Inverzní Čebyševova aproximace

oprava interní funkce MATLABu

Miloš Laipert, Miroslav Vlček, Jan Vrbata

ČVUT - fakulta elektrotechnická, katedra teorie obvodů
Technická 2, 160 00
laipert@feld.cvut.cz, vlcek@fd.cvut.cz, vrbata@hippo.feld.cvut.cz

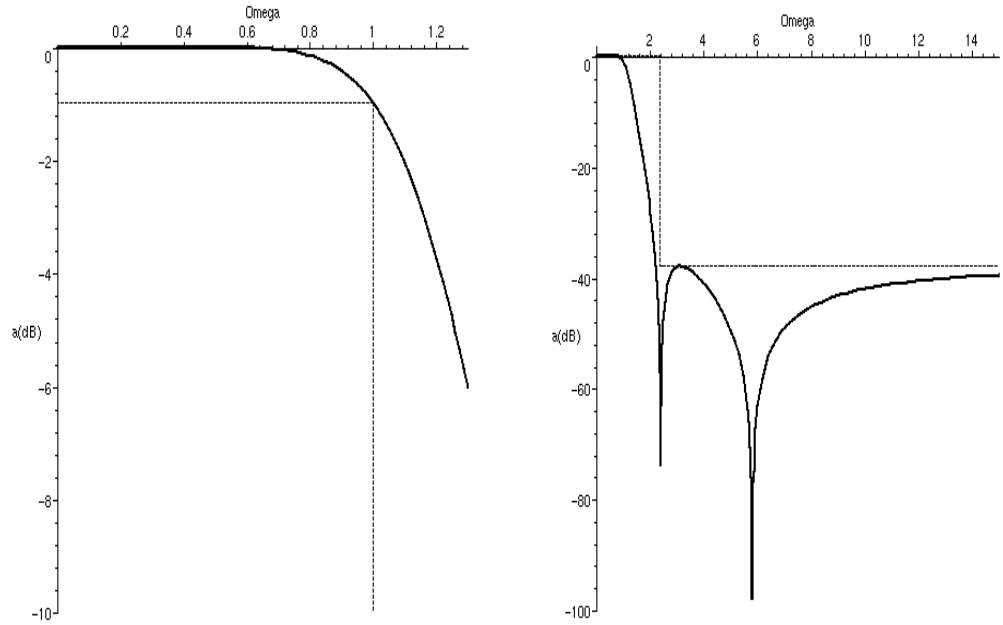
---

Jak bylo zjištěno, obsahuje funkce *cheb2ap*, která realizuje inverzní Čebyševovu aproximaci analogového prototypu normované dolní propusti v MATLABu, závažnou chybu. Tato chyba se týká špatného odnormování přenosové funkce. V příkladech na druhé straně tohoto textu je toto názorně demonstrováno na příkladech. Z důvodu používání MATLABu ve výuce lineárních obvodů na naší katedře, jsme se rozhodli uvedenou funkci přepracovat. Námi upravená funkce byla v průběhu semestru otestována při výuce, výsledky byly kontrolovány na mnoha příkladech.
Jelikož se uvedená chyba vyskytuje už v několika verzích MATLABu, rozhodli jsme se nabídnout Vám opravenou funkci s možností zařazení do systému MATLAB. Formát vstupních a výstupních parametrů opravené funkce odpovídá obdobným funkcím MATLABu, které realizují aproximace analogového prototypu. Nová přepracovaná funkce má jméno *invcheby*.

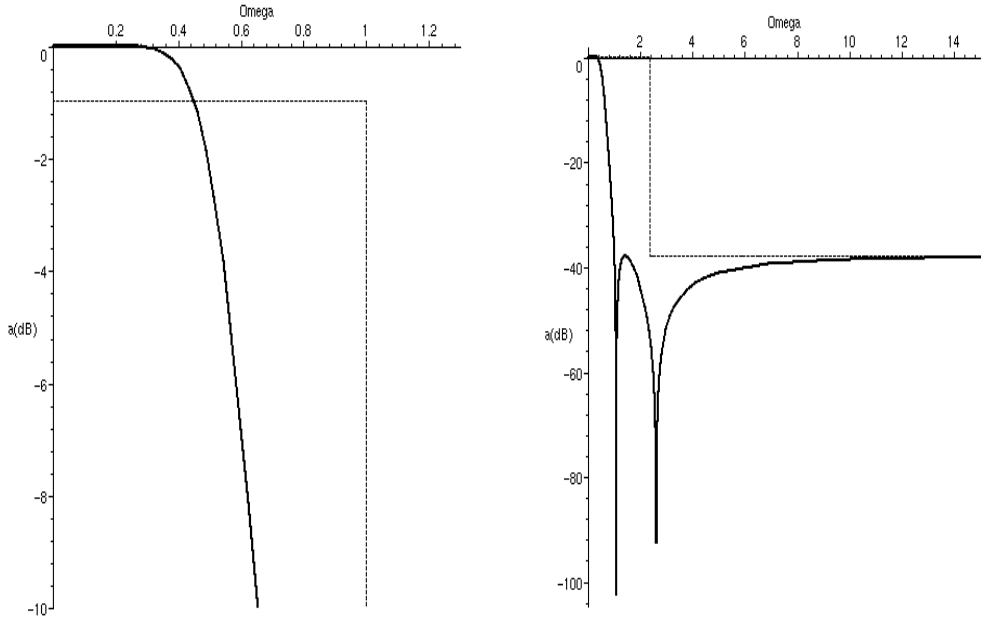
Aproximační funkce pro návrh analogových prototypů obsažené v MATLABu používají nepříliš vhodnou strukturu vstupních údajů tj. stupeň přenosové funkce (značeno N) a hodnotu útlumu v nepropustném pásmu (značeno R_s), případně i hodnotu útlumu v propustném pásmu (značeno R_p).
Návrh analogového filtru však nejčastěji vychází z tolerančního schématu a z něj se teprve určuje stupeň přenosové funkce filtru a ostatní sekundární parametry. Proto jsme vytvořili funkci pro výpočet inverzní Čebyševovy aproximace, jejímž vstupem jsou právě hodnoty tolerančního schématu. Funkce se jmenuje *invcheby2*. V případě zájmu, bychom mohli vytvořit i ostatní funkce, realizující aproximační úlohy analogového prototypu, tak aby vycházely z tolerančního schématu normované dolní propusti (Butterworthova, Čebyševova, Cauerova aproximace).

Na následujících dvou stranách jsou uvedeny dva názorné příklady pro lichý a sudý stupeň přenosové funkce. Graficky je znázorněn průběh přenosové funkce v decibelech pro původní a opravenou funkci inverzní Čebyševovy aproximace.

Příklad : Normovaná dolní propust, a_p=-1 dB, a_s=-38 dB, Ω_s=2.35, n=4

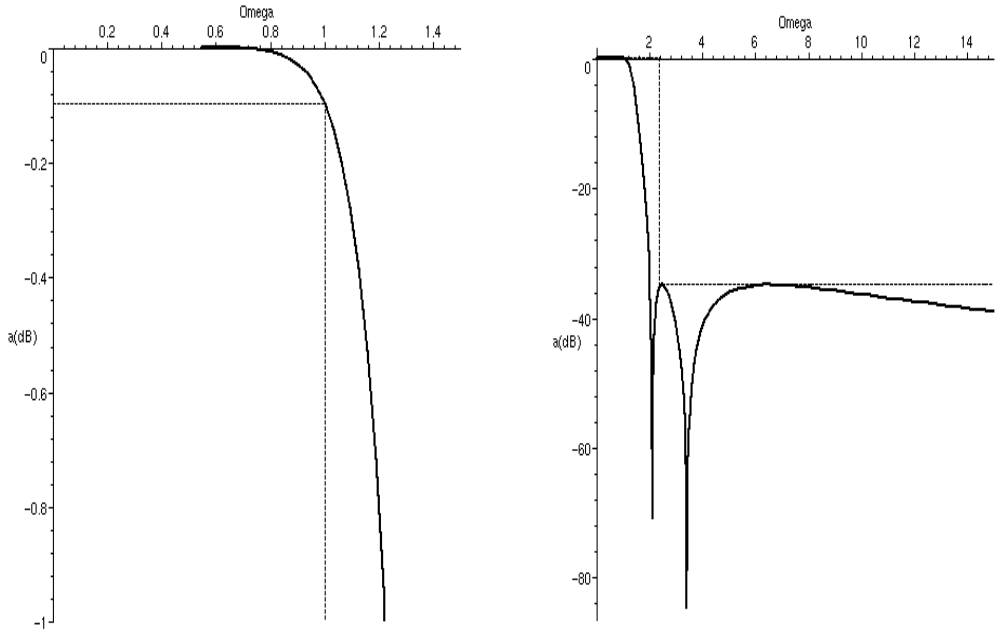


Detail přenosové charakteristiky propustného a nepropustného pásma
NDP - opravená funkce

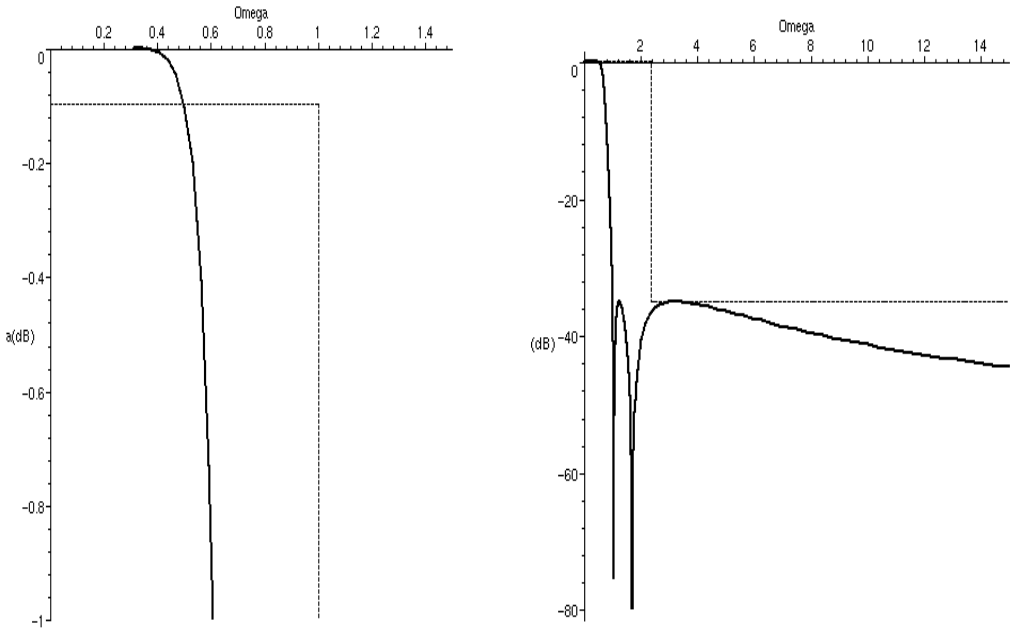


Detail přenosové charakteristiky propustného a nepropustného pásma
NDP - původní funkce

Příklad : Normovaná dolní propust, a_p=-0.1 dB, a_s=-35 dB, Ω_s=2.35, n=5



Detail přenosové charakteristiky propustného a nepropustného pásma
NDP - opravená funkce



Detail přenosové charakteristiky propustného a nepropustného pásma
NDP - původní funkce

Výpis funkce *invcheby.*

Tato funkce implementuje inverzní Čebyševovu aproximaci, se vstupními parametry, které odpovídají formě MATLABu.

Vstupní parametry : n ... stupeň přenosové funkce
ap ... přípustná chyba útlumu v propustném pásmu [dB]
as ... přípustná chyba útlumu v nepropustnm pásmu [dB]

Výstupní parametry : z ... matice nul přenosové funkce
p ... matice pólů přenosové funkce
gain ... násobná konstanta přenosové funkce

```
function [z,p,gain] = invcheby (n,ap,as)
%       INVCHEBY - Chebyshev type II analog lowpass filter prototype.
%       [z,p,gain] = invcheby (n,as,ap) returns the zeros, poles and gain
%       of an normalized prototype type II Chebyshev analog lowpass filter
%       with ap decibels of ripple in the passband, as decibels of ripple
%       in the stopband of n-th order analog lowpass filter.
%       Type II Chebyshev filters are maximally flat in the passband.
%
%       Author(s): M. Laipert, M. Vlcek, J. Vrbata
%       Programmed by Jan Vrbata, december 1999
%       This is a version for GNU OCTAVE 2.0.4

%format long;


eps=sqrt(10^(0.1*ap)-1);
k1=1/sqrt((10^(0.1*as)-1)/(10^(0.1*ap)-1));
k=1/cosh(acosh(1/k1)/n);
oms=1/k;

if (ceil(n/2)==n/2) m=floor(n/2); else m=floor((n-1)/2); end
pom=sqrt(1+((eps^2)/(k1^2)));
a=0.5*((pom+eps/k1)^(1/n)-(pom-eps/k1)^(1/n));
b=0.5*((pom+eps/k1)^(1/n)+(pom-eps/k1)^(1/n));

for f=1:m;
        pom=1/(k*cos(((2*f-1)*pi)/(2*n)));
        z(f)=0+pom*i;
        z(n-f+1)=0-pom*i;
        pom=(a^2)*(sin(((2*f-1)*pi)/(2*n)))^2+(b^2)*(cos(((2*f-1)*pi)/(2*n)))^2;
        alfa=-1*(a*sin(((2*f-1)*pi)/(2*n)))/(k*pom);
        beta=(b*cos(((2*f-1)*pi)/(2*n)))/(k*pom);
        p(f)=alfa-beta*i;
        p(n-f+1)=alfa+beta*i;
end;

if (ceil(n/2)~=n/2)
        p(m+1)=-1/(a*k);
        z(m+1)=0;
        gain=(n*k1)/(eps*k);
else
        gain=1/sqrt(1+(eps^2)/(k1^2));
end;
```

Výpis funkce *invcheby2*

Tato funkce implementuje inverzní Čebyševovu aproximaci, která vychází z tolerančního schématu normované dolní propusti.

Vstupní parametry : ap ... přípustná chyba útlumu v propustném pásmu [dB]
as ... přípustná chyba útlumu v nepropustnm pásmu [dB]
oms ... normovaná mez nepropustného pásma
opt ... opt=0 přepočítává se útlum na mezi nepropustného pásma, opt<>0 přepočítává se mez nepropustného pásma.

Výstupní parametry : z ... matice nul přenosové funkce
p ... matice pólů přenosové funkce
gain ... násobná konstanta přenosové funkce

```
function [z,p,gain] = invcheby2 (ap,as,oms,opt)
%       INVCHEBY2 - Chebyshev type II analog lowpass filter prototype.
%       [z,p,gain] = invcheby2 (ap,as,oms,opt) returns the zeros, poles and gain
%       of an normalized prototype type II Chebyshev analog lowpass filter
%       with ap decibels of ripple in the passband, as decibels of ripple
%       in the stopband and oms is normalized frequency of the start of the
%       stopband. Type II Chebyshev filters are maximally flat in the passband.
%       opt=0 ... attenuation is optimized on the start of the stopband
%       opt<>0 ... start of the stopband is optimized
%
%       Author(s): M. Laipert, M. Vlcek, J. Vrbata
%       Programmed by Jan Vrbata, december 1999
%       This is a version for GNU OCTAVE 2.0.4

%format long;

eps=sqrt(10^(0.1*ap)-1);
k=1/oms;
k1=1/sqrt((10^(0.1*as)-1)/(10^(0.1*ap)-1));

pom=acosh(1/k1)/acosh(oms);
if (pom-floor(pom)>0) n=floor(pom+1); else n=floor(pom); end

k1new=1/cosh(n*acosh(1/k));
asnew=10*log10((1+(eps^2))/(k1new^2));

knew=1/cosh((acosh(1/k1))/n);
omsnew=1/knew;

if (opt==0) k=knew; else k1=k1new; end

if (ceil(n/2)==n/2) m=floor(n/2); else m=floor((n-1)/2); end
pom=sqrt(1+((eps^2)/(k1^2)));
a=0.5*((pom+eps/k1)^(1/n)-(pom-eps/k1)^(1/n));
b=0.5*((pom+eps/k1)^(1/n)+(pom-eps/k1)^(1/n));

for f=1:m;
        pom=1/(k*cos(((2*f-1)*pi)/(2*n)));
        z(f)=0+pom*i;
        z(n-f+1)=0-pom*i;
        pom=(a^2)*(sin(((2*f-1)*pi)/(2*n)))^2+(b^2)*(cos(((2*f-1)*pi)/(2*n)))^2;
        alfa=-1*(a*sin(((2*f-1)*pi)/(2*n)))/(k*pom);
        beta=(b*cos(((2*f-1)*pi)/(2*n)))/(k*pom);
        p(f)=alfa-beta*i;
        p(n-f+1)=alfa+beta*i;
end;

if (ceil(n/2)~=n/2)
        p(m+1)=-1/(a*k);
        z(m+1)=0;
        gain=(n*k1)/(eps*k);
else
        gain=1/sqrt(1+(eps^2)/(k1^2));
end;
```