----- Ingenieurbüro Baumann ----- 30. Mai 2007

Synthese-Verfahren, Beispiel
Erweiterung eines JACOBI-TP 4. Ordnung durch 2 konj. komplexe Polstellen zu einem CAUER-TP math. 6.Grades

Ergebnis: CAUER-TP, der maximal flach ist im Durchlassbereich

man wählt die neue Konstante Theta mit Epsilon/Theta = sqrt(2)...sqrt(3)
als erste konj. komplexe Polstelle des Filters wird aus der Gleichung 1 + (epsilon / theta / (F - Pol1))^2 = 0 an der Stelle F=1 nach Pol1
die zweite konj. komplexe Polstelle wird bestimmt, indem die Gleichung 1 + (epsilon / theta / (F - Pol1) / (F- x0))^2 = 0 an der Stelle F=1 nach x0 gelöst wird
u.s.w.

die Polstellen des JACOBI-TP werden modifiziert und bestimmt durch Lösung der Gleichung `1+(epsilonT*KT(F)/(F-Pol1)/(F-Pol2))^2=0 nach F`

KT(F) ist hier die char. Gleichung des JACOBI-Filters nämlich cos(n * arccos(F))
FT(F) ist hier der Betrag der Übertragungsfunktion nämlich 1 / sqrt(1 + (epsilon * KT(F))^2)
F ist hier die normierte Frequenz f/fg

```
reset():DIGITS:=32:
n:=4:RippledB:=6.02:epsilonT:=sqrt(10^(0.1*RippledB)-
1):theta:=float(epsilonT/sqrt(2)):print("Epsilon =
",epsilonT):print("Theta = ",theta):
```

```
"Epsilon = ", 1.731891306523297611197070793085 8

 "Theta = ", 1.224632087120653311121815368810 2
```

JACOBI-Funktion mit Frequenzkorrektur

```
KT:=(F)->orthpoly::jacobi(n,1,1,F):alpha:=op(numeric::solve(1/(1+(epsilonT*KT(F))^2)=1,F=7/10..2,RestrictedSearch),1);
```

0.765055323929464692851002980408

```
F:=1:Pol:=numeric::solve(1+(epsilonT/theta/(F-x0))^2=0,x0=1..10,RestrictedSearch):Pol1:=Im(op(Pol,1)):delete F:
print("Polstelle 1 = ",Pol1):
```

```
"Polstelle 1 = ", 1.414213562373095048801688724209 7
```

```
F:=1:Pol:=numeric::solve(1+(epsilonT/theta/(F-Pol1)/(F-x0))^2=0,x0=1..10,RestrictedSearch):Pol2:=Im(op(Pol,1)):delete F:
print("Polstelle 2 = ",Pol2):
```

```
"Polstelle 2 = ", 3.414213562373095048801688724209 7
```

```
FT:=(F)->1/sqrt(1+(epsilonT*KT(alpha*F)/(F-Pol1)/(F-Pol2))^2):if frac(n/2)=0 then a0T:=10^(RippledB/20) else a0T:=1
```

```
end_if:a0:=1/limit(FT(F),F=0,Right):
```

- `FTN:=(F)->a0*FT(F):`
- `FTNdB:=(F)->20*log(10,FTN(F)):`
-
```
print("das a0 = ",float(round(a0*1e5)/1e5)," das entspricht [dB]=
",float(round(20*log(10,a0)*100)/100)):
print("die min. Filtersteilheit [dB/Dekade]=
   ",float(round(FTNdB(10)-FTNdB(1))*100)/100):
print("bei F=0 ergibt sich [dB] ",float(round(FTNdB(0)*100)/100)):
print("bei F=1 ergibt sich [dB] ",float(round(FTNdB(1)*100)/100)):
```

```
"das a0 = ", 1.02482, " das entspricht [dB]= ", 0.21

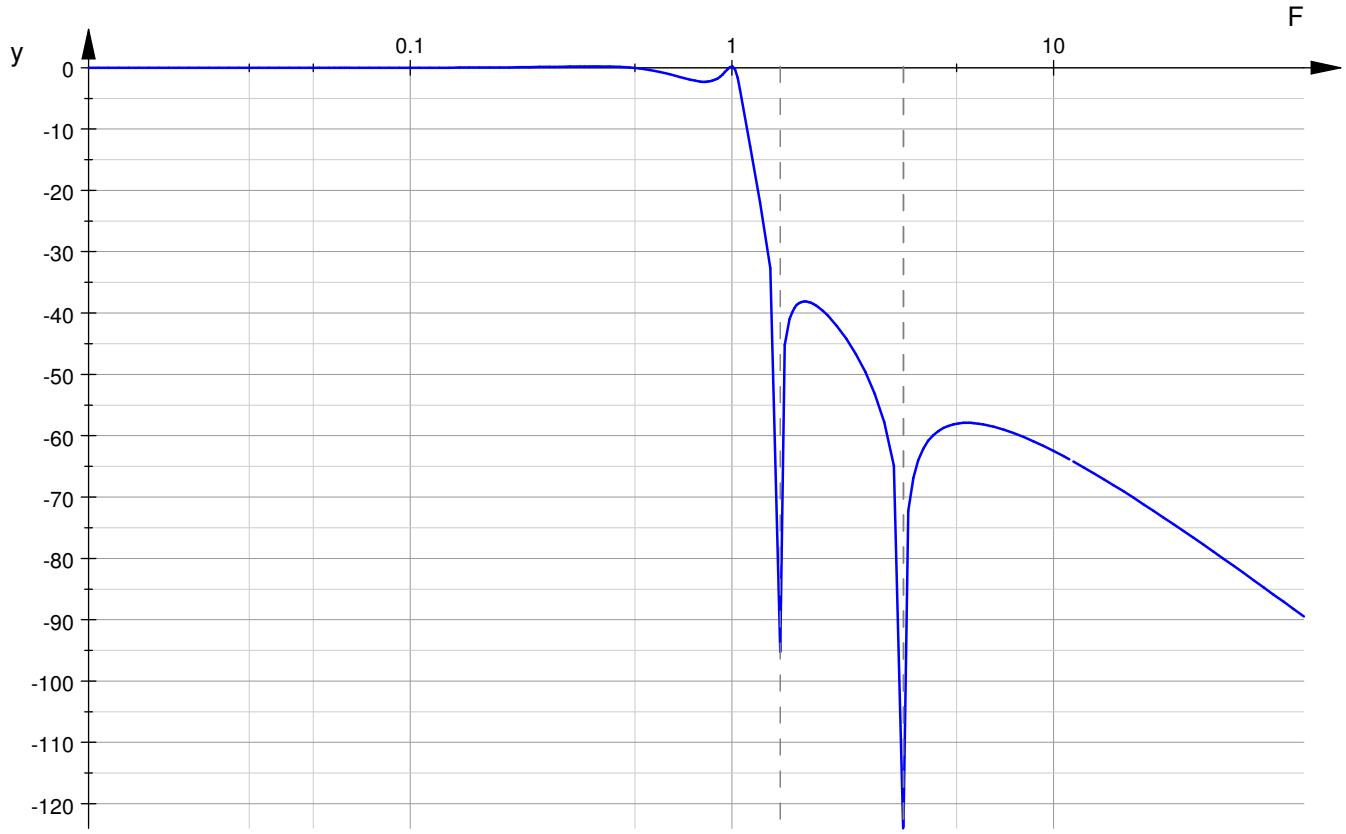
"die min. Filtersteilheit [dB/Dekade]= ", -63.0

"bei F=0 ergibt sich [dB] ", 0.0

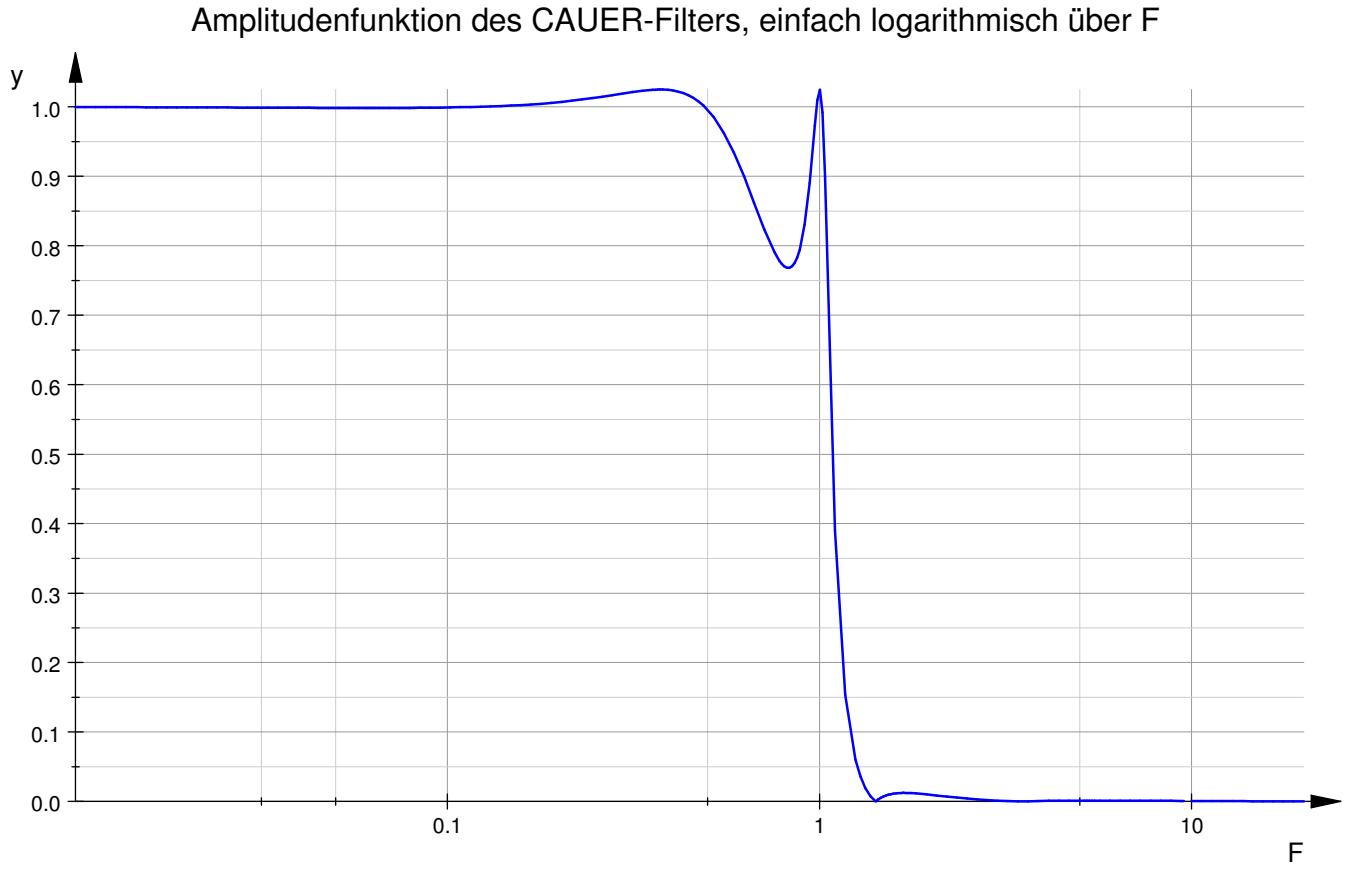
"bei F=1 ergibt sich [dB] ", 0.21
```

-
```
plotfunc2d(FTNdB(F), F=1/100..60, LegendVisible=FALSE,
CoordinateType=LogLin,
          GridVisible=TRUE, SubgridVisible=TRUE,
  Height=120*unit::mm, Width=180*unit::mm,
  Header="Amplitudenfunktion des CAUER-Filters math. 6. Grades,
  doppelt logarithmisch"):
```

Amplitudenfunktion des CAUER-Filters math. 6. Grades, doppelt logarithmisch



- ```
  plotfunc2d(FTN(F), F=1/100..20, LegendVisible=FALSE,
  CoordinateType=LogLin,
              GridVisible=TRUE, SubgridVisible=TRUE,
    Height=120*unit::mm, Width=180*unit::mm,
              Header="Amplitudenfunktion des CAUER-Filters, einfach
    logarithmisch über F"):
  ```




- ```
  print("die JACOBI-Polstellen werden modifiziert"):
  ```

```
                "die JACOBI-Polstellen werden modifiziert"
```

- ```
  ListeT:=[-Im(op(float(solve(1+(epsilonT*KT(alpha*F)/(F-Pol1)/(F-
  Pol2))^2=0,F)),i))-I*Re(op(float(solve(1+(epsilonT*KT(alpha*F)/(F-
  Pol1)/(F-Pol2))^2=0,F)),i)) $ i=1..n];
  ```

```
[

- 0.061899303814639500452757454633905 -
1.022776072460547682404740482999 5 \
I

   ,
```
```

```
0.06189930381463950045275745463390 6 -
1.02277607246054768240474048299955 I

   ,

- 0.35271703978795574812302492929695 -
0.54079428486016601597310924384455 \
I

   ,

0.35271703978795574812302492929695 -
0.54079428486016601597310924384455 I

   ]
```