

- `reset():ta:=time():DIGITS:=32:c0:=299792458:z0:=50:z1:=50:z2:=50:l:=100:x:=100:Cs:=101.049872e-12:Rs:=6.56167979e-3:`

Prozeduren: Numerische, inverse Laplace-Transformation (Talbot), für t nicht 0 eingeben

```
# Talbot suggested that the Bromwich line be deformed into a contour that begins
# and ends in the left half plane, i.e., z ? -8 at both ends.
# Due to the exponential factor the integrand decays rapidly
# on such a contour. In such situations the trapezoidal rule converge
# extraordinarily rapidly.
# For example here we compute the inverse transform of  $F(s) = 1/(s+1)$  at  $t = 1$ 
#-----
#Octave:
#>> pkg load symbolic
#>> syms s
#>> F=1/(s+1)
#F = (sym)
#
# 1
# ----
# s + 1
#
#>> error=talbot(function_handle(F),1,24)-exp(-1)
#ans = 1.6098e-015
#-----
#
# Talbot method is very powerful here we see an error of 1.61e-015
# with only 24 function evaluations
#
# Created by Fernando Damian Nieuwveldt
# email:fdnieuwveldt@gmail.com
# Date : 25 October 2009
#
# Reference
# L.N.Trefethen, J.A.C.Weideman, and T.Schmelzer. Talbot quadratures
# and rational approximations. BIT. Numerical Mathematics,
# 46(3):653 670, 2006.
#
# Shift contour to the right in case there is a pole on the positive real axis : Note the contour will
# not be optimal since it was originally developed for function with
# singularities on the negative real axis
# For example take  $F(s) = 1/(s-1)$ , it has a pole at  $s = 1$ , the contour needs to be shifted with one
# unit, i.e shift = 1. But in the test example no shifting is necessary
```

- `Talbot:=proc(F_s, t, N)
local h, shift, ans, theta, k, z, dz;
begin
h:=2*PI/N;
shift:=0;
ans:=0;
for k from 0 to N do
theta:=-PI+(k+1/2)*h;`

```

z:=shift+N/t*(0.5017*theta*cot(0.6407*theta)-
0.6122+0.2645*I*theta);
dz:=N/t*(-
0.5017*0.6407*theta/sin(0.6407*theta)^2+0.5017*cot(0.6407*theta)+0
.2645*I);
ans:=ans+exp(z*t)*F_s(z)*dz;
end_for:
return (Re(h/(2*I*PI)*ans));
end_proc:

```

Induktivitätsbelag in uH/m

- `Ls:=Z0^2*Cs:float(Ls/1e-6);`
0.25262468

Ableitungsbelag in uS/m

- `Gs:=150*Rs*Cs/Ls:float(Gs/1e-6);`
393.7007874

Ausbreitungsgeschwindigkeit auf der Leitung in m/s

- `v1:=1/sqrt(Ls*Cs);`
197922071.58857163124362987812592

Verhältnis Ausbreitungsgeschw. / Lichtgeschw.

- `v1/c0;`
0.66019696729185772659974614213251

Laufzeit für x Meter in us (2 Methoden)

- `td:=x/v1:float(td/1e-6),float(x*sqrt(Ls*Cs)/1e-6);`
0.50524936, 0.50524936

Übertragungsfunktion der Leitung

- `gam:=sqrt((Rs+p*Ls)*(Gs+p*Cs));`
 - `/* Tp:=(Z2*cosh(gam*(1-x))+Z0*sinh(gam*(1-x)))/((Z1+Z2)*cosh(gam*1)+(Z0+Z1*Z2/Z0)*sinh(gam*1)); */`
- sinh() u. cosh() umformen in e-Funktionen
- `a:=gam*(1-x):b:=gam*1;`
 - `Tp1:=(Z2*(exp(a)+exp(-a))+Z0*(exp(a)-exp(-a)))/((Z1+Z2)*(exp(b)+exp(-b))+(Z0+Z1*Z2/Z0)*(exp(b)-exp(-b)));`

Erregung Sprungfunktion 1/p

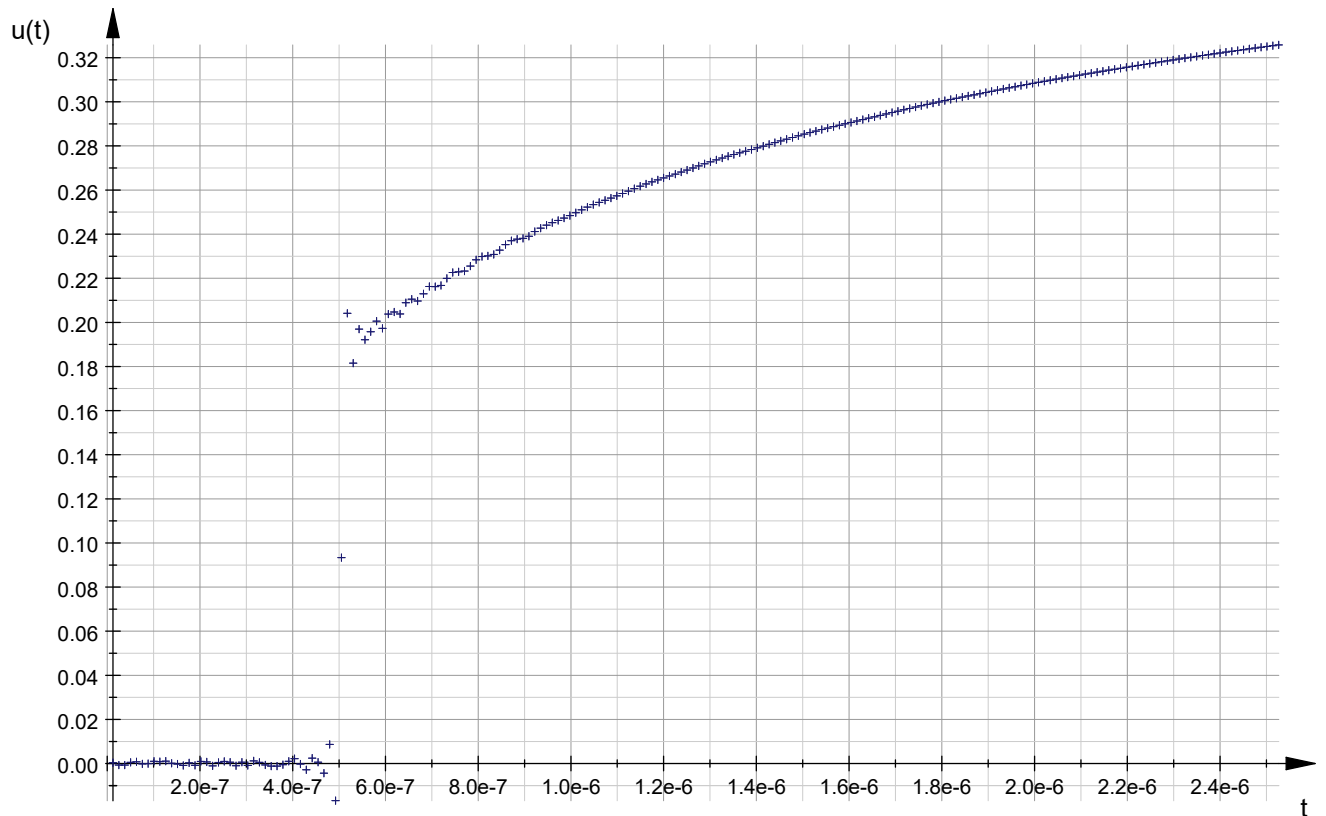
- `lap:=(p)-->expand(1/p*Tp1);`

$$p \rightarrow \frac{1}{2 \cdot p \cdot \left(e^{\sqrt{0.00000000000000002552769157804096 \cdot p^2 + 0.00000000010012159233555751888 \cdot p + 0.000002583338499989666646}} \right)^{100}}$$

Anzahl der Stützstellen u. Talbot-Iterationen

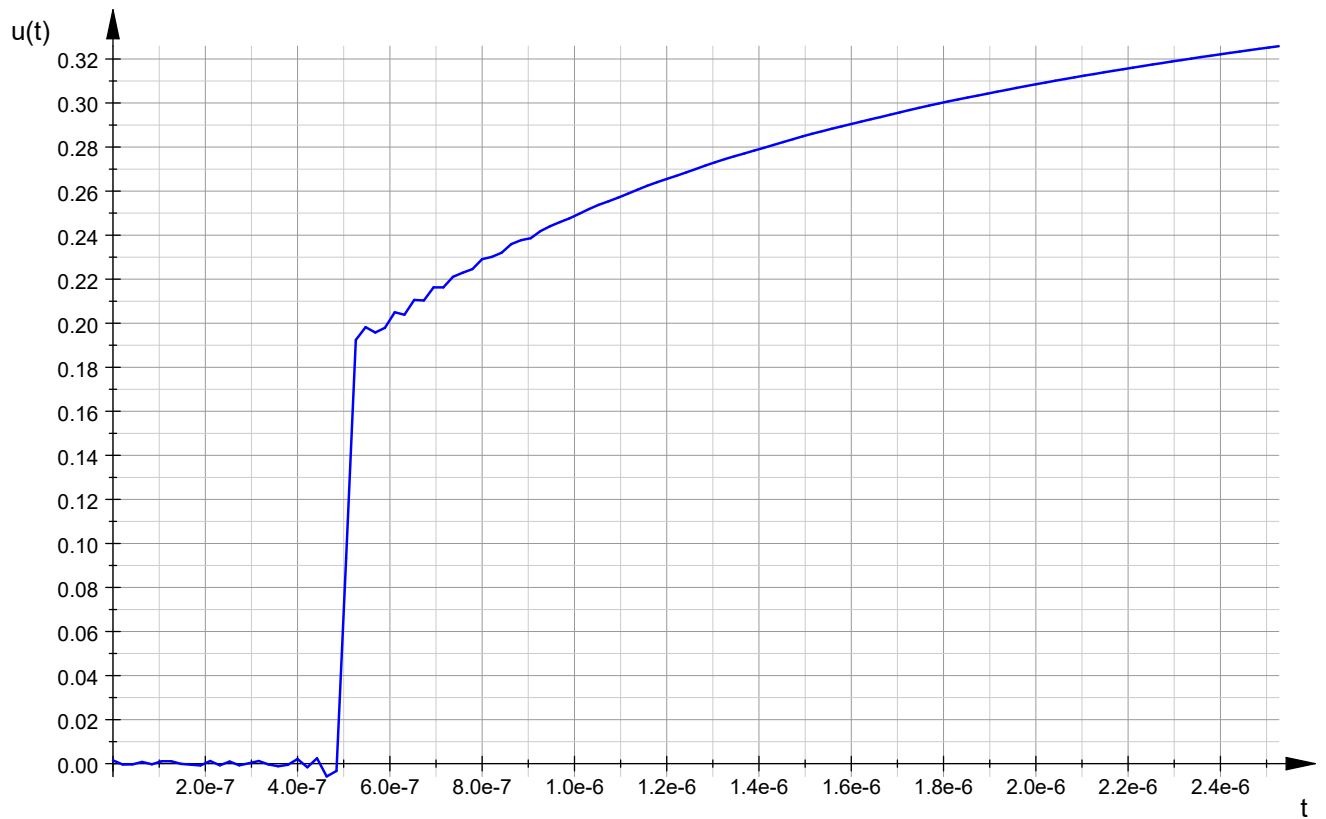
- `M:=200:Talits:=384:Zeit:=5*td:`
- `Liste:=[[float(Zeit/M*i), float(Talbot(lap,Zeit/M*i,Talits))] $ i=1..M]:`
- `plot(plot::PointList2d(Liste, PointStyle=Crosses, PointSize=1, GridVisible=TRUE, SubgridVisible=TRUE, Scaling=Unconstrained, AxesTitles=["t","u(t)"], Height=120*unit::mm, Width=180*unit::mm, Header="Sprungantwort der Leitung an der Stelle x")):`

Sprungantwort der Leitung an der Stelle x



- `x:=array(0..M-1, [float(Zeit/M*(i+1)) $ i=0..M-1]):`
- `S:=numeric::cubicSpline([x[i], float(Talbot(lap,x[i],Talits))] $ i=0..M-1, Natural):`
- `delete x:plot(plot::Function2d(S(x), x=0..5*td), GridVisible=TRUE, SubgridVisible=TRUE, Scaling=Unconstrained, AxesTitles=["t","u(t)"], Height=120*unit::mm, Width=180*unit::mm, Header="Spline der Sprungantwort der Leitung an der Stelle x")):`

Spline der Sprungantwort der Leitung an der Stelle x



Berechnung eines Funktionswertes

- $S(8e-7)$;

0.22908062402378941050314896529032

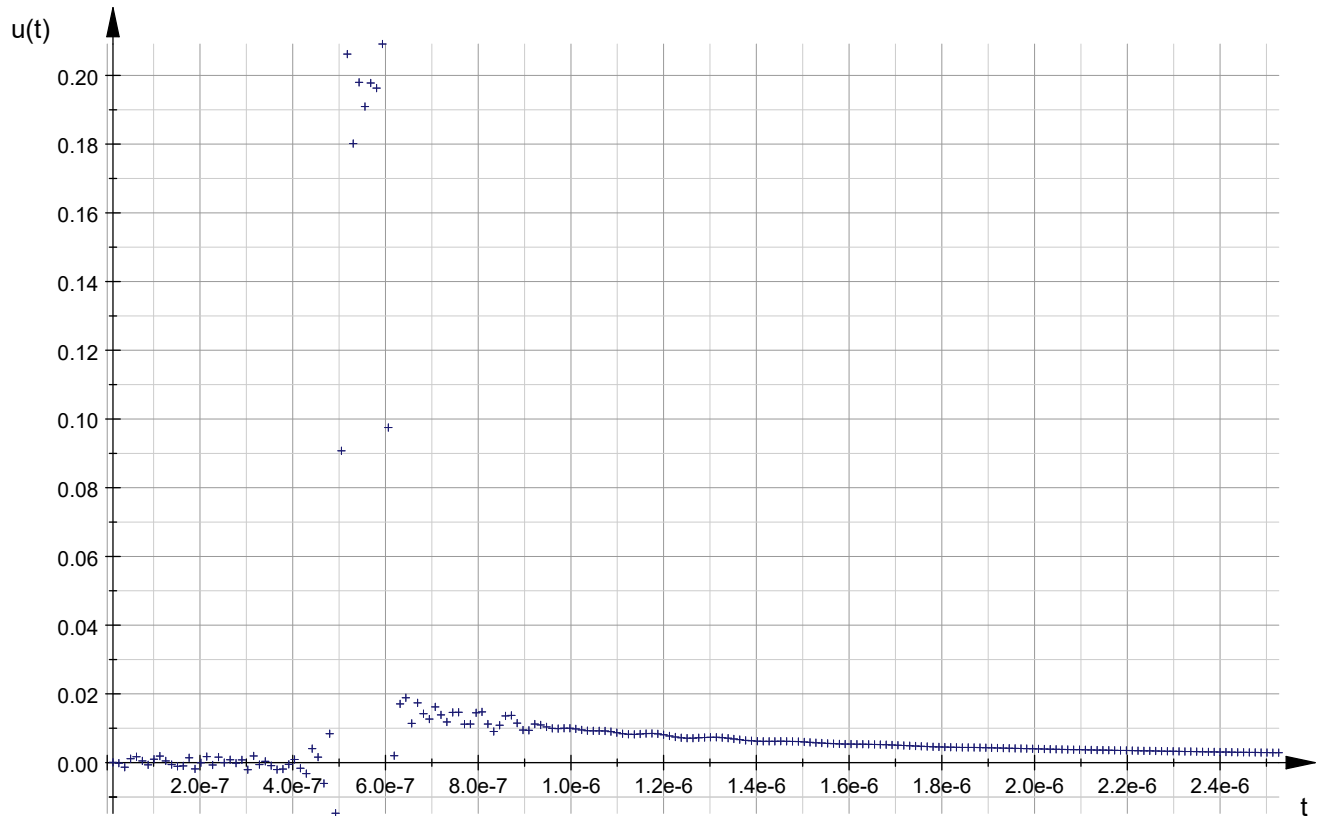
Erregung Rechteckimpuls 0.1 us

- $U_{g1} := 1/p * (1 - \exp(-1e-7 * p))$;
- $lap := (p) \rightarrow \text{expand}(U_{g1} * Tp1)$;

$$p \rightarrow \frac{1}{2 \cdot p \cdot \left(e^{\sqrt{0.00000000000000002552769157804096 \cdot p^2 + 0.00000000010012159233555751888 \cdot p + 0.000002583338499989666646}} \right)^{100}}$$

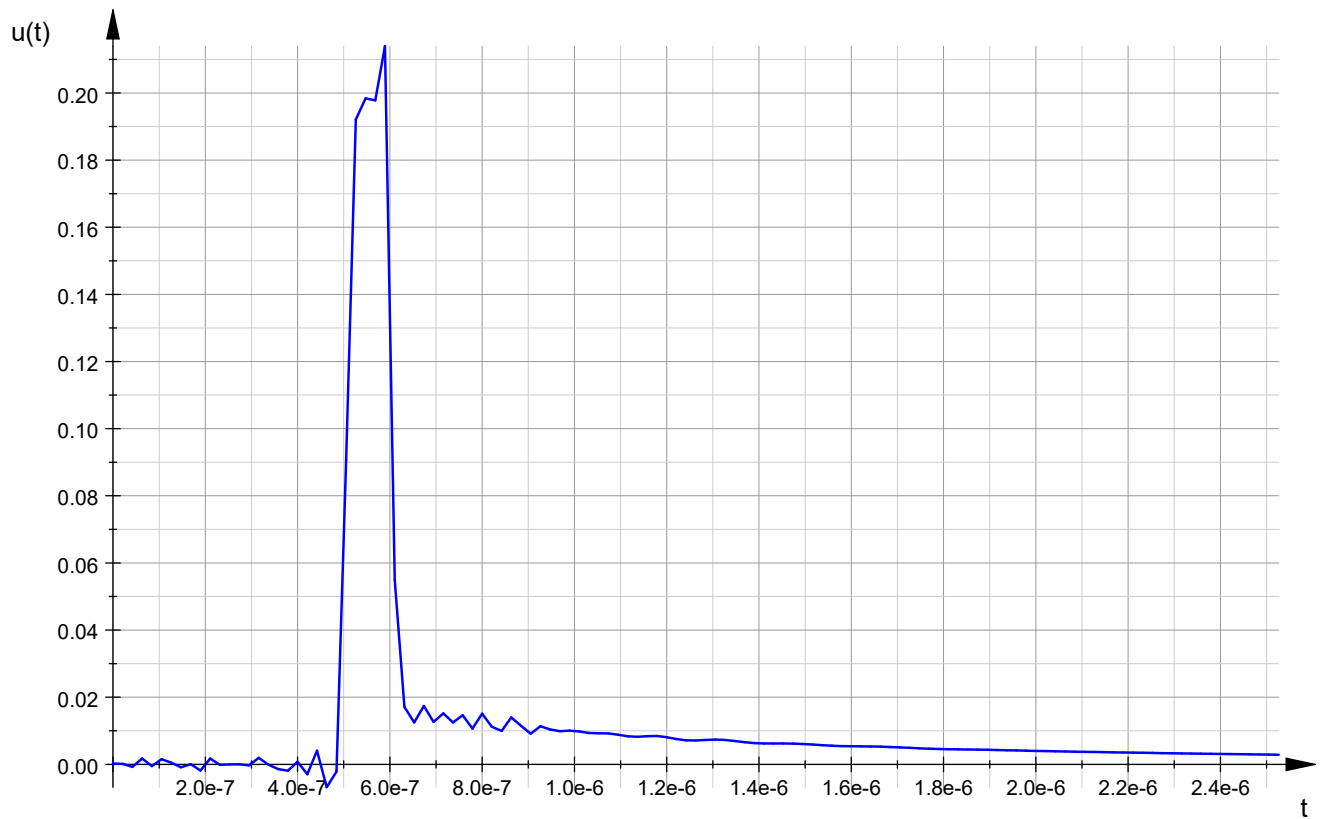
- `delete Liste:Liste:=[[float(5*td/M*i), float(Talbot(lap,5*td/M*i,Talits))] $ i=1..M]:`
- `plot(plot::PointList2d(Liste, PointStyle=Crosses, PointSize=1, GridVisible=TRUE, SubgridVisible=TRUE, Scaling=Unconstrained, AxesTitles=["t","u(t)"], Height=120*unit::mm, Width=180*unit::mm, Header="Rechteckimpulsantwort der Leitung an der Stelle x")):`

Rechteckimpulsantwort der Leitung an der Stelle x



- `x:=array(0..M-1, [float(Zeit/M*(i+1)) $ i=0..M-1]):`
- `S:=numeric::cubicSpline([x[i], float(Talbot(lap,x[i],Talits))] $ i=0..M-1, Natural):`
- `delete x:plot(plot::Function2d(S(x), x=0..5*td), GridVisible=TRUE, SubgridVisible=TRUE, Scaling=Unconstrained, AxesTitles=["t","u(t)"], Height=120*unit::mm, Width=180*unit::mm, Header="Spline der Rechteckimpulsantwort der Leitung an der Stelle x"):`

Spline der Rechteckimpulsantwort der Leitung an der Stelle x



Berechnung eines Funktionswertes

- `S(8e-7);`

0.01512283602983722583639223256606

CPU-Zeit in Sekunden und Minuten

- `float((time()-ta)/1e3);float((time()-ta)/1e3/60);`

4901.766

81.6961

-