

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

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

die Eingangsdaten

- `RippledB:=3:n:=4:fg:=10e3:ue2:=1:`

die Berechnungen

- `epsilon:=sqrt(10^(0.1*RippledB)-1):wg:=2*PI*fg:`
- `gam:=1/n*arcsinh(1/epsilon):`
- `if frac(n/2)=0 then`
  - `k:=n/2:`
  - `b:=[1/(cosh(gam)^2-cos((2*i-1)*PI/2/n)^2) $ i=1..k]:`
  - `a:=[2*b[i]*sinh(gam)*cos((2*i-1)*PI/2/n) $ i=1..k]:`
- `else`
  - `k:=(n+1)/2:`
  - `b:=[0]:a:=[1/sinh(gam)]:`
  - `Liste:=[1/(cosh(gam)^2-cos((i-1)*PI/n)^2) $`
  - `i=2..k]:b:=b.Liste:delete Liste:`
  - `Liste:=[2*b[i]*sinh(gam)*cos((i-1)*PI/n) $`
  - `i=2..k]:a:=a.Liste:delete Liste:`
- `end_if:`

die Koeffizienten für quadratische Glieder ( $T(p) = A0 / [\text{product}(1 + a_i * p/wg + b_i * (p/wg)^2)]$ ) 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;`

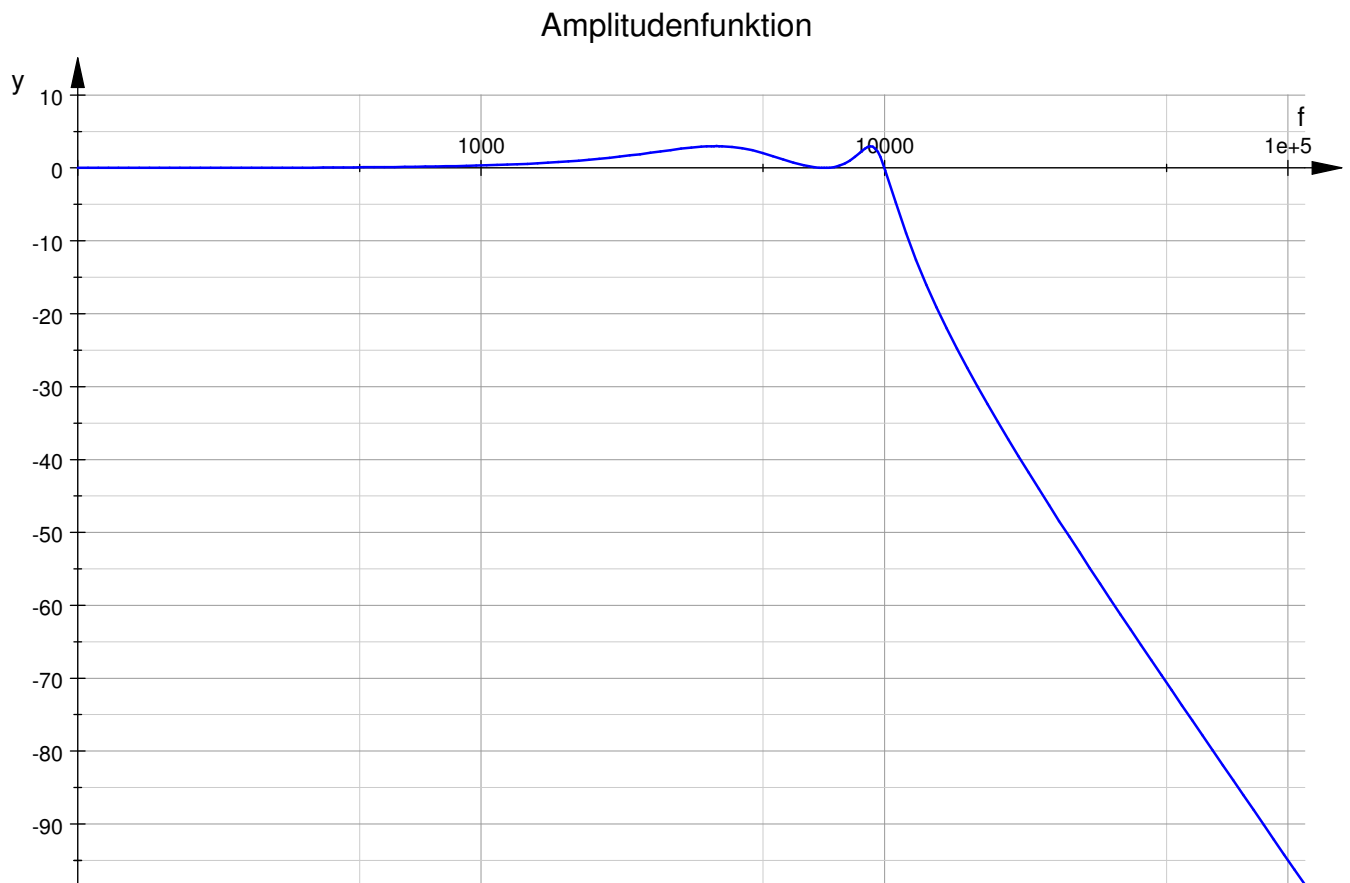
$$\begin{bmatrix} 1 & = & 2.0983726255243650369620324477108 \\ 2 & = & 0.1886206299189319279551248780539 \end{bmatrix}$$

$$\begin{bmatrix} 1 & = & 5.1025615414320170097526063859613 \\ 2 & = & 1.1073132983403206011219777999711 \end{bmatrix}$$

- `delete i:prod:=(f)->product(1+a[i]*I*w/wg+b[i]*(I*w/wg)^2, i=1..k):`
- `f:=0:b0:=float(expand(prod(f))):a0:=b0:`
- `U2U0:=(f)->a0/(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

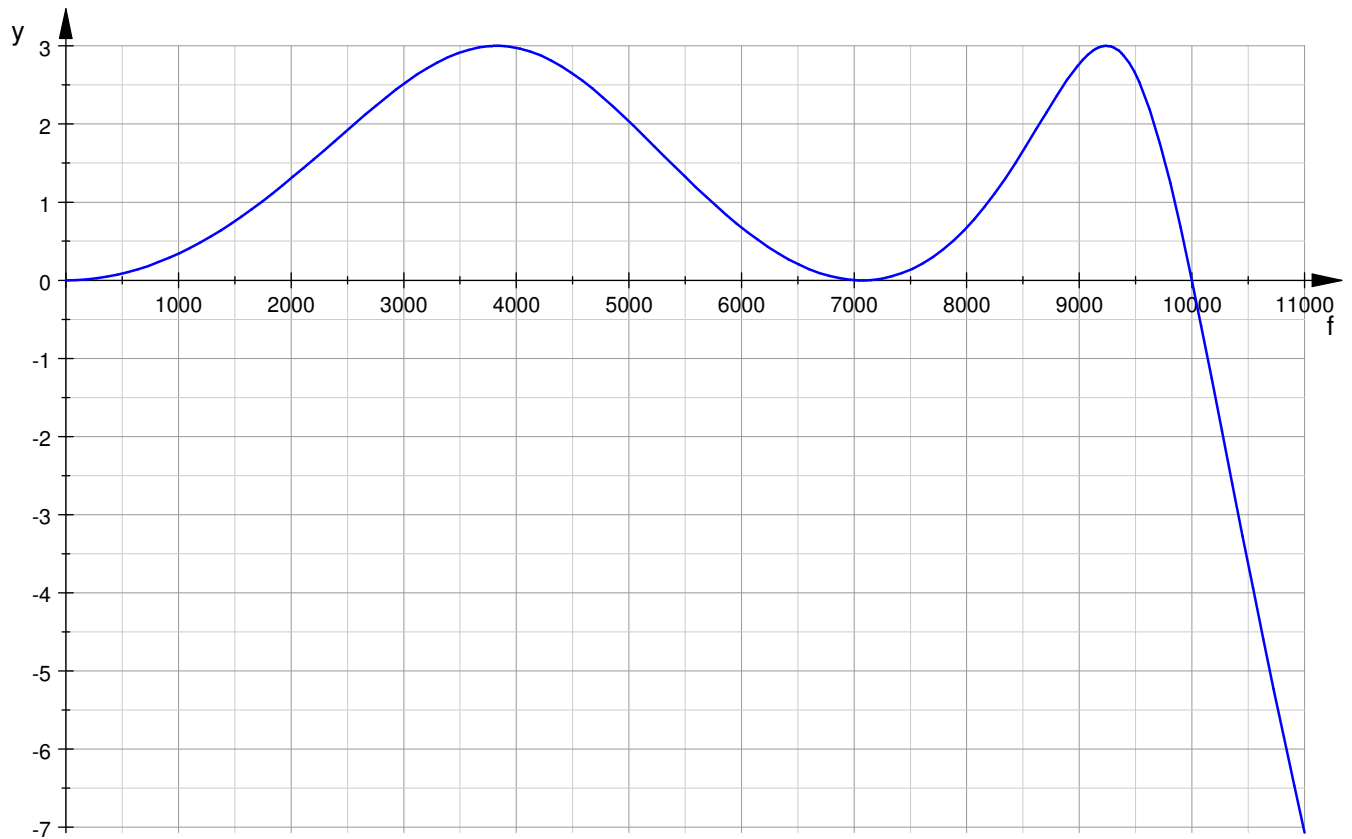
- `delete f:plotfunc2d(U2U0dB(f)+6.02, f=100..11*fg,`
  - `LegendVisible=FALSE, CoordinateType=LogLin,`
  - `GridVisible=TRUE, SubgridVisible=TRUE,`
  - `Height=120*unit::mm, Width=180*unit::mm,`
  - `Header="Amplitudenfunktion", YMax=10):`



Ausschnittsvergrößerung aus dem Betrag der Übertragungsfunktion, einfach logarithmisch

- `plotfunc2d(U2U0dB(f)+6.02, f=0..1.1*fg, LegendVisible=FALSE, CoordinateType=LinLin, GridVisible=TRUE, SubgridVisible=TRUE, Height=120*unit::mm, Width=180*unit::mm, Header="Vergrößerung Amplitudenfunktion"):`

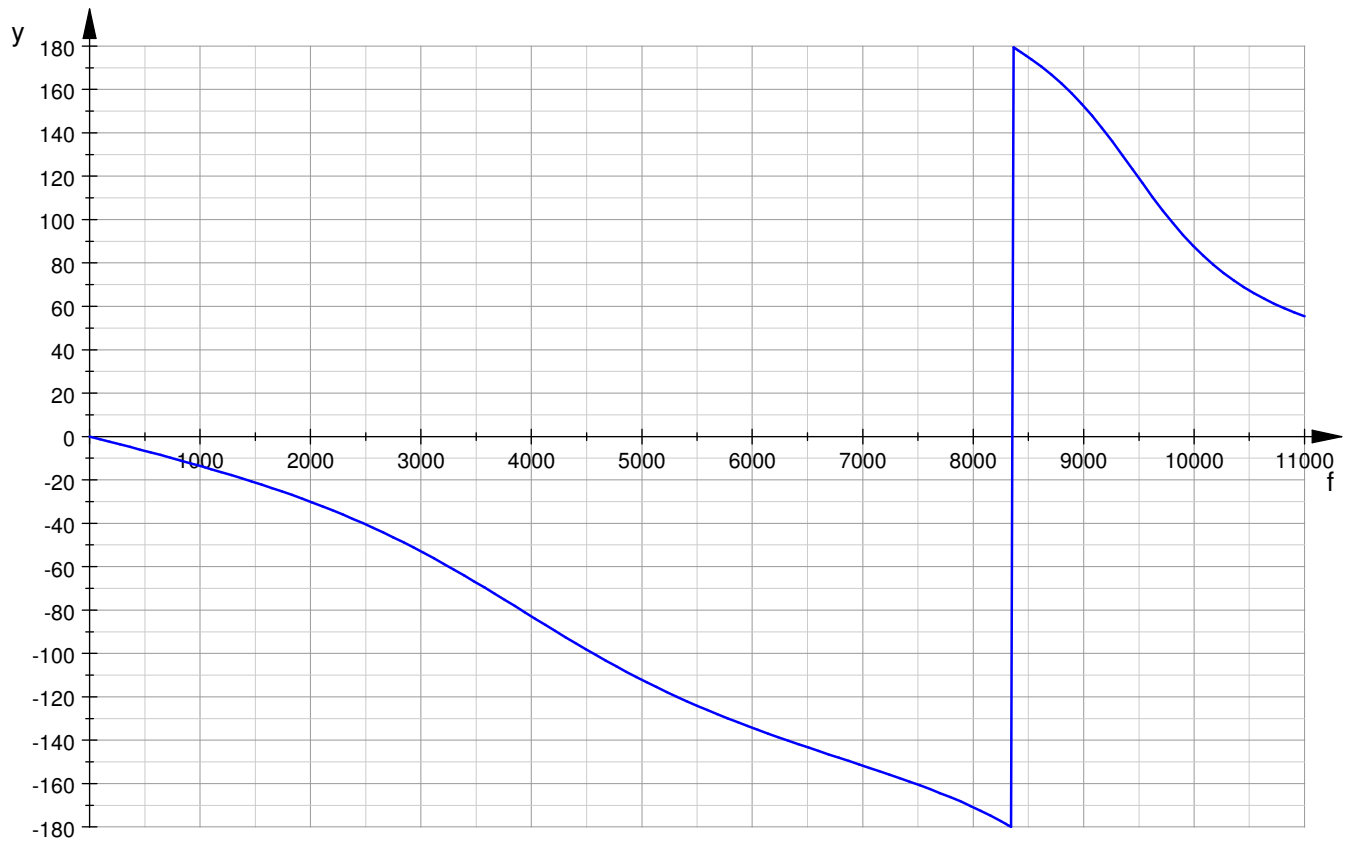
## Vergrößerung Amplitudenfunktion



### 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"):`

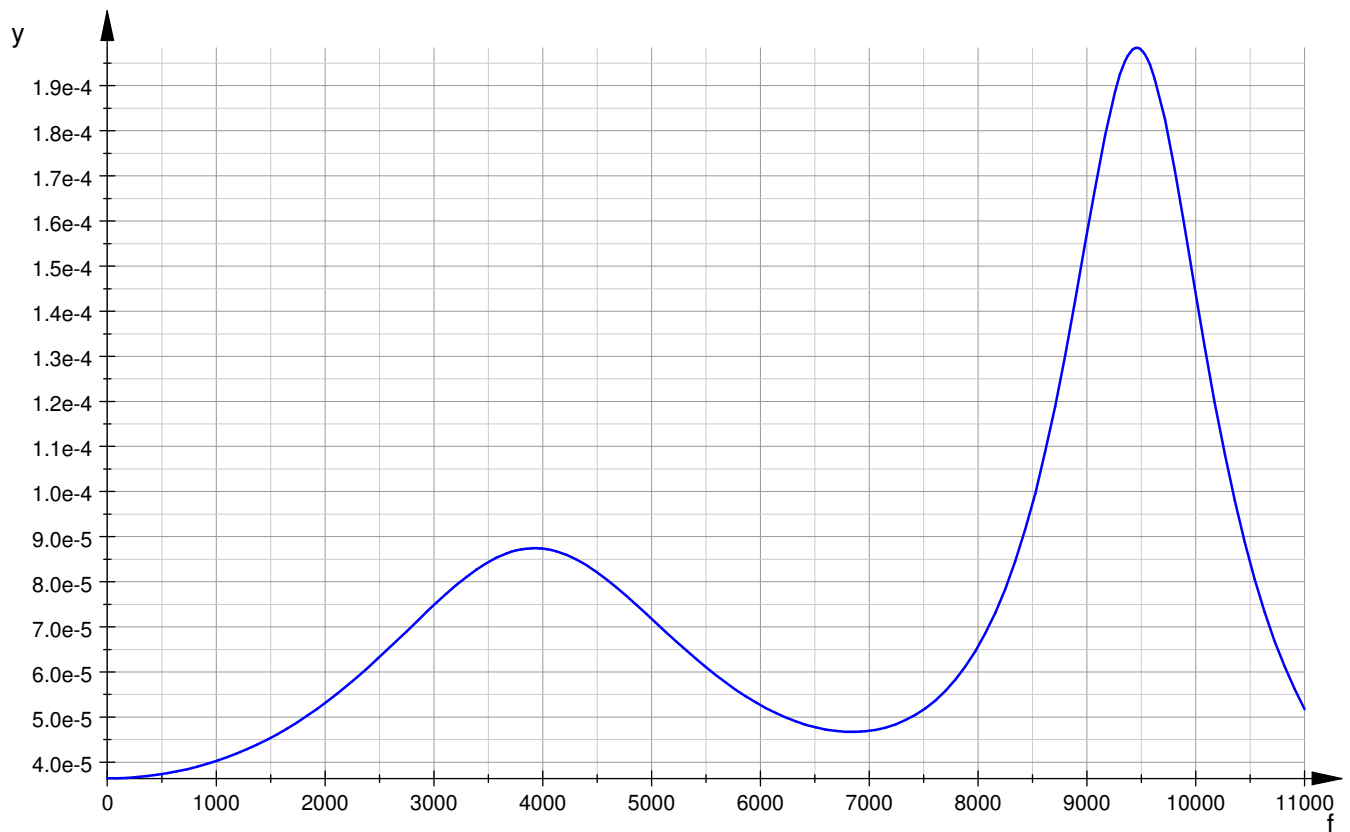
## Phasenfunktion



### 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"):`

## Gruppenlaufzeit

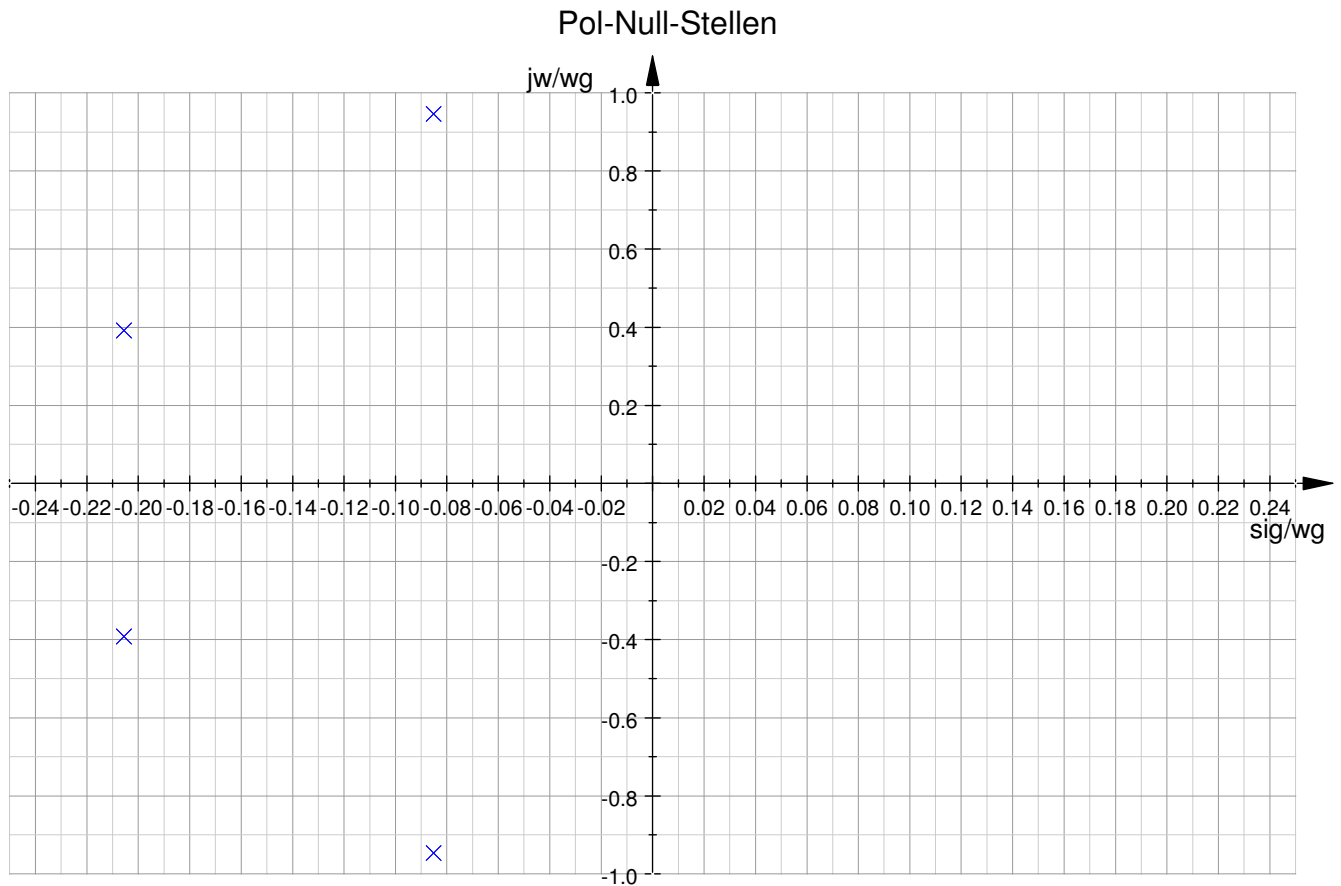


Lage der Pol-Nullstellen des Filters in der komplexen Ebene durch Suche der Nullstellen jedes Linearfaktors einzeln

- `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:=1/4>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"):`



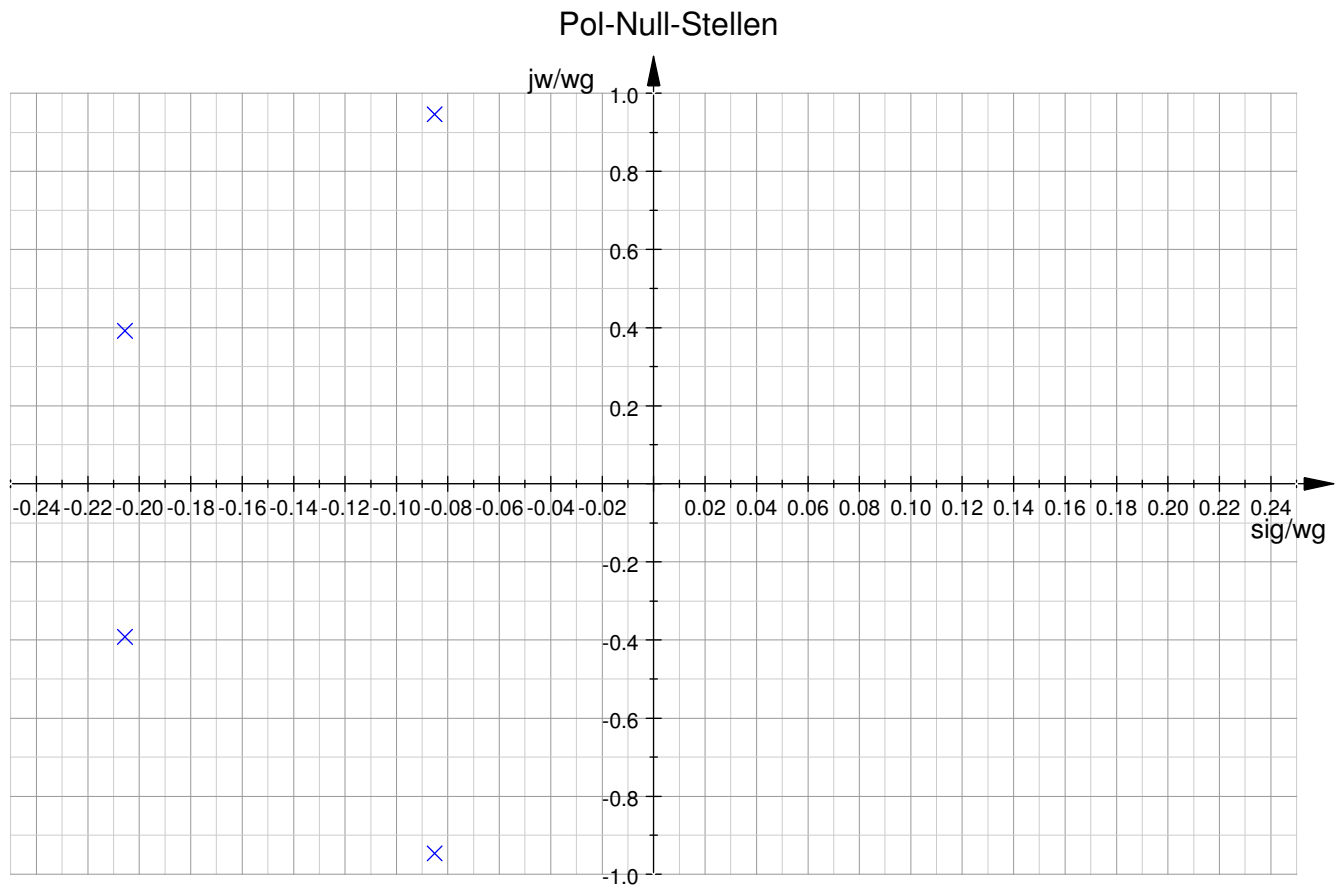
- PolTab;

$$\begin{cases}
 1 &= -0.20561953133596735651924188326189 - 0.39204668879992470427982858634734 \cdot i \\
 2 &= -0.20561953133596735651924188326189 + 0.39204668879992470427982858634734 \cdot i \\
 3 &= -0.085170398568157286537932206761801 - 0.94648443318424240414154999590935 \cdot i \\
 4 &= -0.085170398568157286537932206761801 + 0.94648443318424240414154999590935 \cdot i
 \end{cases}$$

direkte Berechnung der Pole nach HERPY/BERKA

- `sigiwg:=[-sin(PI/2/n*(2*i-1))*sinh(1/n*arcsinh(1/epsilon)) $ i=1..n]:wiwg:=[I*cos(PI/2/n*(2*i-1))*cosh(1/n*arcsinh(1/epsilon)) $ i=1..n]:`
- `delete Liste:for i from 1 to n do PolTabdirekt[i]:=float(op(sigiwg,i))+float(op(wiwg,i)): end_for:`
- `Liste:=[[Re(op(op(PolTabdirekt,i),2)),Im(op(op(PolTabdirekt,i),2)),RGB::Blue] $ i=1..n]:`
- `Breite:=1/4>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"):



- PolTabdirekt;

$$\begin{cases} 1 &= -0.085170398568157286537932206761802 + 0.94648443318424240414154999590935 \cdot i \\ 2 &= -0.20561953133596735651924188326189 + 0.39204668879992470427982858634734 \cdot i \\ 3 &= -0.20561953133596735651924188326189 - 0.39204668879992470427982858634734 \cdot i \\ 4 &= -0.085170398568157286537932206761802 - 0.94648443318424240414154999590935 \cdot i \end{cases}$$

die Grunddämpfung durch  $\ddot{u}^2$

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

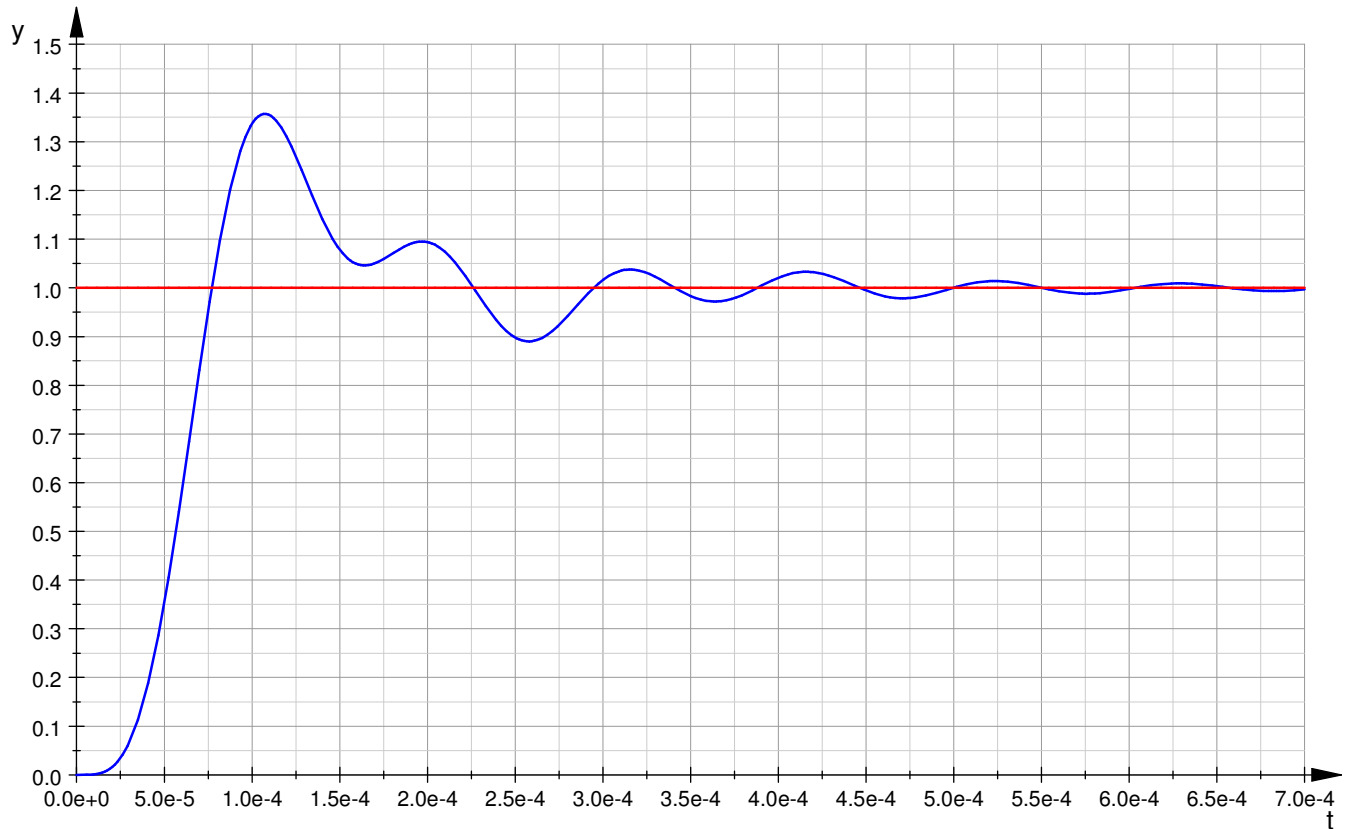
$$\frac{1}{2}$$

Sprungantwort des Filters  $ua(t)=\text{invlaplace}(2/p*T(p))$

- `delete i:prodp:=(p)->product(1+a[i]*p/wg+b[i]*(p/wg)^2, i=1..k):`
- `ua:=(t)->Re(transform::invlaplace(a0/(1+ue2)*2/p/prodp(p),p,t)):`
- `plotfunc2d(ua(t), 1, t=0..7/fg, LegendVisible=FALSE, CoordinateType=LinLin, GridVisible=TRUE, SubgridVisible=TRUE, Height=120*unit::mm, Width=180*unit::mm, Header="Sprungantwort",`

YMax=1.5) :

### Sprungantwort



#### Suchbereich definieren

- `anf:=2e-5;ende:=1.6e-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;`

35.713963433777596308931745543065

107.31238061694148691177422219756

#### Ausschnittsvergrößerung der Sprungantwort

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;`

56.643187391253863538369041042746

#### 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;`

42.319836493190630946857350462614

11.814790448929339990533866998922

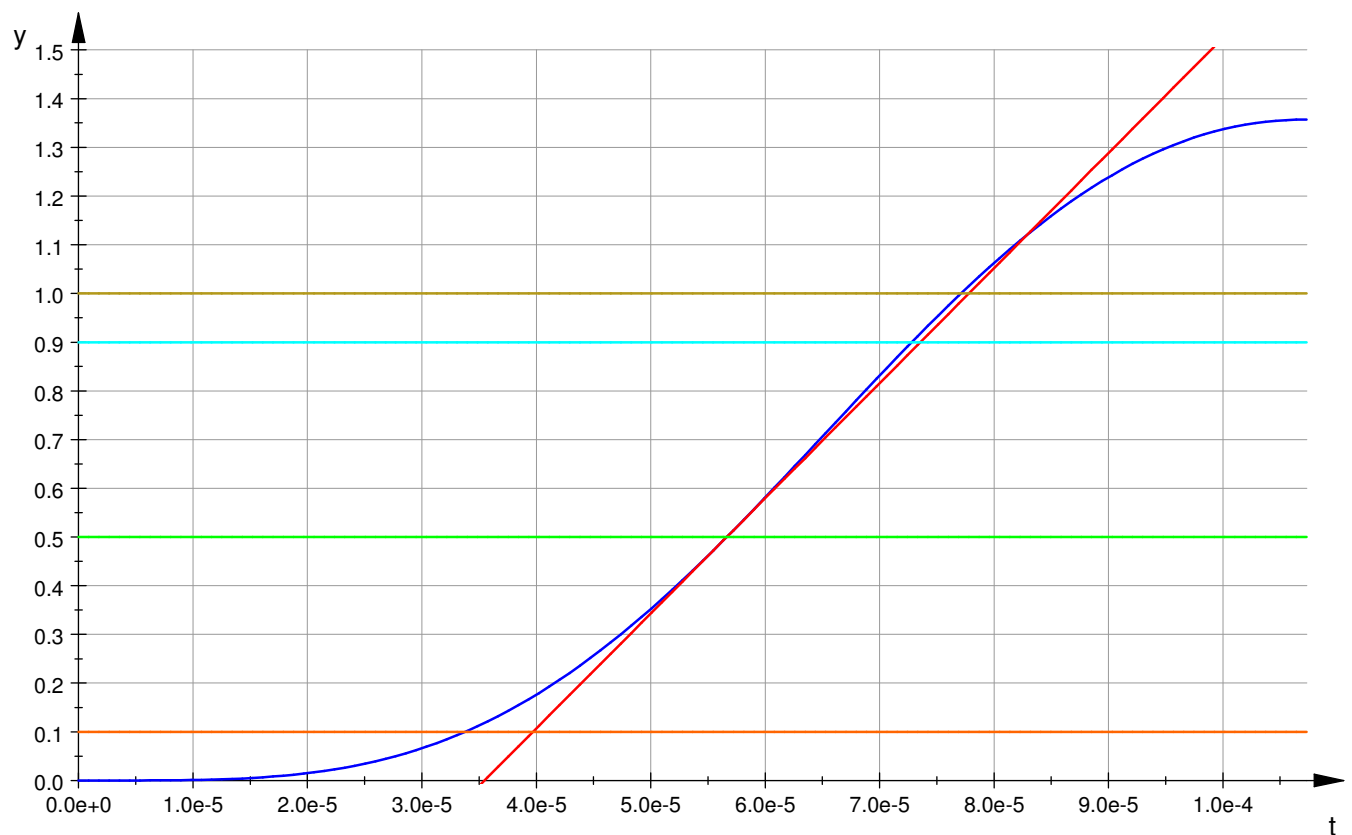
tr, Rise-Time in us

- `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;`

39.027781905057834075949376153593

- `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):`

Vergrößerung Sprungantwort



CPU-Zeit in Sekunden und in Minuten

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

