/(1+1/ue2)/prod(f):
U2U0dB:=(f)->20*log(10,abs(U2U0(f))):
Winkel:=(f)->180/PI*arg(U2U0(f)):
tg:=(f)->1/2/PI/fg*sum(a[i]*(1+b[i]*(w/wg)^2)/(1+(a[i]^2-
2*b[i])*(w/wg)^2+b[i]^2*(w/wg)^4),i=1..k):
tg1:=(f)->-diff(Winkel(f),f)/360:
```

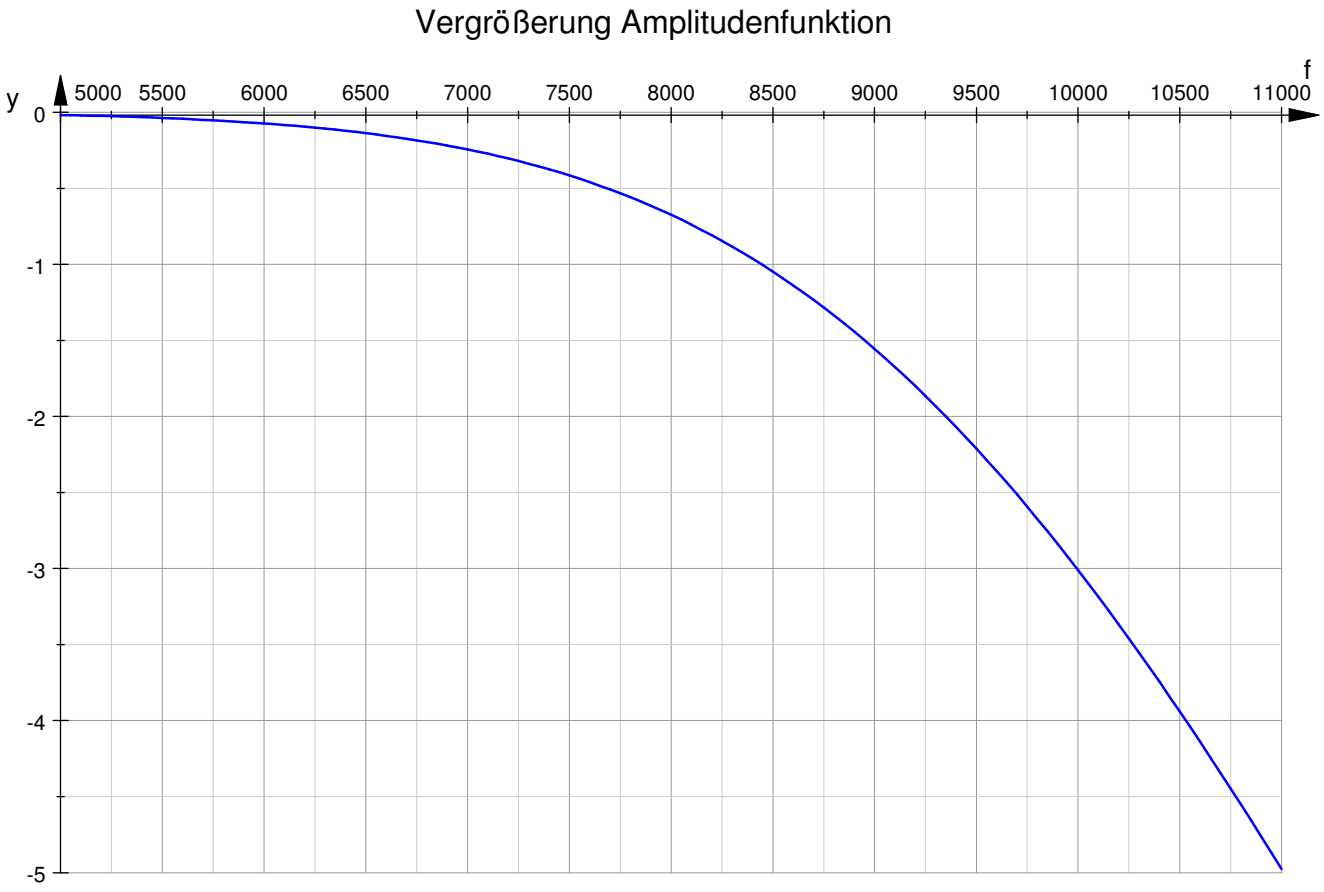
Betrag der Übertragungsfunktion doppelt Logarithmisch

```
delete f:plotfunc2d(U2U0dB(f)+6.02, f=1/10*fg..11*fg,
LegendVisible=FALSE, CoordinateType=LogLin,
           GridVisible=TRUE, SubgridVisible=TRUE,
  Height=120*unit::mm, Width=180*unit::mm,
  Header="Amplitudenfunktion"):
```



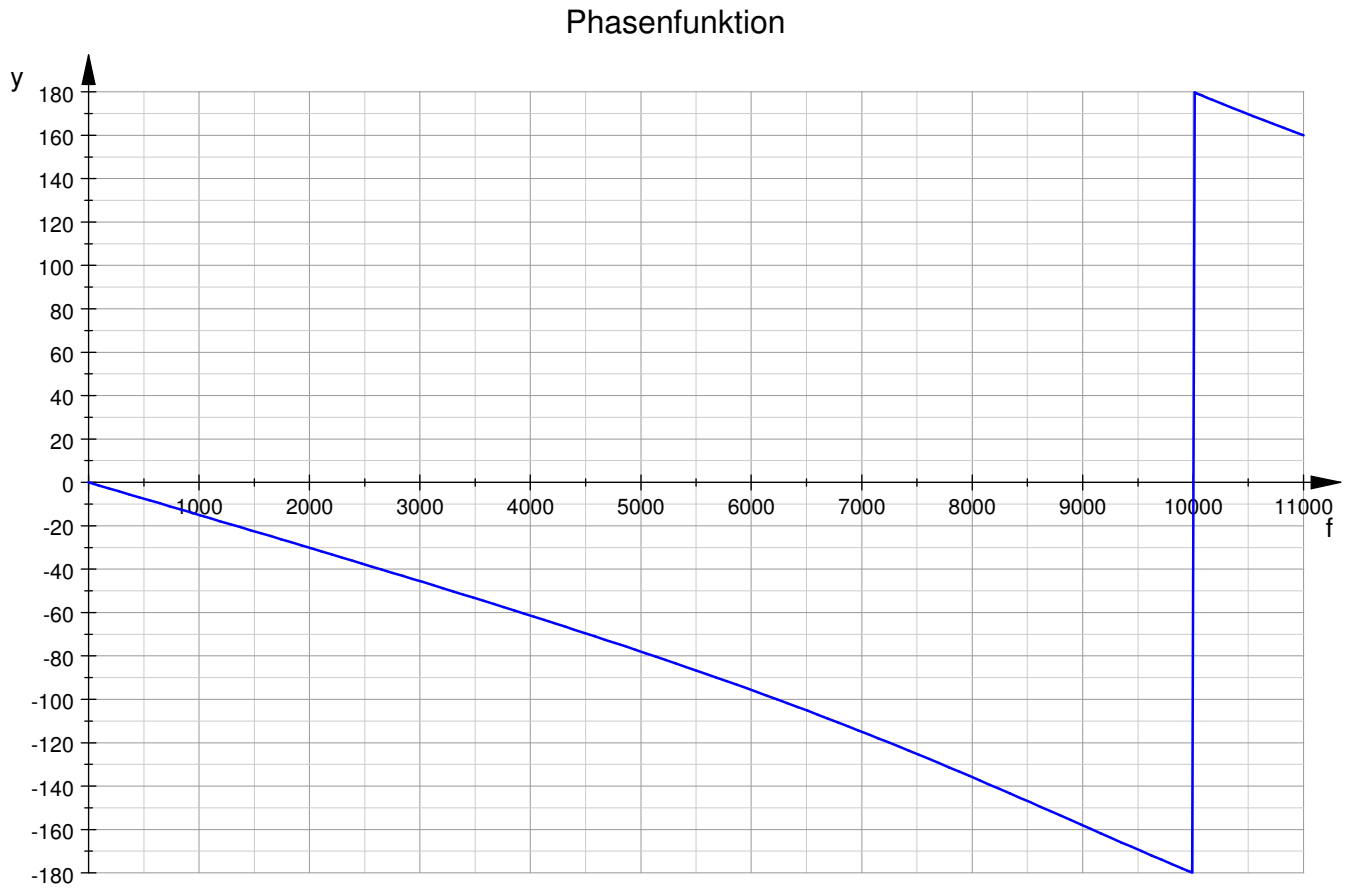

Ausschnittsvergrößerung aus dem Betrag der Übertragungsfunktion, einfach logarithmisch

```
plotfunc2d(U2U0dB(f)+6.02, f=1/2*fg..1.1*fg, LegendVisible=FALSE,
CoordinateType=LinLin,
          GridVisible=TRUE, SubgridVisible=TRUE,
  Height=120*unit::mm, Width=180*unit::mm, Header="Vergrößerung
  Amplitudenfunktion"):
```



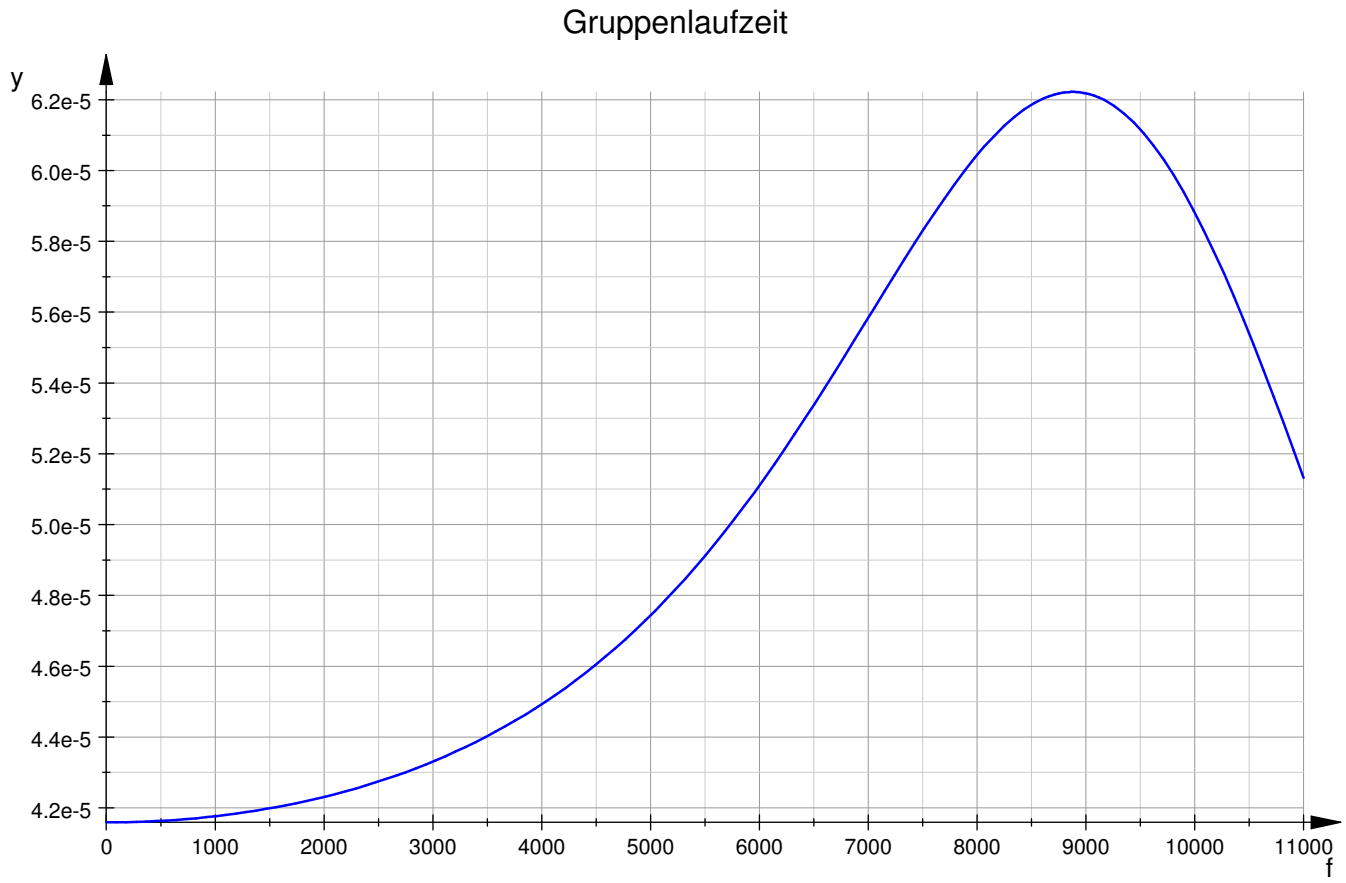

die Phasenverschiebung des Filters

- ```
  plotfunc2d(Winkel(f), f=0..1.1*fg, LegendVisible=FALSE,
             GridVisible=TRUE, SubgridVisible=TRUE,
    Height=120*unit::mm, Width=180*unit::mm, Header="Phasenfunktion"):
  ```
  ```




Berechnete Gruppenlaufzeit

```
plotfunc2d(tg(f), f=0..1.1*fg, LegendVisible=FALSE,
           GridVisible=TRUE, SubgridVisible=TRUE,
  Height=120*unit::mm, Width=180*unit::mm,
  Header="Gruppenlaufzeit"):
```

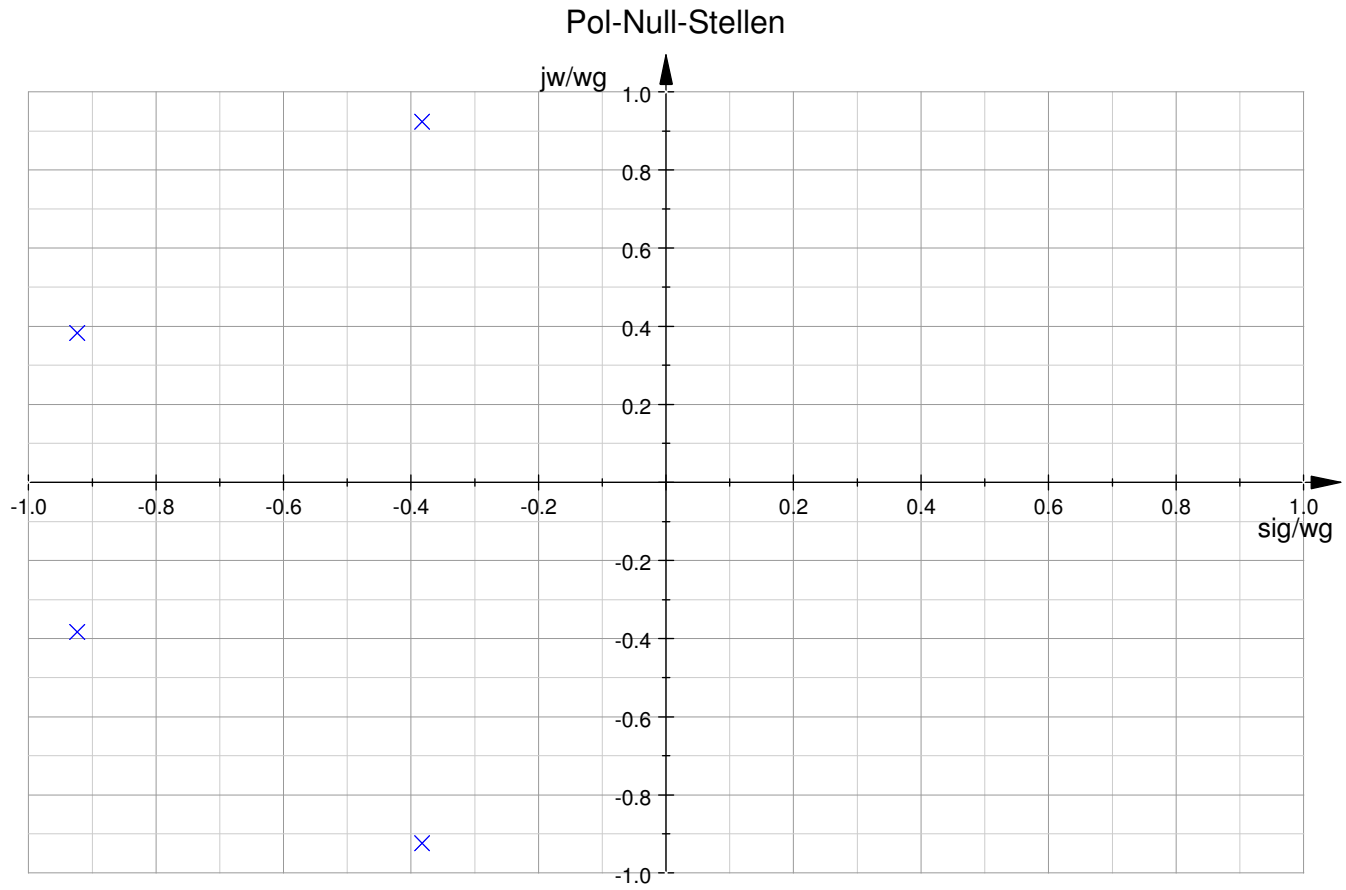



Lage der Pol-Nullstellen des Filters in der komplexen Ebene

- `Pol:=solve(prod(f)=0,f)/fg:`
- ```
  delete PolTab:for i from 1 to n do
       PolTab[i]:=-Im(op(Pol,i))-I*Re(op(Pol,i)):
  end_for:
  ```

(bei den Nullstellen wurden Im und Re vertauscht und mit -1 multipliziert, sonst liegen die komischerweise in der sig-Ebene mit der Hauptachse der Ellipse !)

- `Liste:=[[Re(op(op(PolTab,i),2)),Im(op(op(PolTab,i),2)),RGB::Blue] $ i=1..n]:`
- `Breite:=1:Liste:=Liste.[[Breite,0,RGB::White]].[[0,1,RGB::White]].[[-Breite,0,RGB::White]].[[0,-1,RGB::White]]:`
- ```
  plot(plot::PointList2d(Liste, PointStyle=XCrosses, PointSize=2,
  Color=RGB::Blue, GridVisible=TRUE, SubgridVisible=TRUE,
                          Scaling=Unconstrained,
    AxesTitles=["sig/wg","jw/wg"]), Height=120*unit::mm,
    Width=180*unit::mm, Header="Pol-Null-Stellen"):
  ```




- `PolTab;`

$$\begin{bmatrix} 1 & = -0.382683432365089771728459984030 4 - 0.923879532511286756128183189396 79 \cdot i \\ 2 & = -0.382683432365089771728459984030 4 + 0.923879532511286756128183189396 79 \cdot i \\ 3 & = -0.923879532511286756128183189396 79 - 0.382683432365089771728459984030 4 \cdot i \\ 4 & = -0.923879532511286756128183189396 79 + 0.382683432365089771728459984030 4 \cdot i \end{bmatrix}$$

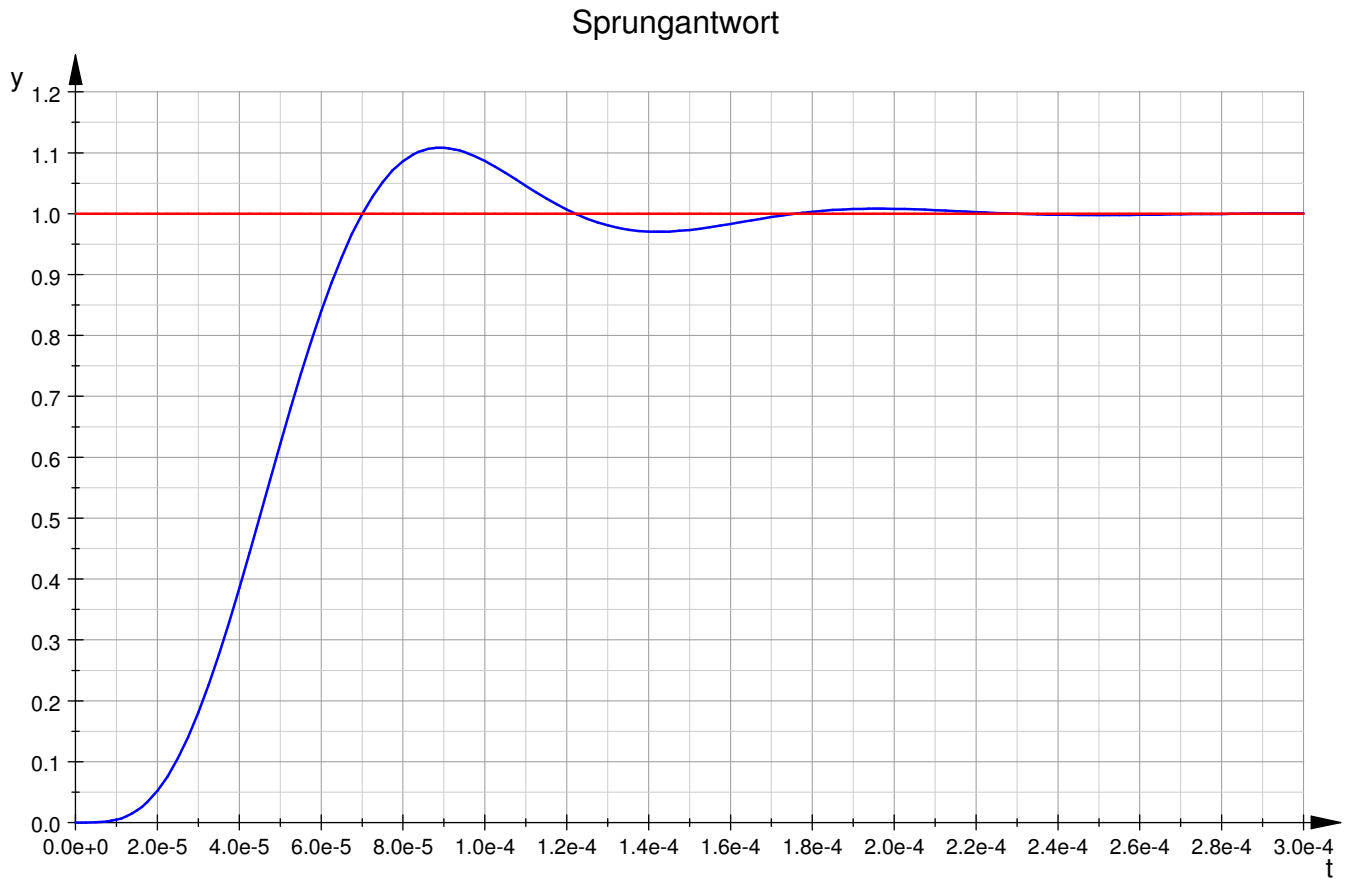
die Grunddämpfung durch ü^2

- `1/(1+1/ue2);`

$$\frac{1}{2}$$

Sprungantwort des Filters ua(t)=invlaplace(2/p*T(p))

- `delete i:prodp:=(p)->product(1+a[i]*p/wg+b[i]*(p/wg)^2, i=1..k):`
- `ua:=(t)->Re(transform::invlaplace(1/(1+ue2)*2/p/prodp(p),p,t)):`
- `plotfunc2d(ua(t), 1, t=0..3/fg, LegendVisible=FALSE,`
  `CoordinateType=LinLin,`
  `GridVisible=TRUE, SubgridVisible=TRUE,`
  `Height=120*unit::mm, Width=180*unit::mm, Header="Sprungantwort",`
  `YMax=1.2):`




Ausschnittsvergrößerung der Sprungantwort

Suchbereich definieren

- `anf:=2e-5:ende:=1e-4:`

Überschwingen in % bei t in us

- `maximum:=op(numeric::solve(diff(ua(t),t)=0,t=anf..ende,RestrictedSearch),1):`
- `(ua(maximum)-1)*100;maximum/1e-6;`

10.830150888537364280420386368286

89.091495433129426852445810912783

to für ua(t)=1/2 in us

- `tx:=op(numeric::solve(Re(ua(t))=1/2,t=anf..maximum,RestrictedSearch),1):tx/1e-6;`

44.885890189741794003857925638927

die Einschwingzeit tau in us und die daraus resultierende Grenzfrequenz in kHz

- `m:=ua'(t):t:=tx:m:=float(m):delete t:yt:=t->1/2-m*(tx-t):`
- `tau:=op(solve(yt(t)=1,t),1)-op(solve(yt(t)=0,t),1):tau/1e-`

```
6;1/2/tau/1e3;
```

41.776687025127236519570794894073

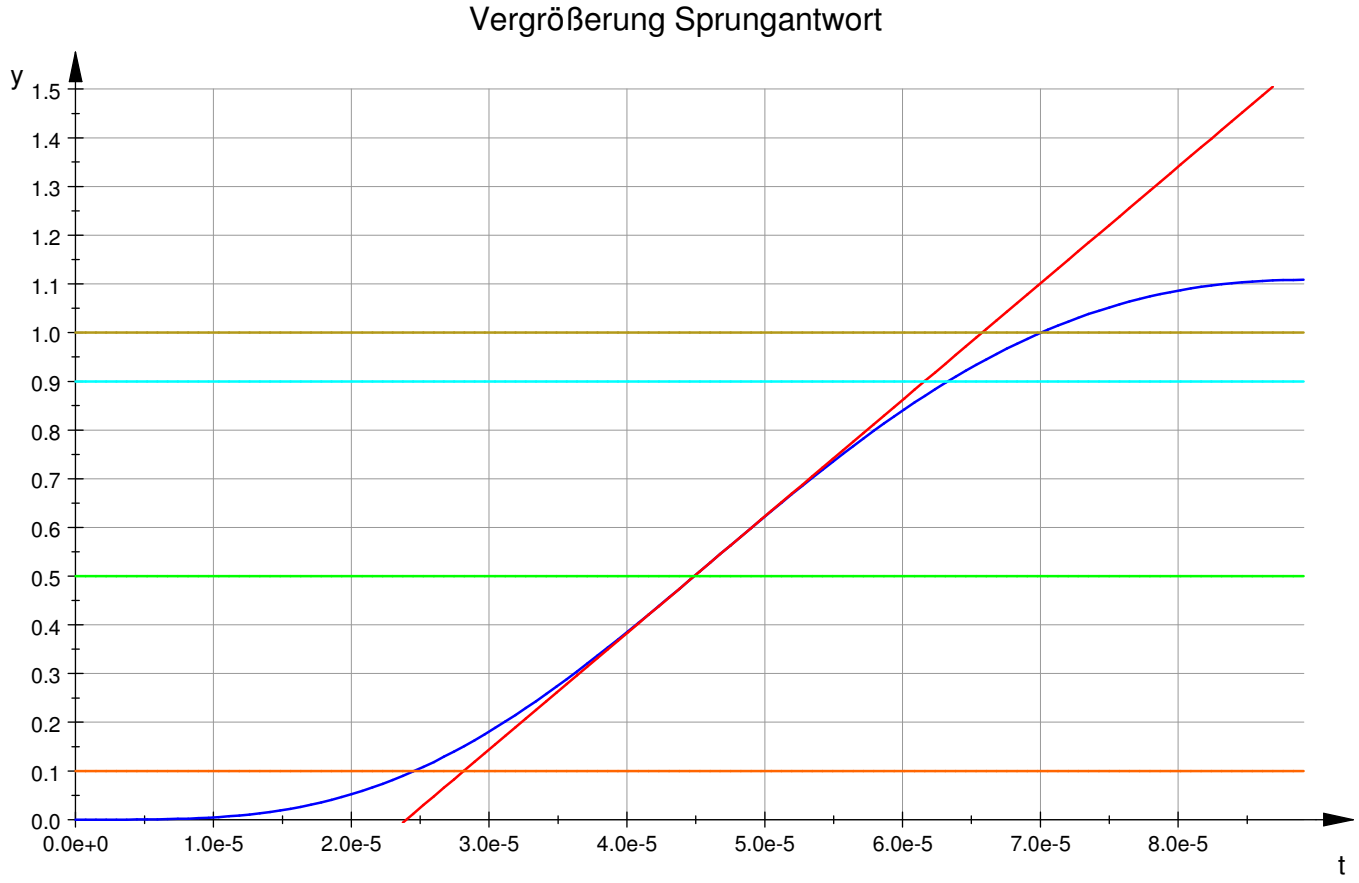
11.968397582586365529226696635187

tr, Rise-Time in us

- ```
  tr:=op(numeric::solve(ua(t)=9/10,t=anf..ende,RestrictedSearch),1)-
  op(numeric::solve(ua(t)=1/10,t=anf..ende,RestrictedSearch),1):tr/1e-
  6;
  ```

  38.712994723645717725617057861113

- ```
  plotfunc2d(ua(t), yt(t), 1/2, 1, 1/10, 9/10, t=0..maximum,
  LegendVisible=FALSE, CoordinateType=LinLin,
              GridVisible=TRUE, SubgridVisible=FALSE,
    Height=120*unit::mm, Width=180*unit::mm, Header="Vergrößerung
    Sprungantwort", YRange=0..1.5):
  ```




CPU-Zeit in Sekunden und in Minuten

- ```
  te:=time():float((te-ta)/1000);float((te-ta)/1000/60);
  ```

  2.813
  ```

0.046883333333333333333333333333333

- 