Delft University of Technology
Faculty of Electrical Engineering,
Mathematics and Computer Science
Circuits & Systems Group

(L)WDF Toolbox for MATLAB

User's Guide

Version 1.0

Ing. H.J. Lincklaen Arriëns
February 2006

Table of Contents

Introduction.

This Toolbox contains two GUIs and a number of functions to design continuous-time and discrete-time filters, featuring the design of Wave Digital and Lattice Wave Digital filters. Although (nearly) all functions are accessible through the GUIs, we start with giving a short description of some of the separate functions to indicate our way of thinking and to clarify some of the notations used.

We will not go into the details of filter theory since numerous excellent books have already been written about this subject. We assume the reader's acquaintance with common denominations like pass-band (ripple), transition-band, stop-band, etc. We also assume a reasonable experience with the MATLAB interfaces.
Even though, the toolbox presented here will turn out to be easy to use, even without an in-depth knowledge of filter theory.

At this moment the toolbox includes functions for designs based on a number of classical filter approximations, viz. Butterworth, Chebyshev, Inverse Chebyshev and Cauer designs.
Next to these, an approximation method with more freedom for tailoring the stop-band, described by Jiri Vlach [Vla69], has been added. This method also enables the design of filters using unit elements, in which case these unit elements will contribute to a better approximation of the ideal filter characteristic.
The filters above are all classified as Infinite Impulse Response (IIR) filters. With the toolbox, two types of discrete-time structures can be created that show excellent performance for IIR realizations. These structures, Wave Digital Filters (WDFs) and Lattice Wave Digital Filters (LWDFs) [Ant79] [Gaz86][Law90][Nou79], have been derived using the less-known wave digital theory. For an exhaustive description of these (L)WDFs, the reader is referred to the (invited) paper by Fettweis [Fet86].

Unlike filter theory in its early days where usually attenuation functions where used as a reference, we prefer to work with transfer function: for plots these are restricted to magnitude and phase characteristics, although it is of course possible to also work with phase delays and group delays, but this is left to the user. Being MATLAB functions, there are no reasons not to extent the number of functions with more elaborate or esoteric approximations, plot-functions, etc.

Included with the Toolbox are a number of example m-files that can help in getting acquainted with the syntax and the possibilities of the toolbox. Most of the following designs can be found in one of the example files.

In the following chapters we will briefly show how the toolbox functions work, and indicate what can be done with them, by means of listing snippets of code and the resulting pictures and command window answers. For more detailed information and help text, the reader is referred to the "(L)WDF Toolbox Reference Guide" and the several reports and test cases listed on and available from http://ens.ewi.tudelft.nl/~huib/mtbx/

Installation and requirements.

Installation is very simple: just unzip the Toolbox's zip-file to a directory of your choice. Either set MATLAB's Current Directory to this directory or add it to MATLAB's search path, with e.g.

```
>> addpath('Your_Directory');
```

Only basic MATLAB functionality is needed: there are no dependencies on other MATLAB Toolboxes.

Designing in the Continuous-time Domain.

We will describe a continuous-time filter with a transfer function, $H(s)$, which is the quotient of two polynomials in the complex frequency variable s, viz. $H(s) = \dfrac{f(s)}{g(s)}$.

The magnitude of the transfer function is given by $|H(j\omega)|$, i.e. the modulus of the complex function, and the phase angle $\arg(j\omega)$.

Usually we start by approximating a normalized low-pass design with a function, the magnitude of which approximates the ideal low-pass characteristic as good as possible, viz. showing

- a pass-band for which the input frequencies are passed (nearly) unaltered,
- a stop-band for which input frequencies are attenuated as much as possible,
- and in between a transition-band as narrow as possible.

For a normalized low-pass filter the cut-off frequency that indicates the edge of the pass-band is located at a radian frequency equal to 1. In the literature, the normalized angular frequency is commonly denoted with the symbol Ω.

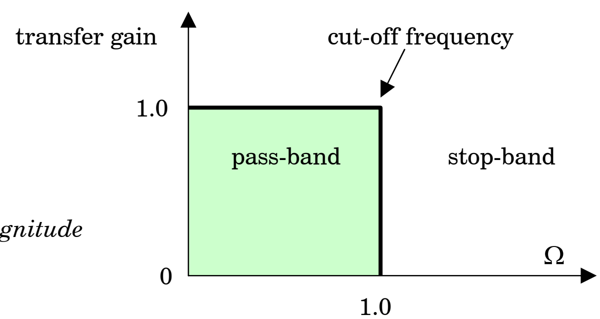


Figure 1. *The ideal low-pass magnitude transfer function.*

Comparing Butterworth Transfer Functions.

In Figure 2 a number of transfer functions that use the Butterworth approximation method have been plotted. The approximations differ from each other in the order of the transfer function.
The commands, partly basic MATLAB functions, partly functions from this toolbox, to construct this picture, are:

```
Hs = struct([]);
for i = 1:5
    Hs = [Hs; nlpf('butter',i)];
end
plotHs(Hs,2,1,[0.01 1000],1,1000, ...
    'Butterworth Characteristics', ...
        [ 'N = 1'; 'N = 2'; 'N = 3'; 'N = 4'; 'N = 5'] );
figure(1)
axis([ xlim -80 10])
xlabel('Normalized Frequency');
figure(2)
axis([ xlim -460 10])
xlabel('Normalized Frequency');
```

To be more exact, the magnitude and phase transfer functions for the 1st order upto and including the 5th order Butterworth low-pass filters, are calculated and shown.

Figure 2 clearly shows that increasing the filter order results in an increasingly better approximation of the ideal, rectangular low-pass transfer function, at the expense of larger phase-differences between in- and output signals.

See the Reference Guide for a detailed explanation of the syntax to be used for the Toolboxes' functions.

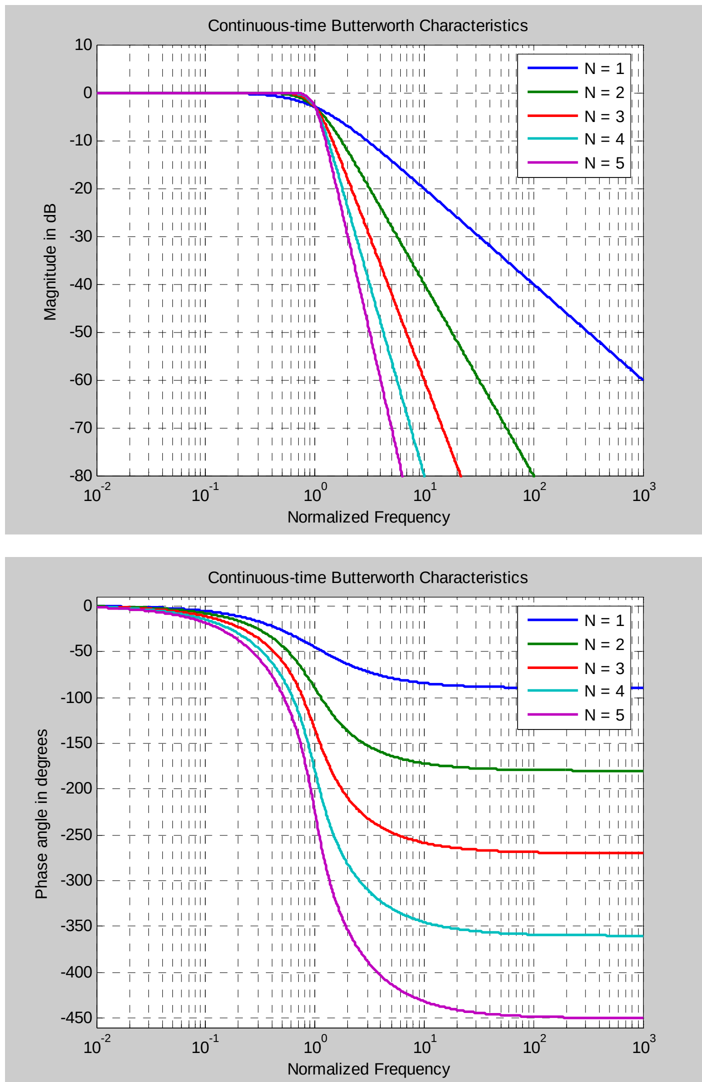


Figure 2. *Magnitude and phase transfer functions for Butterworth low-pass filters.*

All 5 transfer functions are combined in one structure array, `Hs`:

```
Hs is 5x1 struct array with fields:
    poly_fs
    poly_gs
    ident
    roots_fs
    roots_gs
```

from which we can access all individual transfer functions, e.g. the one for N= 5:

```
>> Hs(5)
ans =
     poly_fs: 1
     poly_gs: [1.0000    3.2361    5.2361    5.2361    3.2361    1.0000]
       ident: 'LP PROTOTYPE: 'butter',5,1'
    roots_fs: []
    roots_gs: [ -1.0000
                -0.8090 - 0.5878i
                -0.8090 + 0.5878i
                -0.3090 - 0.9511i
                -0.3090 + 0.9511i ]
```

$f(s)$ and $g(s)$ for this $H(s)$ are directly available in MATLAB's polynomial notation, as

$$f(s) = 1.0$$

$$g(s) = s^5 + 3.2361s^4 + 5.2361s^3 + 5.2361s^2 + 3.2361s + 1.0000$$

$f(s)$ thus has no roots, while the relationship between $g(s)$ and `roots_gs` is given by:

$$g(s) = \prod_{i=1}^{N} \left\{ s - \texttt{roots\_gs(i)} \right\}$$

According to the theory, this 5th order polynomial results in one real root, and two pairs of complex conjugated roots.

Recapitulating:

$$H(s) = \frac{1.0}{s^5 + 3.2361s^4 + 5.2361s^3 + 5.2361s^2 + 3.2361s + 1.0000}$$

Comparing different filter approximation methods.

With the following instructions a Butterworth, a Chebyshev, an Inverse Chebyshev and a Cauer approximation – all of them of order 3 – are drawn in one plot.

```
N = 3;
Hs_btw = nlpf('butter',N);
Hs_che = nlpf('cheby',N,1,1);
Hs_ich = nlpf('invcheby',N,40,1);
Hs_cau = nlpf('cauer',N,1,40,'a',1);
plotHs([Hs_btw;Hs_che;Hs_ich;Hs_cau],2,1,[0.1 100],0,1000, ...
           'Transfer Characteristics', ...
           [ ' butterworth '; ' chebyshev   '; ' inverse cheb'; ' cauer       ' ] );
axis([ xlim -60 10])
xlabel('Normalized Frequency');
```

For a better comparison of the filter shapes, the normalization of the cut-off frequencies is chosen such that all magnitude transfer function show a 3dB attenuation at `Normalized Frequency` = 1.

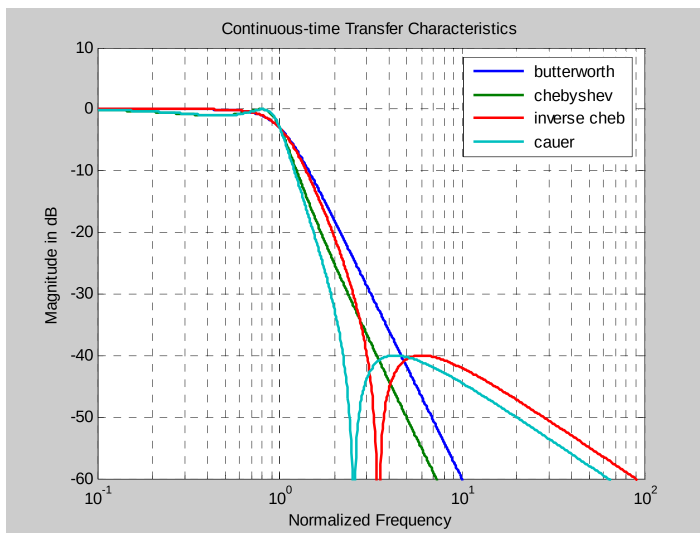


Figure 3. *Magnitude transfer functions for 3rd order Butterworth, Chebyshev, Inverse Chebyshev and Cauer low-pass filters.*

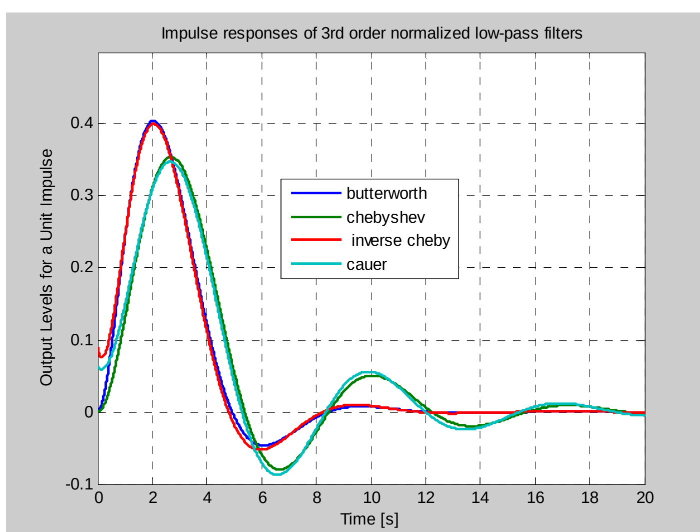


Figure 4. *Impulse response in the time domain for the filters of Figure 3.*

From Figure 3, the phase plot is omitted. To obtain a phase plot, follow the previous example: it should show that the phase characteristics are less smooth then the Butterworth' one.

In Figure 4, the impulse responses of the same four filters are shown, obtained using MATLAB's `impulse.m` function (from the Control Toolbox). It can be seen that these filters belong to the class of Infinite Impulse Response (IIR) filters.

The parameters for the Cauer filter are now:

```
>> Hs_cau
Hs_cau =
     poly_fs: [0.0638 0 0.4121]
     poly_gs: [1 0.9015 1.0560 0.4121]
       ident: 'LP PROTOTYPE: 'cauer',3,1,40,'A',1,1'
    roots_fs: [2x1 double]
    roots_gs: [3x1 double]
>> Hs_cau.roots_fs
ans =
        0 - 2.5420i
        0 + 2.5420i
>> Hs_cau.roots_gs
ans =
  -0.4826
  -0.2094 - 0.9000i
  -0.2094 + 0.9000i
>>
```

from which can be read that $H(s) = \dfrac{0.0638\,s^2 + 0.4121}{s^3 + 0.9015\,s^2 + 1.0560\,s + 0.4121}$

The locations of the roots of $g(s)$ in the complex plane are shown in Figure 5 and can be seen to be connectible with an elliptical function.

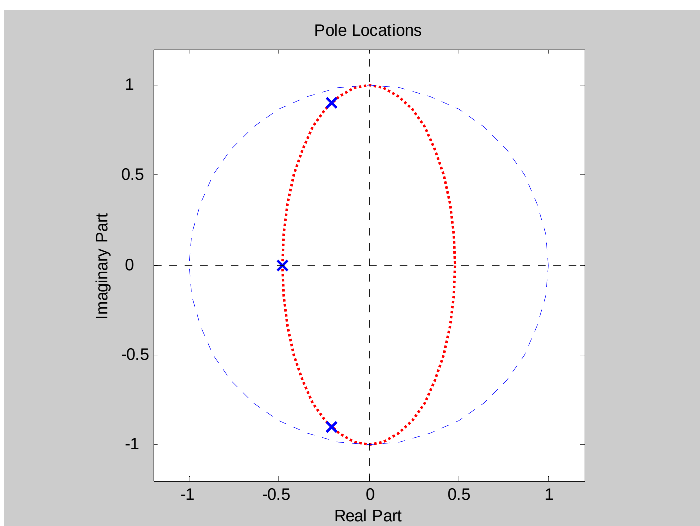


Figure 5. *Location of the poles of the 3rd order Cauer filter from this example.*

From filter theory it is known that *odd* order Cauer approximations are usually realizable as lumped element ladder circuits (see also later on). This is not so for *even* order transfer functions, which end up with negative component values and thus are not realizable as ladder structures. The cause is that for even order approximations, both at zero as at infinite frequencies, the transfer gain should have a distinct value instead of 1 or zero. Skwirzynski [Skw65] described a solution for this problem, viz. to either only shift the highest notch in the stop-band to infinity (Type ‘B’), or to also shift the first peak in the pass-band to zero frequency (Type ‘C’). Both Type B and C transfer functions result in realizable ladder structures.
The unaltered function will be denoted as Type ‘A’.

The following functions are plotted in Figure 6:

```
nlpf('cauer',4,1,45,'A',1)
nlpf('cauer',4,1,45,'B',1)
nlpf('cauer',4,1,45,'C',1)
```

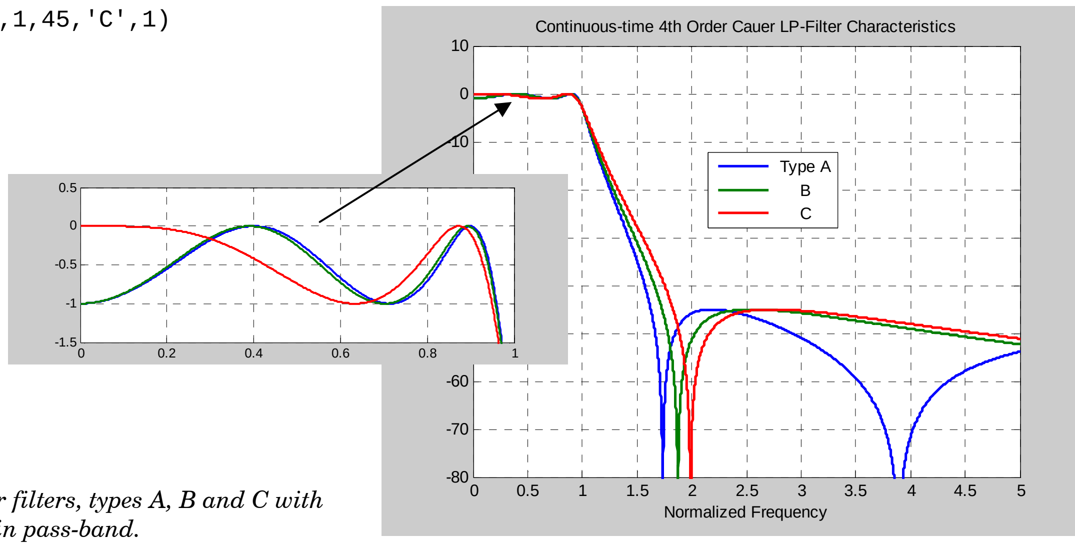


Figure 6. *4^{th} Cauer filters, types A, B and C with zoomed-in pass-band.*

Cauer Filter Parameter Conversion.

If you are more accustomed with specifying Cauer filters in terms of reflection coefficient ρ and modular angle θ, then function `cc2pars.m` can help with translating ρ and θ into the parameters used by the Toolbox.
A filter, identified in a catalog classification as e.g. ‘C06 B 25 48’ can be translated with

```
>> [N,rp,rs,ftype,Wn,normtd] = cc2pars(6,25,48,'b');
```

and then computed with e.g.

```
>> [Hs,wp] = Hs_cauer(N,rp,rs,ftype,Wn,normtd);
```

or translated into a ladder structure using about the same parameters (see later on).
Functions `ripple2rho` and `rho2ripple` are simple conversion routines with self-explanatory names.

Frequency Transformations.

With the toolbox the following well-known frequency transformations are possible:

`nlp2lp:` converts a normalized low-pass to low-pass with an arbitrary cut-off frequency,
`nlp2hp:` creates a high-pass transfer function,
`nlp2bp:` obtains a band-pass filter, and
`nlp2bs:` transforms the normalized low-pass into a band-stop filter.

Below follows an example of the `nlp2bp` function, given our previously calculated 3rd order, type 'A', Cauer filter:

```
Hs_cau_bp = nlp2bp(Hs_cau,1,1);
Hs_cau_b1 = nlp2bp(Hs_cau,1,2);
Hs_cau_b2 = nlp2bp(Hs_cau,10,10);
plotHs([Hs_cau_bp;Hs_cau_b1;Hs_cau_b2],2,1,[],0,1000, ...
            '3rd order Cauer lp-bp Transformations', ...
            [ ' fc =   1, BW =  1'; ' fc =   1, BW =  2'; ' fc = 10, BW = 10 ' ] );
axis([ xlim -60 20])
xlabel('Normalized Frequency');
```

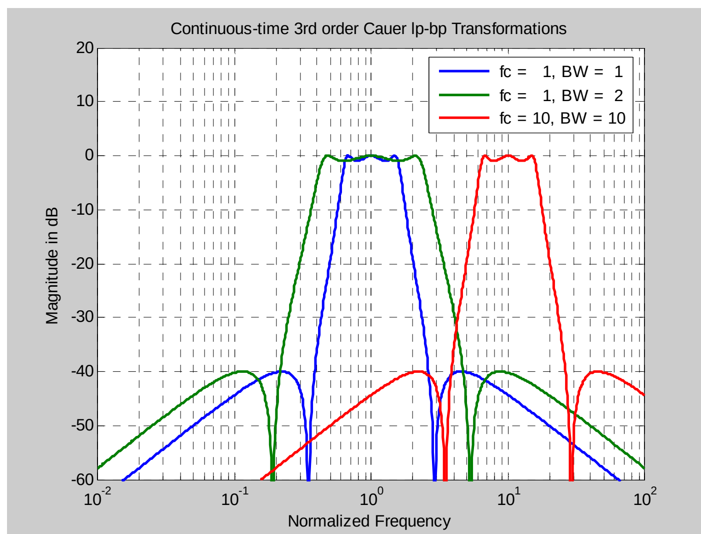


Figure 7. *Three 6th order Cauer filters, each obtained with a normalized low-pass to band-pass transformation.*

Notice that the transformation causes the band-pass filter to be arithmetically symmetric around the center-frequency in Figure 7 (thus geometrically symmetric if plotted with a logarithmic frequency scale).

To obtain the transfer function of a low-pass filter with a -3dB cut-off frequency at 15 kHz, e.g.

```
>> Hs_cau_15k = nlp2lp(Hs_cau,2*pi*15000);
```

Note that if one is only interested in low-pass filter designs, it is also possible to use the functions `Hs_butter`, `Hs_cheby`, `Hs_invcheby`, `Hs_cauer` and/or `Hs_Vlach` which allow a parameter `cutOffFrequency` to be passed directly, instead of the combination `nlpf` and `nlp2lp`.

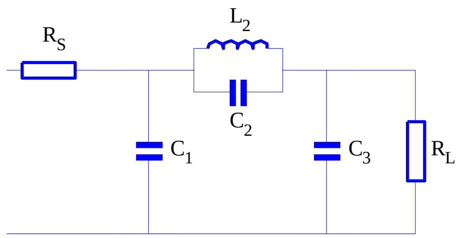
Ladder Circuits.

The normalized 3rd order Cauer filter from a previous example can be realized as a ladder circuit:

```
>> nlpLadder = nlp_ladder('cauer',3,1,40,'a',1);
```

```
Rs     1.00000 Ohm
C01    2.07190 F   in shunt arm
L02    0.98116 H, parallel with
C02    0.15773 F   in series arm
C03    2.07190 F   in shunt arm
RL     1.00000 Ohm
```

a)



or, if its dual realization form is wanted, starting with an inductor in a series arm (use 'z'):

```
>> nlpLadder_d = nlp_ladder('cauer',3,1,40,'a',1,'z');
```

```
Rs     1.00000 Ohm
L01    2.07190 H   in series arm
L02    0.15773 H, in series with
C02    0.98116 F   in shunt arm
L03    2.07190 H   in series arm
RL     1.00000 Ohm
```

b)

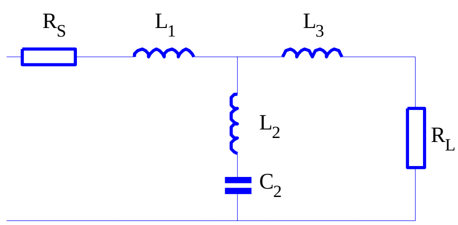


Figure 8a and b. *Two functionally equivalent ladder circuits.*

It was indicated above that even order Cauer filters of type 'A' are not realizable as ladder structures with positive element values. Below we show 4th order designs, realizing a Type B and a Type C transfer function. The ladder structure is the same for both Types, only the component values differ. Because of the fact that for a Type B function the transfer gain at zero frequency differs from 1, the source and load resistances cannot be equal. If this should be a problem, then a Type C will be needed.

4th order Cauer low-pass, type B: *different* source and load resistance values are needed.

```
Rs     1.00000 Ohm
C01    1.84916 F   in shunt arm
L02    0.86103 H, parallel with
C02    0.41910 F   in series arm
C03    2.65202 F   in shunt arm
L04    0.83131 H   in series arm
RL     0.37598 Ohm
```

same specifications as above, but now a type C versic
equal source and load resistances, both 1.0 Ω.

```
Rs     1.00000 Ohm
C01    1.40699 F   in shunt arm
L02    1.29079 H, parallel with
C02    0.24945 F   in series arm
C03    1.50477 F   in shunt arm
L04    1.62097 H   in series arm
RL     1.00000 Ohm
```

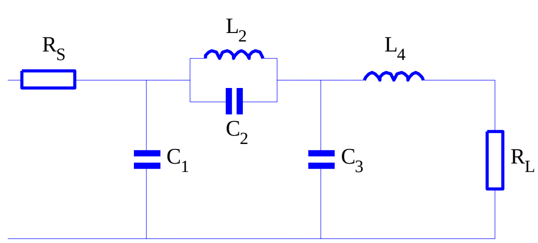


Figure 9. *Ladder circuit for both 4th order types B and C Cauer low-pass filters.*

The same transformation methods that we mentioned before are also applicable on ladder structures, called resp. `nladder2lp`, `nladder2hp`, `nladder2bp` and `nladder2bs` here.

With `nladder2bp`, the ladder of Figure 8a will be transformed into a structure that realizes the transfer function of `Hs_cau_bp` (shown in blue in Figure 7).

```
>> bpLadder  = nladder2bp(nlpLadder,1,1);
>> showLadder(bpLadder,3,'after 3rd order Cauer lpladder-bpladder Transformation');
```

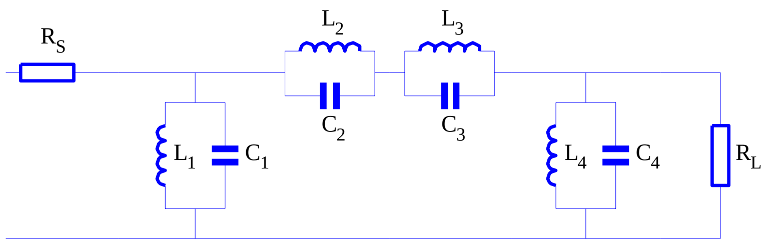


Figure 10. *6^{th} order Cauer band-pass ladder circuit, resulting from a transformation of the ladder in Figure 8a.*

```
Rs    1.00000 Ohm
L01   0.48265 H, parallel with
C01   2.07190 F   in shunt arm
L02   0.67867 H, parallel with
C02   0.17663 F   in series arm
L03   5.66148 H, parallel with
C03   1.47347 F   in series arm
L04   0.48265 H, parallel with
C04   2.07190 F   in shunt arm
RL    1.00000 Ohm
```

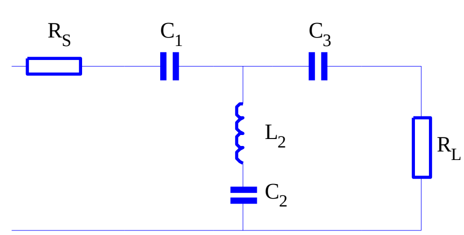
Another example:
a high-pass filter with a cut-off frequency of 8 kHz, with source and load resistance values of 2700 Ω .

This is a two-step process (transform, then scale), for we first have to execute the transformation:

```
>> nlpLadder = nlp_ladder('cauer',3,1,40,'a',1,'z');
>> hpLadder  = nladder2hp(nlpLadder,2*pi*8000);   % 2*pi for realistic frequencies!
```

```
Rs    1.00000 Ohm
C01   0.00001 F   in series arm
L02   0.00002 H, in series with
C02   0.00013 F   in shunt arm
C03   0.00001 F   in series arm
RL    1.00000 Ohm
```

a)



Notice that every time a ladder structure is computed, its magnitude transfer function will be reconstructed from the ladder's topology and will be compared with the original one.

b)

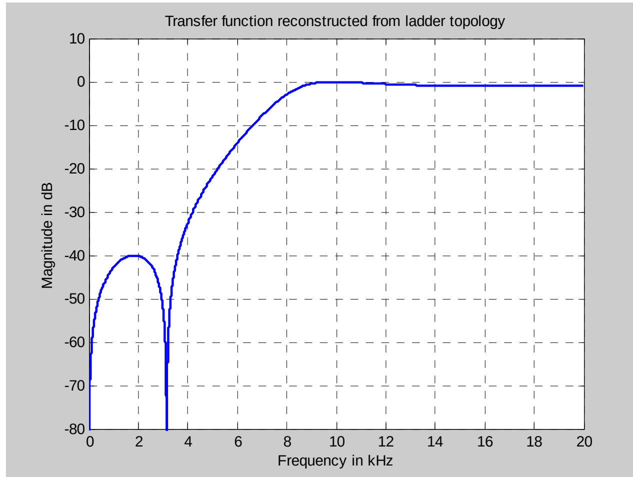


Figure 11a and b. *High-pass ladder circuit and its Magnitude Transfer function for the component values in Table 1.*

In the Toolbox as is, there is no special function to do impedance scaling, since the design of real lumped element filters is not really the goal of this toolbox.
However, since the structure of the ladder is given (see the Reference Guide for a detailed description),

```
hpLadder =
    elements: 'rcScR'
      values: [6x1 double]
```

the following code will do the job (also applicable to band-pass and band-stop structures, but no unit elements allowed):

```
function scVals = impScale(Ladder,R)
  scVals   = Ladder.values;
  elements = strrep(lower(Ladder.elements),'s','lc');
  elements = strrep(elements,'p','lc');
  isC = (elements == 'c');
  scVals( isC) = scVals( isC)/R;
  scVals(~isC) = scVals(~isC)*R;
```

For `hpLadder` just calculated, and the given $R = 2700\,\Omega$, we can find:

Table 1.

Rs	2k7	Ω
C01	3.6	nF
L02	54.7	mH
C02	46.7	nF
C03	3.6	nF
RL	2k7	Ω

Check in the plot that the notch is really at the calculated frequency of $\dfrac{1}{2\pi\sqrt{L02 \cdot C02}} = 3.147$ kHz.

Note:
Several books, like [Zve67] and [Chr66], contain numbers of tables to calculate the values of ladder structures with. When comparing those values with the ones calculated with the Toolbox, be sure to use the same frequency normalization method with the Toolbox as has been done by the author(s).

Some 3rd order Vlach Low-pass Transfer Functions.

In [Vla69], J. Vlach described an approximation method that we have implemented as the 'Vlach-method'. The basic idea had been published by Sharpe [Sha54], but was extended, made more accurate, and was also applied to band-pass approximations by Vlach.

Next to that, this method has the advantages that

- notches can be inserted in the stop-band (stop-bands for bandpass filters), and that
- unit elements that contribute to the filter approximation can be inserted.

When no notches are applied, a Vlach low-pass equals a Chebyshev approximation. This will be shown in the next example for a 3rd order filter. Also, we will investigate what will be the result when we position the notch at the exact frequency as where a 3rd order Cauer approximation (same pass-band ripple and same frequency normalization method) would have located its notch.

```
% First find the frequency of the notch in the stop-band for a 3rd order Cauer filter.
% But the relation ws=1/wp holds true for the 'symmetric' Cauer transfer functions.
[Hsx,wp] = Hs_cauer(3,1,40,'a',1,-1);
wp2 = wp(2);              % skip wp at w=0
ws2 = 1/wp(2);
% Again the Cauer filter, but now with its cut-of frequency at the -3dB point.
[Hs_cau,wp] = Hs_cauer(N,1,40,'a',1,1);
ws = wp(2)/wp2 * ws2;     % This will be the notch frequency for our cut-off
                          % frequency reference method (-3 dB)

% Four Vlach transfer functions:
Hs_vla_a = Hs_Vlach(3,1,1,[],0,1);      % no notch
Hs_vla_b = Hs_Vlach(3,1,1, 3 , 0,1);    % notch at w = 3
Hs_vla_c = Hs_Vlach(3,1,1, ws , 0,1);   % notch at Cauer's ws
Hs_vla_d = Hs_Vlach(3,1,1, 2 , 0,1);    % notch at w = 2
```

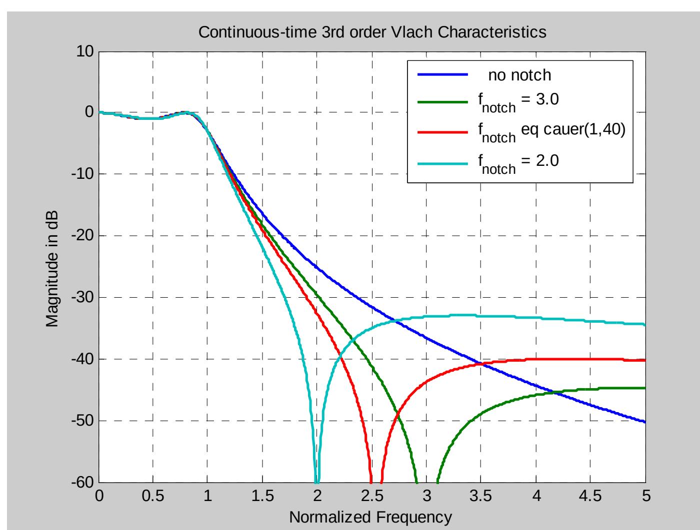


Figure 12. *Four 3rd order Vlach filters.*

MATLAB shows that

```
>> ws
ws =
   2.54202809474093
```

Alternatively, we could have realized that the roots of $f(s)$, at which frequency or frequencies the transfer function $H(s)$ becomes zero, immediately could have given us the value of `ws`.
Indeed, we can find that

```
>> Hs_cau.roots_fs
ans =
                 0 - 2.54202809474093i
                 0 + 2.54202809474093i
```

from which we see that the roots are purely imaginary and equal to $\pm j$`ws`.

To be exact: in fact `ws` equals

```
>> fs = conv( [1 -Hs_cau.roots_fs(1)], [1 -Hs_cau.roots_fs(2)] );
>> ws = sqrt( fs(end) );
```

Now compare `Hs_che` with `Hs_vla_a`.
The `poly_fs`'s and `poly_gs`'s are –apart from accuracy matters in the computation– almost exactly equal, i.e. a Vlach approximation without additional notches equals the Chebyshev approximation.

```
Hs_che =
     poly_fs: 0.37434096543120
     poly_gs: [1 0.90270351281101 1.03309585058947 0.37434096543120]
       ident: 'LP PROTOTYPE: 'cheby',3,1,1,1'
Hs_vla_a =
     poly_fs: 0.37434096541660
     poly_gs: [1 0.90270351279928 1.03309585056261 0.37434096541660]
       ident: 'LP PROTOTYPE: 'vlach',3,1,1,[],1,0'
```

Also, there is a very large similarity between `Hs_cau` and `Hs_vla_c` (the 'simulated' Cauer version). This means that if the notch is positioned at the same frequency where the Cauer approximation would have located it, then the Vlach and Cauer transfer functions will be identical.

```
Hs_cau =
     poly_fs: [0.06377455880975 0 0.41210525743689]
     poly_gs: [1 0.90152483145585 1.05600921457728 0.41210525743689]
       ident: 'LP PROTOTYPE: 'cauer',3,1,40,'A',1,1'
Hs_vla_c =
     poly_fs: [0.06377455885052 0 0.41210525770037]
     poly_gs: [1 0.90152483163557 1.05600921500788 0.41210525770037]
       ident: 'LP PROTOTYPE: 'vlach',3,1,1,[2.542],1,0'
```

Note: Given a filter order `N`, the maximum number of notches that can be inserted is `floor(N/2)`.

Vlach Band-pass Transfer Functions.

The Vlach approximation is perfectly suitable for designing band-pass filters directly (without a transformation), since the cut-off frequencies of the pass-band can be chosen separately and without affecting each other, while the position of notches (if any) in the stop-bands can be tuned to be exactly at the frequencies of disturbing signals.

Also, more or less geometrically symmetric filters are obtainable, as in shown in the next example:

```
>> Hs = Hs_bpVlach(10,0.1,[1.5 2.5], [0 1.0 3.0],0,1);
>> plotHs(Hs,1)
```

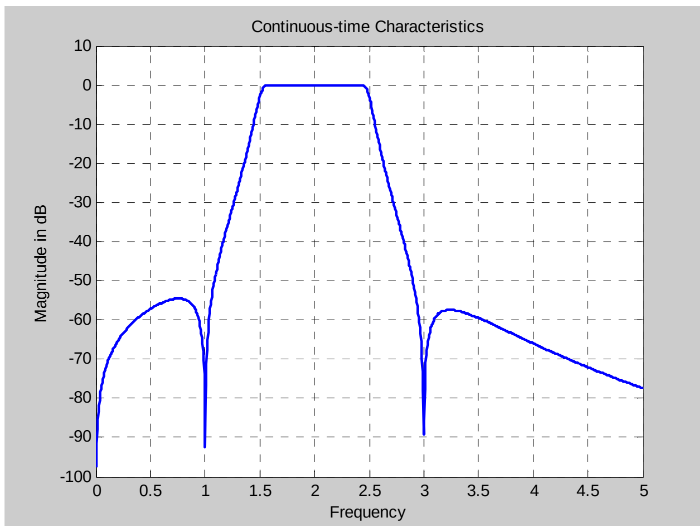


Figure 13. *Vlach 10^{th} order band-pass design with one notch in the lower and one notch in the upper pass-band.*

As can be seen from Figure 13, the transfer function between the notches is (almost) symmetrical around 2.0, i.e. $\frac{f_{c\_lower} + f_{c\_upper}}{2}$ (the geometrical mean).

We will reveal more about the capabilities of the Vlach functions when treating microwave filter designs.

Note:
For a band-pass filter to be realizable there should be at least one notch at frequency 0.
Given a filter order `N`, the maximum number of notches (notches at zero are treated differently because of the theoretical symmetry with the negative frequency part of the spectrum) is given by

```
(number of notches in 0) + 2*(number of notches not in 0) <= N.
```

Designing in the Discrete-time Domain.

We assume that the reader is familiar with the theory on the *z*-transform and knows about the methods to translate a continuous-time function into a discrete-time function.

In this Toolbox, we will only use the *bilinear transformation*, mainly because of the fact that the theory of Wave Digital Filters is also based on this method.

The transformation uses the substitution $s = \frac{2}{T} \cdot \frac{z-1}{z+1}$

In here, we will always set $T = 2$, the result of which is that the continuous-time Normalized Frequency value 1 always transforms into the discrete-time value 0.25 (i.e. a frequency value equal to ¼ of the Sample-frequency) and vice versa. The complete relationship between the continuous-time frequency scale (from 0 to ∞) and the discrete-time relative frequency scale (0 to 0.5) is given by an $\tan^{-1}$-function, as shown in Figure 14.

The discrete-time transfer function is denoted as $H(z)$.

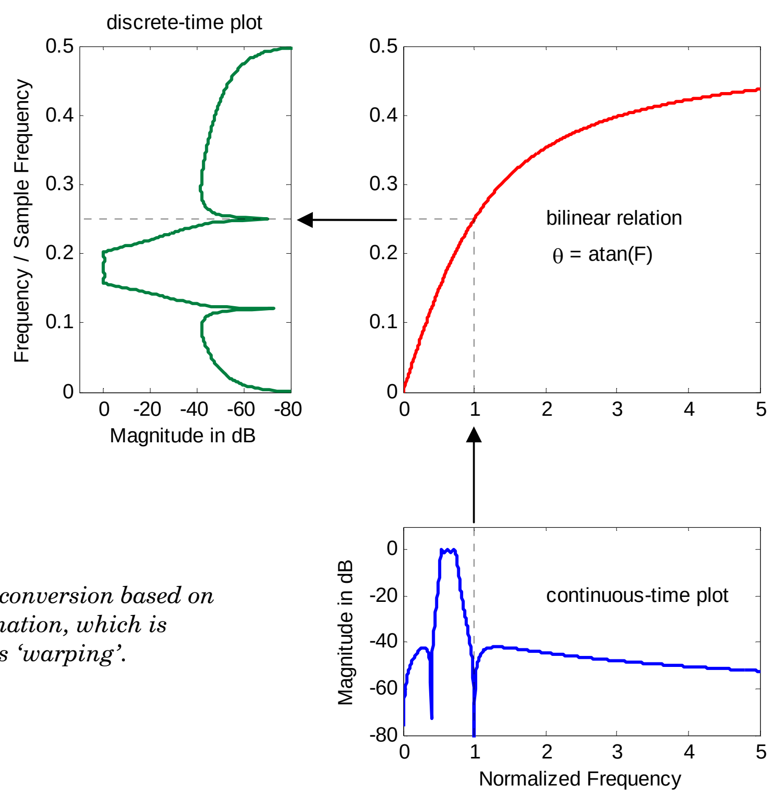


Figure 14. *Domain conversion based on the bi-linear transformation, which is commonly indicated as 'warping'.*

The functions `fs2fz`, `fz2fs`, `Hs2Hz`, `Hz2Hs` and `plotHz` are specially meant for the design of discrete filters and for switching between the domains.

The actual design of a discrete-time filter is a two step process, i.e. the design of a continuous-time filter followed by the bilinear transformation.
Since the relation between frequencies in the two domains is exactly known, it is possible to pass 'pre-warped' discrete-time frequencies to the continuous-time design, that, after transformation will result in the correct discrete-time frequencies.

```
>> Hs = Hs_Vlach(7,0.01, fz2fs(0.1), fz2fs([0.15 0.2 0.35]), 0,0);
>> Hz = Hs2Hz(Hs);
>> plotHz( Hz,1 );
```

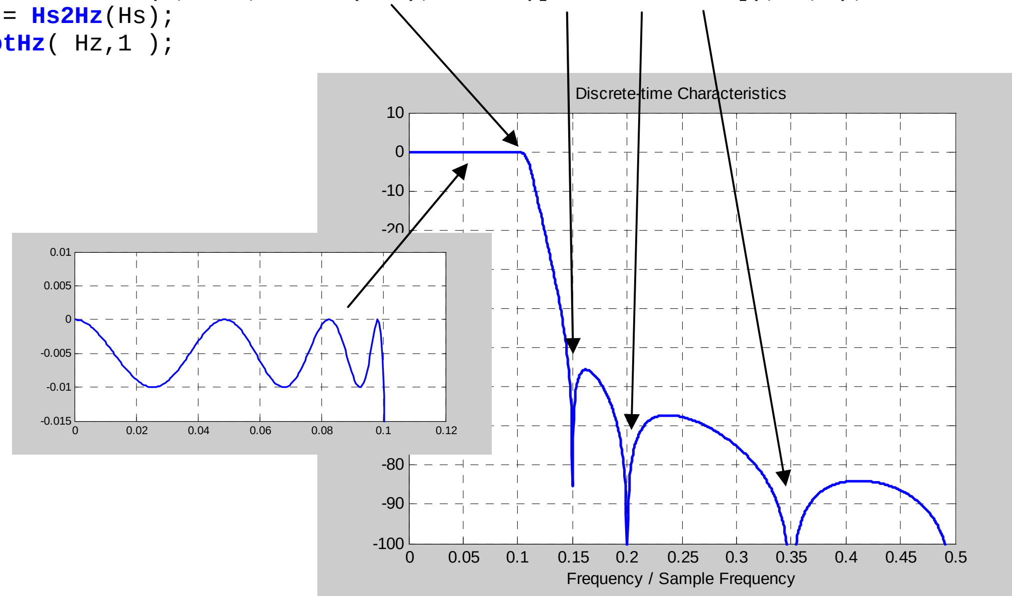


Figure 15. *A discrete-time domain Vlach low-pass filter with its cut-off frequency and three notches specified in the discrete-time domain.*

The complete transfer function $H(z)$ of this example then turns out to be

$$H(z) = \frac{10^{-2} \cdot \left(0.1212 + 0.0463\,z^{-1} + 0.1212\,z^{-2} + 0.1498\,z^{-3} + 0.1498\,z^{-4} + 0.1212\,z^{-5} + 0.0463\,z^{-6} + 0.1212\,z^{-7}\right)}{1.0 - 4.8352\,z^{-1} + 10.6717\,z^{-2} - 13.7583\,z^{-3} + 11.1224\,z^{-4} - 5.6162\,z^{-5} + 1.6363\,z^{-6} - 0.2119\,z^{-7}}$$

Just for fun: Comparing a Cauer with a FIR Filter.

It is interesting to compare toolbox designs with the familiar FIR filters.
Shown here is a 7^{th} order Cauer filter (with a pass-band ripple of 0.1 dB, a stop-band ripple of 50 dB, `freqNormMode` set to 0) next to a 128 taps FIR filter that has been designed using the Remez algorithm from the Signal Processing Toolbox (the original function `remez.m` has been renamed to `firpm.m`).

In MATLAB's notation:

```
>> n = 127;
>> f = [ 0 0.3 0.34102 1 ];
>> m = [ 1 1 0 0 ];
>> b = firpm( n, f, m );    % Parks-McClellan optimal equiripple FIR filter design.
```

In here, vector `f` is adjusted such that the stop-band ripple of the FIR filter also shows peaks at maximally –50 dB.

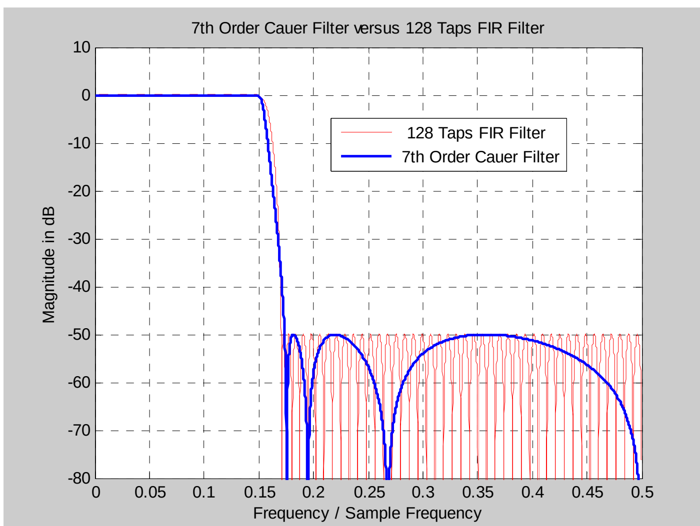


Figure 16. *Comparing the Magnitude Transfer functions of a FIR filter and a 7^{th} order discrete-time Cauer filter.*

Note that the FIR filter needs 128 multiplications (or 64, depending on the implementation method) while if the Cauer filter is implemented as a (Lattice) Wave Digital Filter, it only needs 7 or 8 multiplications.

Designing Wave Digital Filters (WDFs).

A Wave Digital Filter is essentially a translation of a ladder structure into the discrete-time domain. See [Fet86] and all other literature on WDFs to learn about all the benefits of this kind of filters. Unfortunately, there exists very little software to design WDFs. With this Toolbox it is possible to translate all the ladder structures that can be designed with the Toolbox's functions into WDFs.

For the translation, the circuits have to be described using the so-called wave variables, A and B, instead of the common voltages and currents. This way of describing circuits has originally been developed for microwave and high-frequency circuitry.
According to the theory, components like inductances and capacitors translate into Delay Elements, while the 'wiring' needed to connect these components translates into 'adaptors'. These adaptors contain computational elements to perform additions, sign-inversions (or subtractions) and one or two multiplications with constant coefficients. The ladder circuit determines the structure of the WDF, while the values of the lumped elements in the ladder determine the multiplication coefficients.

An overview of the adaptors supported by this Toolbox are given in Figures 18 and 19, with an additional explanation of the (commonly used in Signal Processing) symbols in Figure 17.

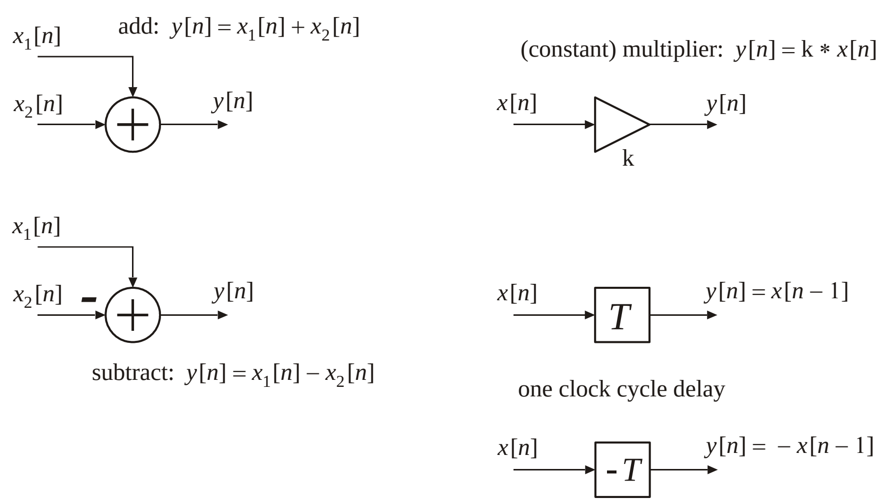


Figure 17. *Explanation of the basic elements.*

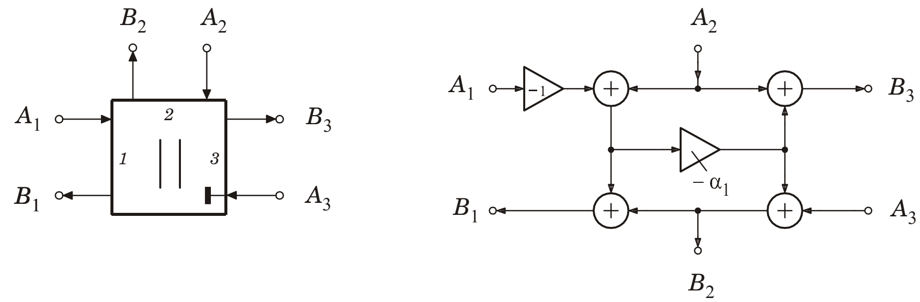


Figure 18a. *Symbol and Block scheme for a* ‘matched 3 port parallel adaptor’

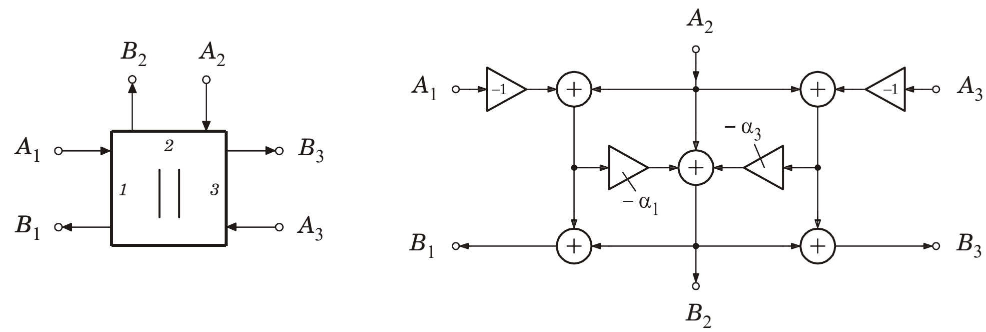


Figure 18b. *Symbol and Block scheme for a* ‘3 port parallel adaptor’

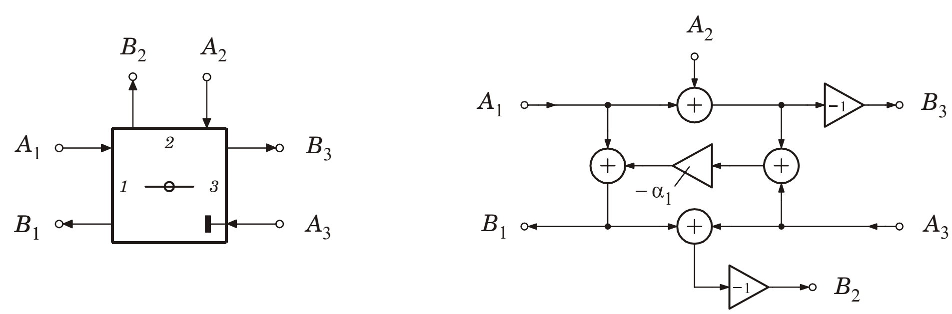


Figure 18c. *Symbol and Block scheme for a* ‘matched 3 port serial adaptor’

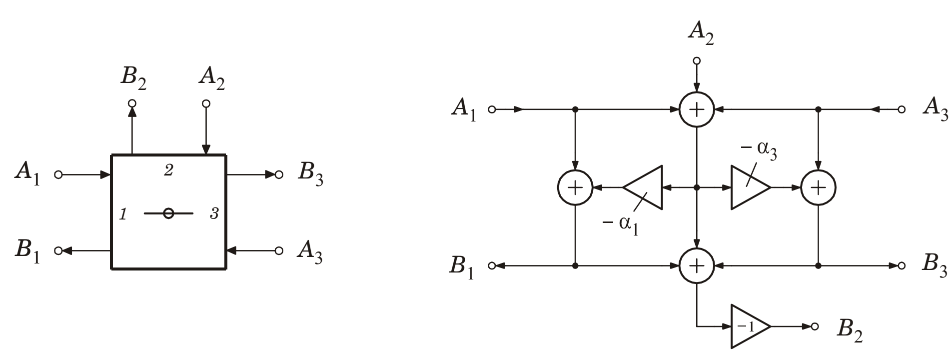


Figure 18d. *Symbol and Block scheme for a* ‘3 port serial adaptor’

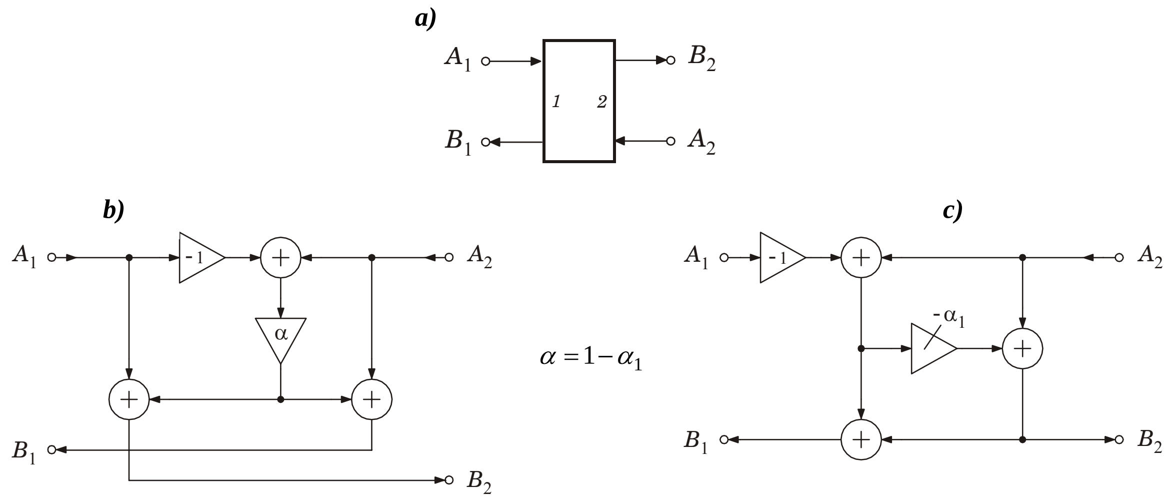


Figure 19. *Symbol* ***a)****, and two functionally equivalent realization schemes* ***b) and c)*** *for a* '2 port adaptor'

Given a structure `xLadder` that has been determined before, the syntax to obtain a WDF is simply

```
>>  WDF = ladder2WDF(xLadder)
```

Here `WDF` will also be a structure, containing the fields

```
WDF.wdaStruct
WDF.wdaNo
WDF.mulFacs
```

`WDF.wdaStruct` describes the WDF block diagram, where the block diagram is represented with 2 strings, one describing the adaptors in the signal path (bottom row), the second one (top row) describing the elements or adaptors connected to the serial or parallel ports of the first mentioned adaptors.
So, the bottom row can only consist of the following codes

- `'s'` - for a reflection free 3-port serial adaptor,
- `'p'` - a reflection free 3-port parallel adaptor,
- `'S'` - a 3-port serial adaptor with two coefficients,
- `'P'` - a 3-port parallel adaptor with two coefficients,
- `'m'` - an output inverter or scaling factor, if needed.

Some rules have been formulated for connecting the adaptors to each other:
For all these adaptors, port 1 is the input, port 3 the (reflection free) output, and port 2 the interface to the top row elements.
Each element in the top row string is connected to port 2 of the adaptor in the same position in the bottom row string. Possible codes are:

- `'+'` - a single delay element (translation of a capacitance),
- `'-'` - a delay element in series with an inverter (inductance),
- `'s'` - a reflection free serial adaptor (series LC resonator),
- `'p'` - a reflection free parallel adaptor (parallel LC resonator),
- `'x'` - for an empty slot.

With the 's' and 'p' adaptors, port 1 is connected to a single delay element (translation of the capacitance), port 2 to a delay element in series with an inverter (the inductance), while the reflection free port 3 is connected to port 2 of the corresponding bottom row adaptor.

`WDF.wdaNo` defines the numbering of the individual adaptors,

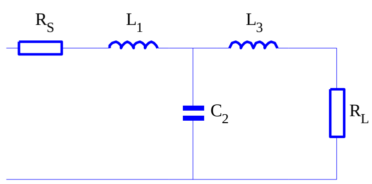
`WDF.mulFacs` lists the multiplication coefficients of the adaptors, starting from adaptor one. The very last adaptor in a asymmetric or the middle adaptor in a symmetric structure, which is not reflection free, needs two coefficients, while, if the bottom row string ends with an 'm', the last value will be the scaling coefficient.

The next commands create a WDF from a 3rd order Butterworth low-pass:

```
>> nlpLadder = nlp_ladder('butter',3,'z');

   Rs    1.00000 Ohm
   L01   1.00000 H   in series arm
   C02   2.00000 F   in shunt arm
   L03   1.00000 H   in series arm
   RL    1.00000 Ohm
```

a)




```
>> lpLadder = nladder2lp(nlpLadder,fz2fs(0.15));
>> WDF = ladder2WDF(lpLadder)
wdfType not specified : '3p' assumed ...

Adaptor 1:  3p serial,   p3 matched :  alpha1 =  0.33754
Adaptor 2:  3p parallel, p3 matched :  alpha1 =  0.07918
Adaptor 3:  3p serial               :  alpha1 =  0.14675
                                       alpha3 =  0.62555

Adaptor   port1        port2       port3
-----------------------------------------
   1       A1/B1       -T_L01     3pA(2).p1
   2     3pA(1).p3      T_C02     3pA(3).p1
   3     3pA(2).p3     -T_L03       A3/B3

WDF =
    wdaStruct: [2x3 char]
        wdaNo: [2x3 double]
      mulFacs: [4x1 double]
```

b)

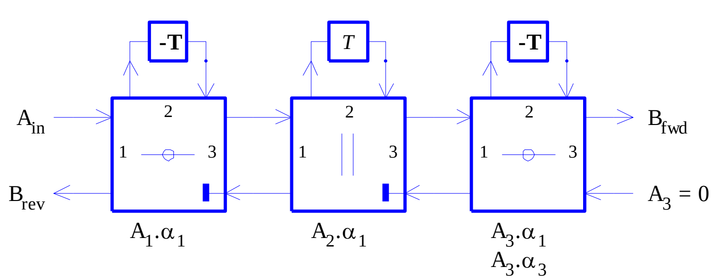
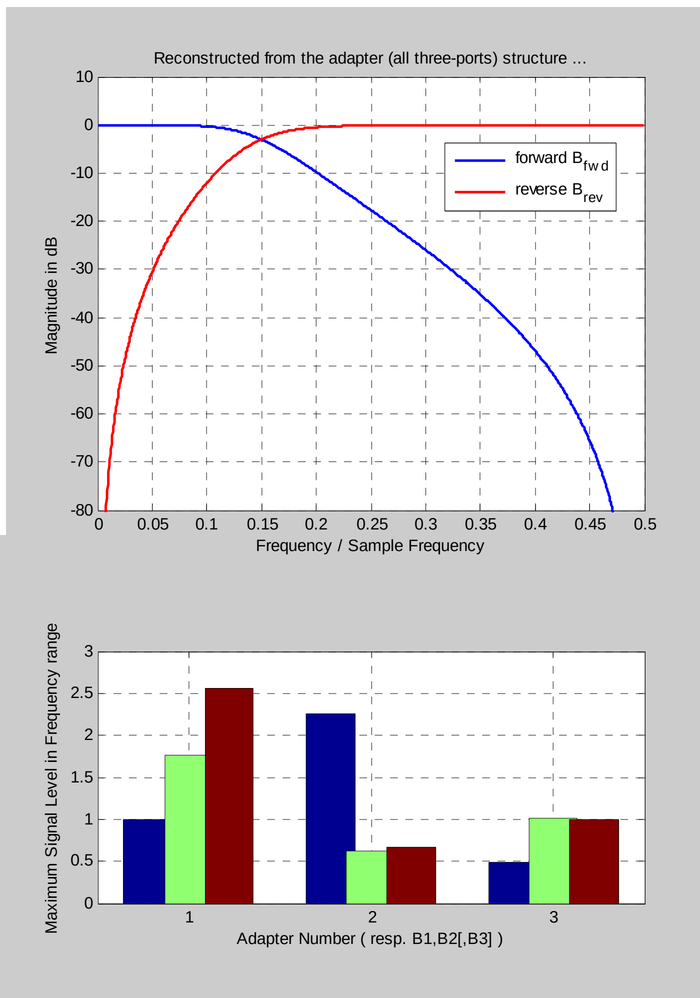


Figure 20a and b. *3rd order Butterworth low-pass realizations as a ladder circuit and as a Wave Digital Filter.*

WDFs actually show two outputs, denoted here as B_{fwd} and B_{rev}. (see Figure 21a).
B_{fwd} is the 'wanted' or 'forward' output (in terms of scattering matrix notation also known as S_{21}), while B_{rev} is the 'reflected' or 'reverse' output (S_{11}), which shows a complementary characteristic w.r.t. B_{fwd}, viz. $\left|\frac{B_{fwd}}{A_{in}}\right|^2 + \left|\frac{B_{rev}}{A_{in}}\right|^2 = 1$ or $\left|B_{fwd}(j\omega)\right|^2 + \left|B_{rev}(j\omega)\right|^2 = 1$.

If the filter is specified to be a low-pass, then the reflected output will show a high-pass behavior. In many designs, however, this reflected output is hardly usable, showing particularly bad stop-band behavior. Butterworth and bireciprocal Cauer designs (see later) can make good use of this feature.

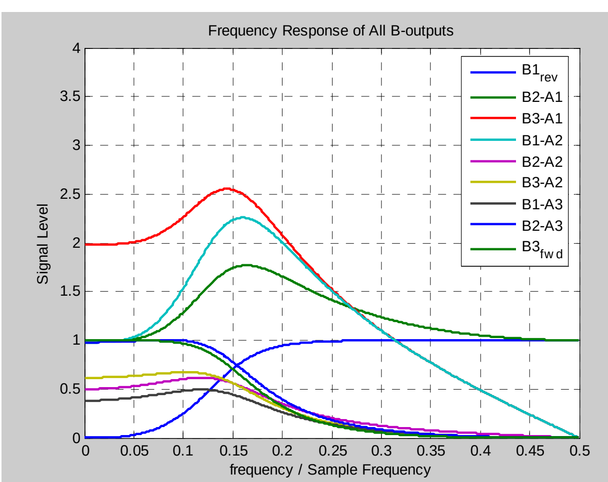
Just like the passive ladder circuits can show resonance behavior with the result that on some of the nodes voltages or currents can be larger than the input values, WDFs can internally also be more sensitive for certain frequencies than initially expected. These effects should be taken into account when implementing the filter in hardware with e.g. fixed-point arithmetic support.
This phenomenon has been made visible if Figure 21b) where the complete frequency responses for all *B*-outputs of the 3rd order Butterworth WDF of Figure 20b have been plotted (**Note**: see the Reference Guide which syntax for ladder2WDF to use for obtaining the impulse reponses for all *B*-outputs).
In the plots that are returned by ladder2WDF (or showWDF) the maximum levels of these responses will be shown in the lower subplot (again Figure 21a).
When such large signal values can really occur within a particular environment, the WDF's internal buswidth should be adapted to prevent overflow errors.
Be warned that the other nodes inside the adaptors are not monitored here.



a)

Figure 21a and b. *Output information for the WDF of Figure 20b.*

b)



For symmetric ladder circuits, like that in Figure 20a, it is also possible to translate them into symmetric WDF structures. If also the component values are mirrored, this will result in mirrored coefficients too, which can benefit the accuracy of the structure.

The preference for a specific structure can be made available to the `ladder2WDF` function:

```
>> ladder2WDF(lpLadder,'3p_sym');

Adaptor 1:  3p serial,   p3 matched :  alpha1 =  0.33754
Adaptor 2:  3p parallel             :  alpha1 =  0.14675
                                       alpha3 =  0.14675
Adaptor 3:  3p serial,   p3 matched :  alpha1 =  0.33754
```

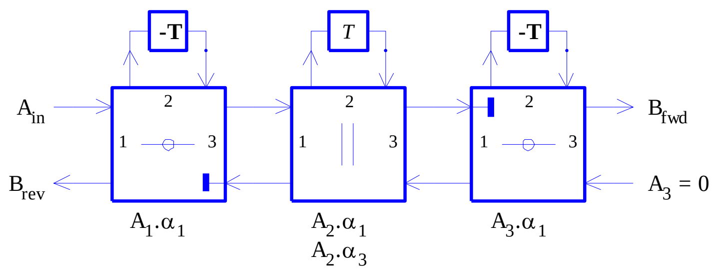


Figure 22. *Symmetrical version of the WDF of Figure 20b.*

The next structure is s translation of the band-pass ladder of Figure 10, again using 3-port adaptors.

```
>> ladder2WDF(bpLadder,'3p');
```

The parallel L-C resonators result in additional adaptors in the top row with two delay elements, one for the L and one for the C connected to port 2 and port 1 respectively.

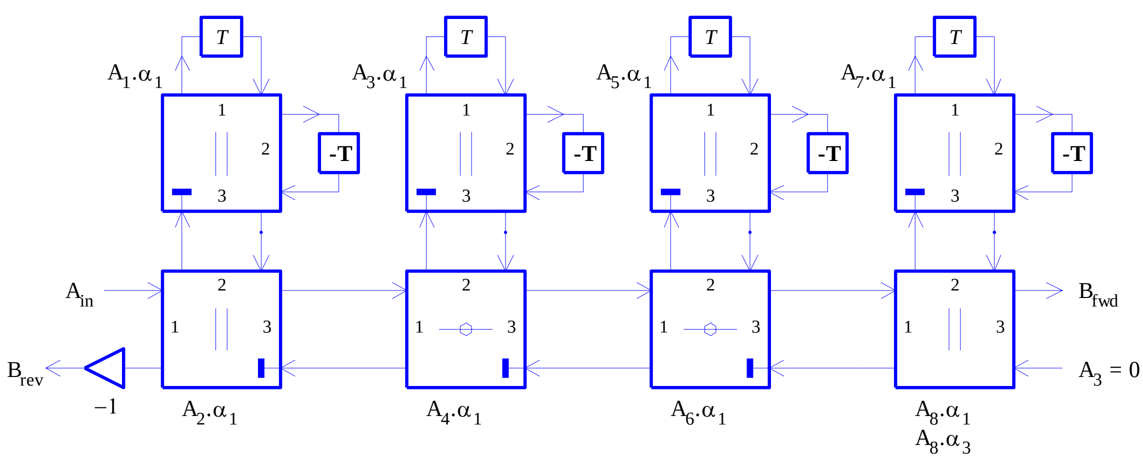


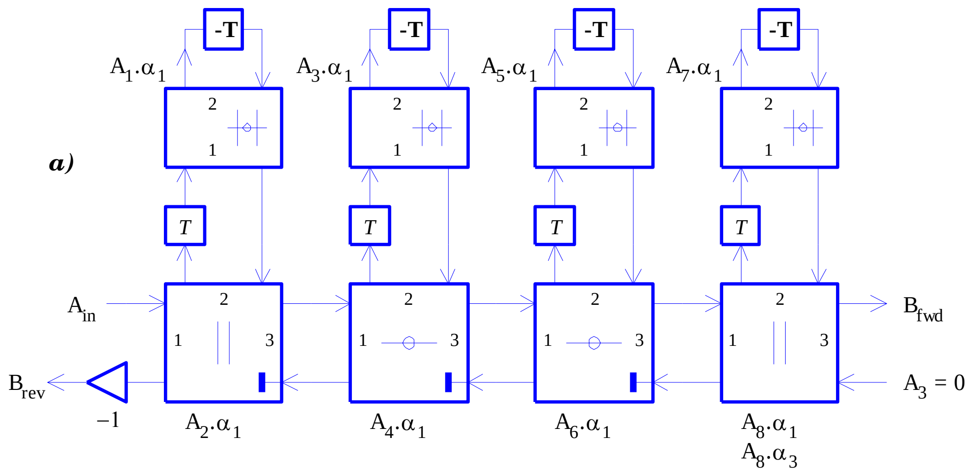
Figure 23. *WDF version of the band-pass ladder of Figure 10.*

Instead of using 3-port adaptors for implementing resonators, serial or parallel, it is also possible to use translations with 2-port adaptors. This is shown in Figure 24 that will result in exactly the same transfer functions as the structure from Figure 23, although with different intermediate maximum signal levels.

The use of 2-port adaptors can be forced with:

```
>> ladder2WDF(bpLadder,'2p');
```

a)



b)

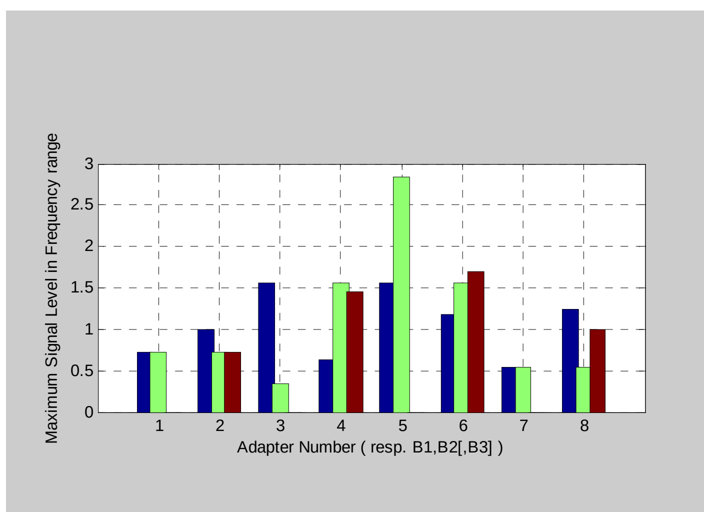


Figure 24. *WDF version of the band-pass ladder of Figure 10 with 2-port adaptors in the top row* **a)***, and* **b)** *the maximum transfer gains at each B-output of every adaptor.*

Lattice Wave Digital Filters (LWDFs).

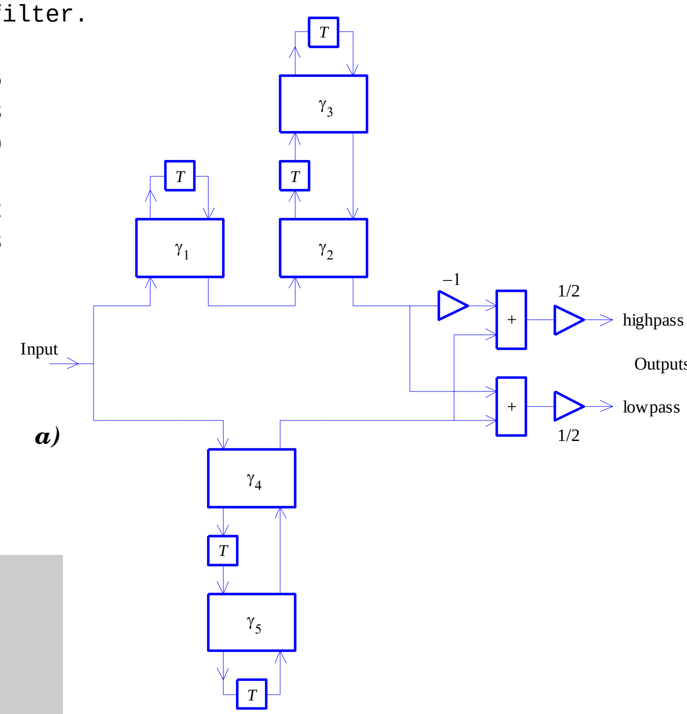
Lattice WDFs are constructed with two parallel operating all-pass functions (made up with 2-ports), i.e. transfer functions showing a gain factor of 1 for all frequencies but differing in their phase responses. By adding the two outputs we get an overall magnitude transfer function with maximum output levels of 2 at those frequencies where the output signals are in phase, and notches when the signals are exactly in anti-phase. Subtracting the signals has the complementary effect.
Now the trick is to create phase functions such that the overall magnitude transfer functions shows the desired characteristic.
It can be shown [Gazsi85] that odd order LWDFs can be used to create all odd order low-pass and high-pass transfer functions that have been describes before, while even order LWDFs result in band-pass and band-stop functions.

In the following example, we will design a high-pass LWDF:

```
>> [Hs,wp] = Hs_Vlach(5,0.1, fz2fs(0.2), fz2fs([0.29 0.35]), 0, 1);
>> Hshp = nlp2hp(Hs,1);
>> LWDF = Hs2LWDF(Hshp);
LWDF: ODD filter order, so LOW/HIGH pass filter assumed

Structure appears to be a lowpass/highpass filter.
Top row all-pass sections :
  1st degree section       :  y01 = -0.43626
  2nd degree section       :  y02 = -0.78933
                              y03 = -0.33269
Bottom row all-pass sections :
  2nd degree section       :  y04 = -0.40012
                              y05 = -0.53078

>> Hz2 = LWDF2Hz(LWDF);
>> plotHz(Hz2,1);
```



a)

b)

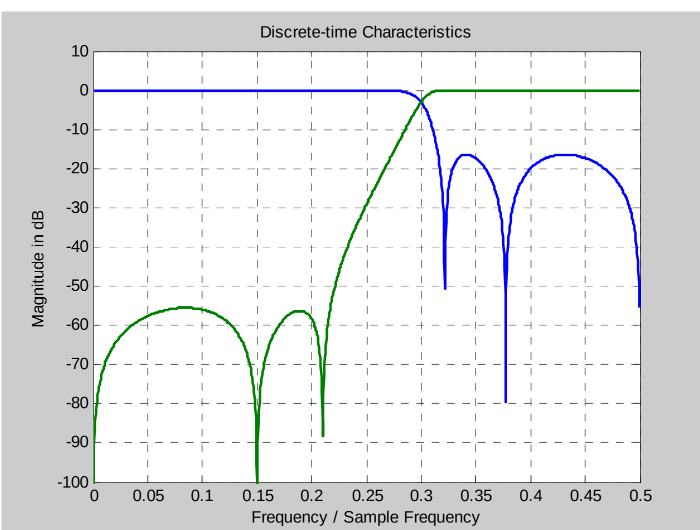


Figure 25a and b. *Lattice Wave Digital high-pass filter, structure and transfer characteristics.*

Notice that the specific frequency points were specified in the discrete-time domain directly (viz. at 0.2, 0.29 and 0.35 of the Sample Frequency), and that these point are mirrored by the low-pass to high-pass transformation to respectively $0.5 - 0.2 = 0.3$, $0.5 - 0.29 = 0.21$ and $0.5 - 0.35 = 0.15$ of the Sample Frequency.
Also, the peaks in the pass-band of the normalized low-pass function are returned in the variable `wp` by `Hs_Vlach`, so the peaks in the pass-band of the high-pass characteristic can be calculated (zoom in the real MATLAB plot to see it):

```
>> 0.5 - fs2fz(wp)
ans =
   0.5000
   0.3774
   0.3217
```

Since low-pass and high-pass outputs are complementary functions, these are exactly the relative frequencies where the low-pass characteristic shows its notches.

In Figure 26, the phase characteristics of the individual all-pass sections (top and bottom) are shown, together with the resulting absolute phase difference. All characteristic points mentioned above are clearly recognizable. Also very obvious is the relative weakness of this type of structures, viz. the very small phase differences in the stop-band. In case the coefficients are not realized accurately enough, e.g. due to quantizing effects, the stop-band cannot be guaranteed to be exactly as calculated.

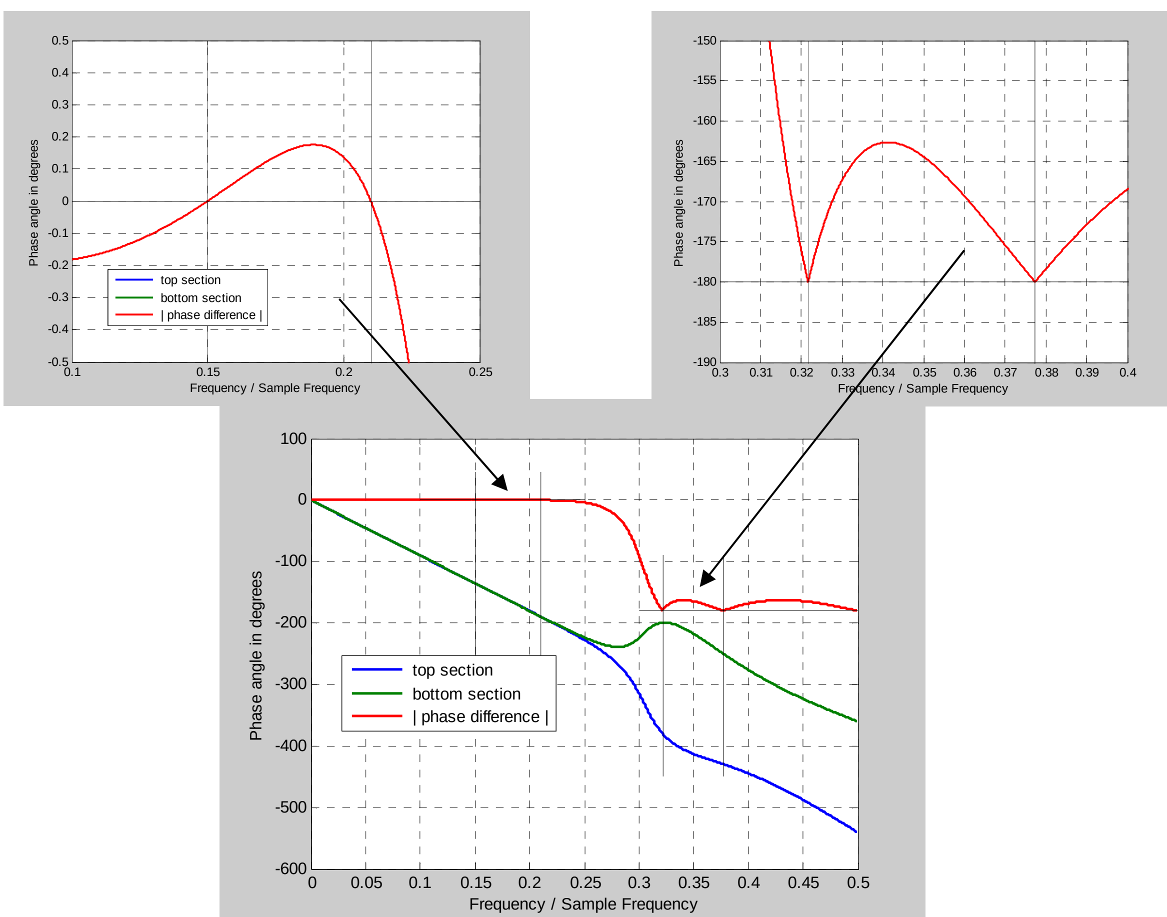
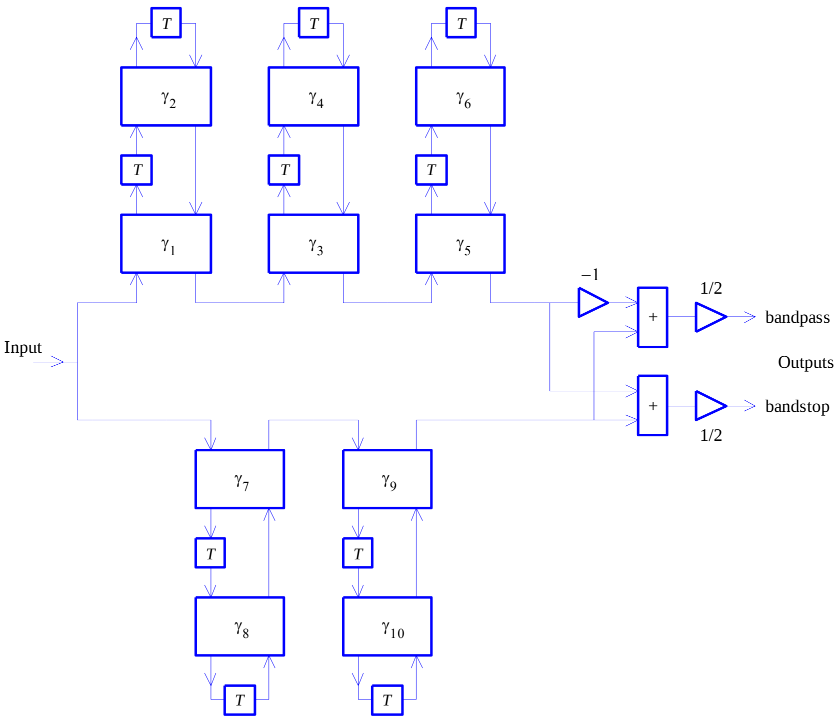


Figure 26. *Phase characteristics describing the LWDF of Figure 25.*

As said before, even order LWDFs are used to realize band-pass (band-stop) transfer functions:

```
>> Hs = nlpf('invcheby',5,45,1);
>> Hsbp = nlp2bp(Hs,fz2fs(0.15),fz2fs(0.1));
>> LWDF = Hs2LWDF(Hsbp); Hz2 = LWDF2Hz(LWDF); plotHz(Hz2,1,2);
```

a)



b)

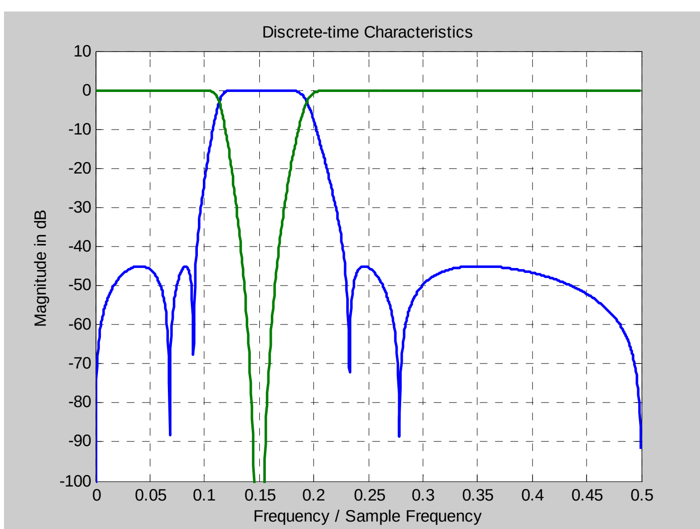


Figure 27a and b. *Lattice Wave Digital band-pass filter obtained with a normalized low-pass to band-pass transformation, structure and magnitude transfer characteristics.*

Bireciprocal LWDF designs.

A bireciprocal filter has the property that it's outputs, low-pass and high-pass, are exactly their mirror images, being reflected around ¼ of the sample frequency. The LWDF structure in such a case will be less complex then the structures that were shown before.
Not all filter types can be exactly mirrored, since there should be a distinct relationship between the behavior of the transfer function in the pass-band and that in the stop-band. In fact, only Butterworth and Cauer approximations (type 'A') will be usable. Butterworth low-pass filters with a cut-off frequency of ¼ of the sample frequency are inherently bireciprocal, for Cauer filters only one of the ripple values may be chosen while this fixes the other one.

The following example (re)calculates the "Cauer parameter (elliptical) bireciprocal low-pass filter" that has been described in detail by L. Gaszi as Example 5 in [Gaz85].
The parameters copied from Gazsi, are the filter order N=19, and the stop-band loss value of 76.89 dB.

```
>> Hs = Hs_cauer_birec(19, 76.89);
>> LWDF = Hs2LWDF( Hs(1) );
>> plotHz( LWDF2Hz(LWDF),1,2 );
```

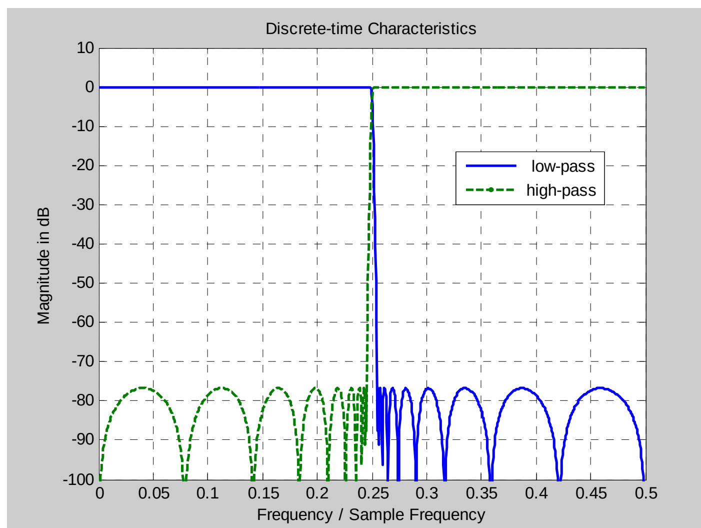
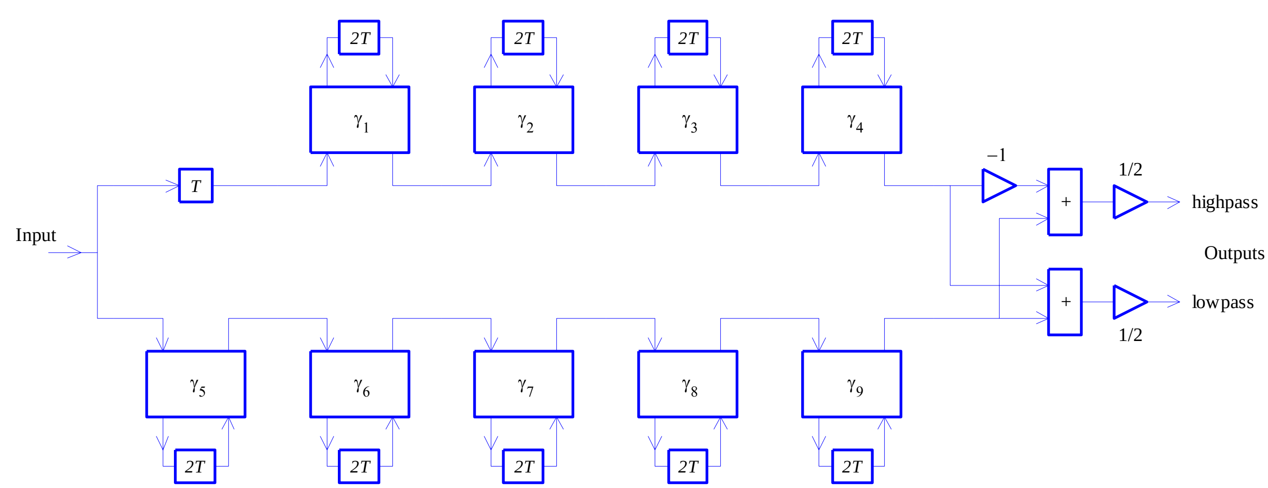


Figure 28. *Transfer characteristic and structure of the 19th order bireciprocal Cauer LWDF.*



The coefficients are calculated to be:

```
LWDF: ODD filter order, so LOW/HIGH pass filter assumed

Structure appears to be a bireciprocal lowpass/highpass filter.
Top row all-pass sections :
  single delay
  single section, 2 delays :  y01 = -0.22671
  single section, 2 delays :  y02 = -0.60341
  single section, 2 delays :  y03 = -0.84001
  single section, 2 delays :  y04 = -0.95112
Bottom row all-pass sections :
  single section, 2 delays :  y05 = -0.06417
  single section, 2 delays :  y06 = -0.42397
  single section, 2 delays :  y07 = -0.74221
  single section, 2 delays :  y08 = -0.90604
  single section, 2 delays :  y09 = -0.98481
```

Compare the results obtained here with Gazsi's Figures 14a and 14c. Gazsi's coefficients are copied below. Note that there is a difference in the numbering of the adaptors and coefficients.

Table 2.

Toolbox notation	Gazsi
y01	$\gamma_3 = -0.226\,119$
y02	$\gamma_7 = -0.602\,422$
y03	$\gamma_{11} = -0.839\,323$
y04	$\gamma_{15} = -0.950\,847$
y05	$\gamma_1 = -0.063\,978$
y06	$\gamma_5 = -0.423\,068$
y07	$\gamma_9 = -0.741\,327$
y08	$\gamma_{13} = -0.905\,567$
y09	$\gamma_{17} = -0.984\,721$

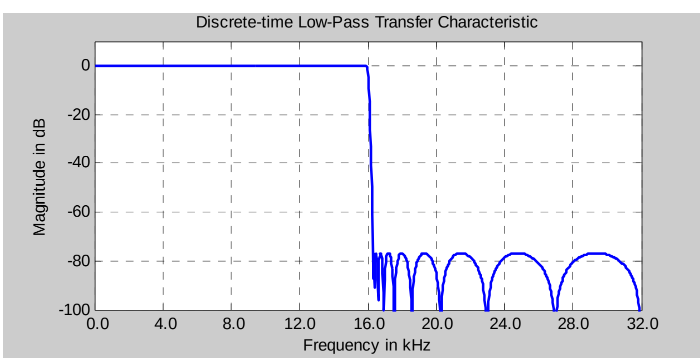


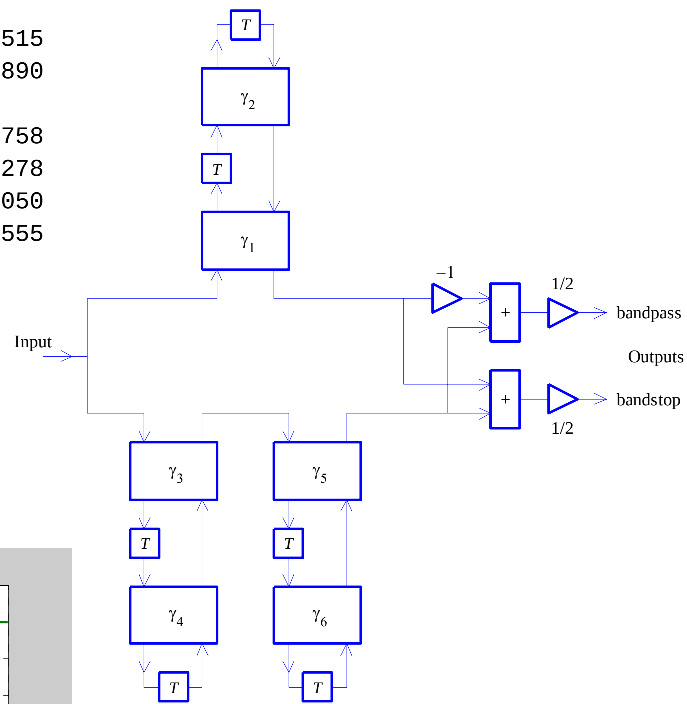
Figure 29. *Magnitude transfer characteristic of the low-pass part of the 19th order bireciprocal Cauer LWDF with the same absolute horizontal frequency scale as used by Gazsi.*

Vlach Band-pass LWDF Designs.

Of course it is also possible to create band-pass LWDFs based on the Vlach approximation:

```
>> Hs = Hs_bpVlach(6,1, fz2fs([0.13 0.17]), fz2fs([ 0 0.1 0.2]),0,1);
>> LWDF = Hs2LWDF(Hs);
LWDF: EVEN filter order, so BANDpass/stop filter assumed

Structure appears to be a bandpass/bandstop filter.
Top row all-pass sections :
  2nd degree section       :  y01 = -0.88515
                              y02 =  0.58890
Bottom row all-pass sections :
  2nd degree section       :  y03 = -0.94758
                              y04 =  0.49278
  2nd degree section       :  y05 = -0.95050
                              y06 =  0.67555
```



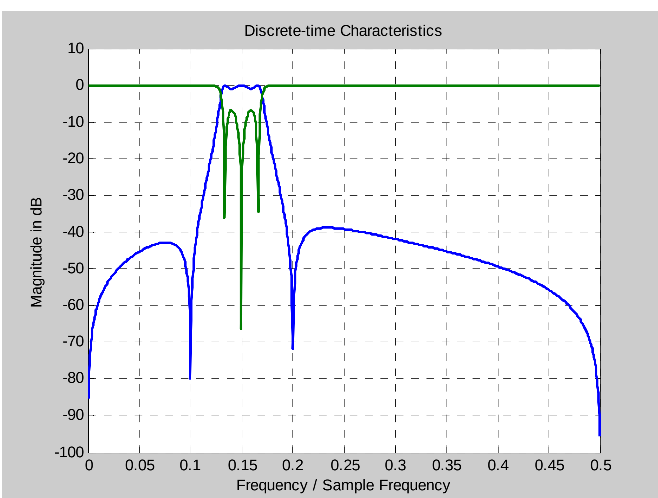


Figure 28. *Structure and Magnitude Transfer characteristics for a 6th order Vlach band-pass LWDF.*

There are two possibilities to obtain $H(z)$ for the LWDF. The first is by translating the originating $H(s)$ with `Hs2Hz`, but this will only return the forward transfer function. In the LWDF examples, therefore `LWDF2HZ` has been used, which returns a MATLAB structure with both the forward and reverse function that are reconstructed from the LWDF parameters.

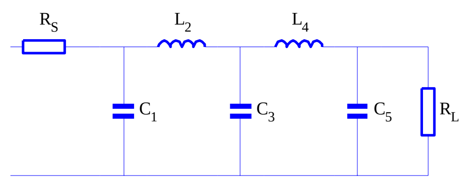
```
>> Hz1 = Hs2Hz(Hs);        % only a single Hz
>> Hz2 = LWDF2Hz(LWDF);    % Hz(1) and Hz(2), obtained from the LWDF data
```

Transmission Line Filter Designs.

A lumped element ladder circuit has to be transformed first, before it can be realized in a microwave technology. In such a high frequencies environment, it is no longer possible to work with floating inductances or capacitors as occur in series arms. A transformation is usually accomplished by shifting in Unit Elements from the input and/or the output into the circuit making use of Kuroda identities to transform a “Unit Element – series arm element” into a “Unit Element – shunt arm element” combination.
With the Vlach-approximation functions from the toolbox, the insertion of Unit Elements is straightforward and done with the aid of a binary vector. A Unit Element can be inserted to the left of every inductor or capacitor and changes an impedance function to its right describing the rest of the circuit into a reactance function and vice versa. The examples below illustrate the process. A zero means the absence, a one the presence of a Unit Element before the particular inductance or capacitance.
A strong feature of this toolbox is that the Unit Elements actually contribute to the filter’s transfer characteristic: each Unit Element increases the order of the approximation of the pass-band and adds a small amount of attenuation in the stop-band (up to 7.7 dB at high frequencies).

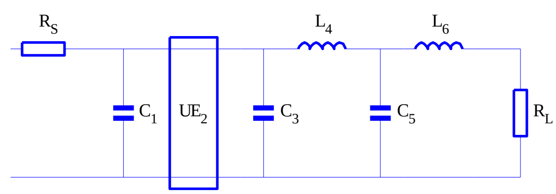
without Unit Elements:

```
>> nlp_ladder('vlach',5,1,[],[],[]);
```



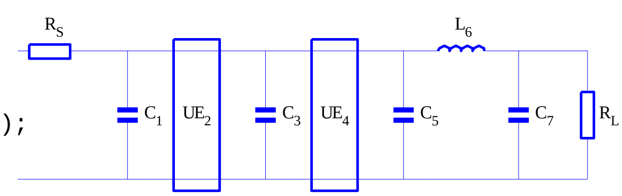
One Unit Element, before the first inductor:

```
>> nlp_ladder('vlach',5,1,[],[],[ 0 1 ]);
```



Two Unit Elements:

```
>> nlp_ladder('vlach',5,1,[],[],[ 0 1 1 0 0 0]);
```



… or, to immediately obtain the complete filter in which only shunt capacitors are present:

```
>> nlp_ladder('vlach',5,1,[],[],[ 0 1 1 1 1 0]);
```

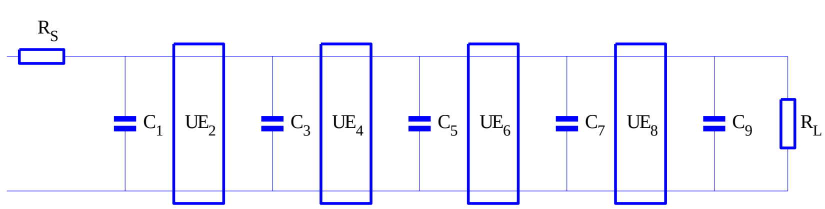


Figure 29.
Chebyshev low-pass filter with 4 additional Unit Elements..

with the following element values:

```
 Rs    1.00000 Ohm
 C01   2.17298 F   in shunt arm
UE02   1.37838 Ohm
 C03   3.07531 F   in shunt arm
UE04   1.43977 Ohm
 C05   3.14506 F   in shunt arm
UE06   1.43977 Ohm
 C07   3.07531 F   in shunt arm
UE08   1.37838 Ohm
 C09   2.17298 F   in shunt arm
 RL    1.00000 Ohm
```

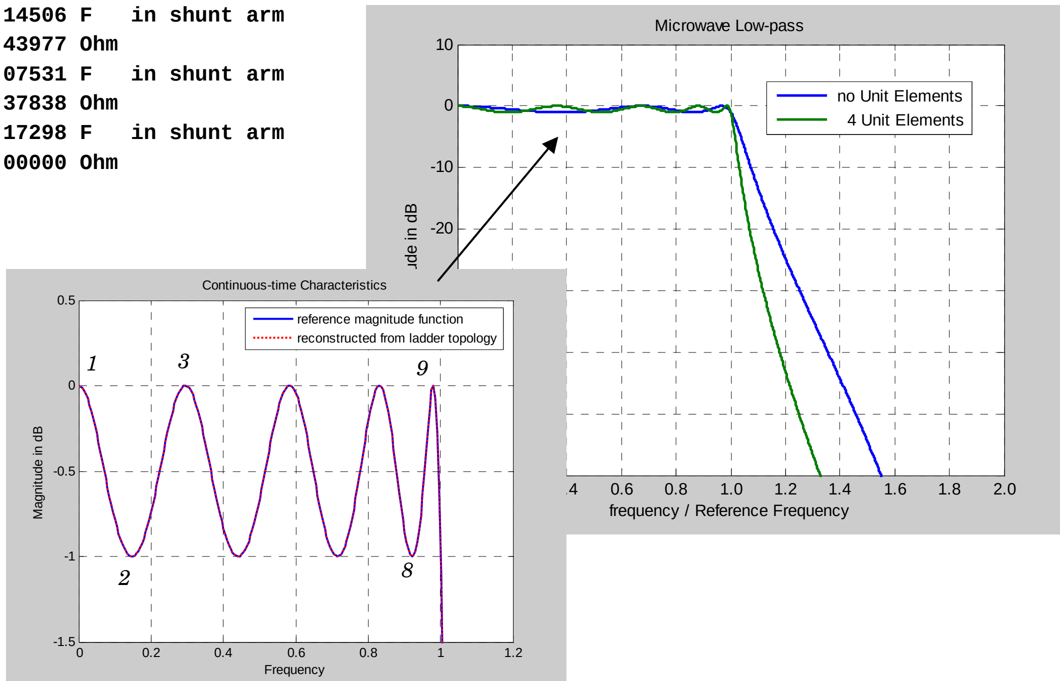


Figure 30. *Magnitude Transfer functions of the 5^{th} order low-pass Chebyshev(!) filter without and including 4 Unit Elements. The horizontal scale has been adapted to facilitate the comparison with the APLAC plots.*

The plot of Figure 30 clearly shows the increased roll-off in the transition band and the increased number of peaks and valleys in the pass-band (5^{th} order with 4 UEs = 9, equivalent to 9^{th} order). The equations describing the transfer function can be obtained with

```
>> Hs4 = nlpf('vlach',5,1,[],4);
>> Hz4 = Hs2Hz(Hs4);
```

because of the similarities in the frequency behavior of discrete-time and microwave circuits.

The correctness of the computations above can be verified with the aid of a third party program like the APLAC RF DESIGN TOOL. With APLAC it is possible to simulate transmission line filters. The element values listed above have been entered in the APLAC input format in a file `'vlach_5_1_4ue.i'` (see Figure 33), describing a transmission line filter with its cut-off frequency at 1 GHz. The resulting simulation output is given in Figure 31, and reveals a striking resemblance with what the Toolbox predicts.
Figure 31 has been created with the latest release of APLAC at the time of writing this document, viz. version 8.10 of the free downloadable Student Version (see http://www.aplac.com/).

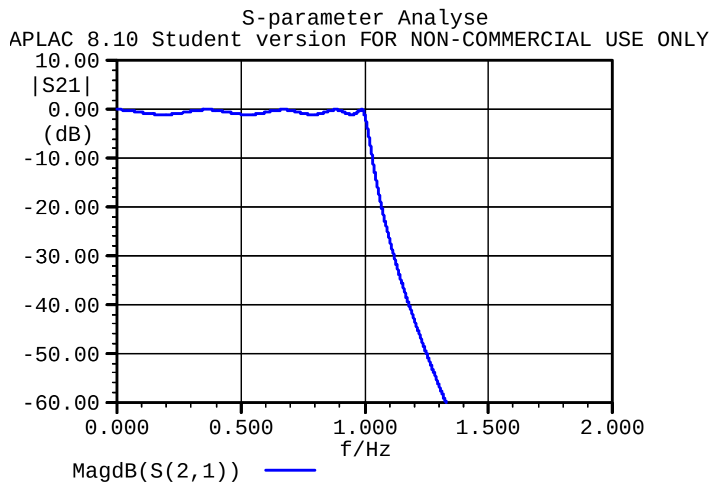
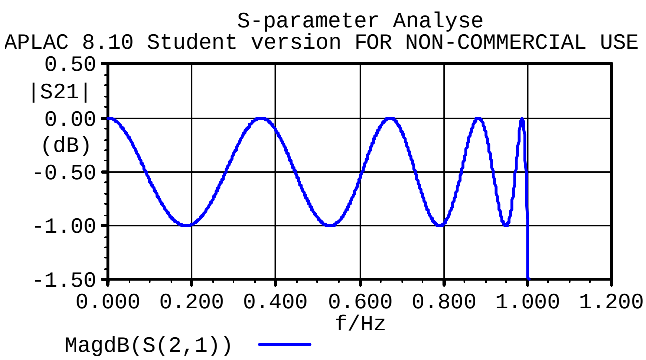
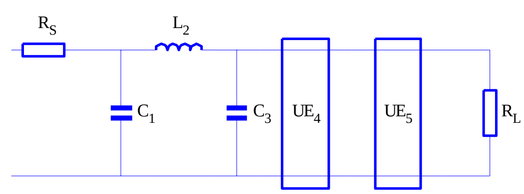


Figure 31. *Results of the APLAC .i-file listed in Figure 33 based on the values calculated for the example of Figure 29 with a cut-off frequency of 1 GHz.*
The plots use the same axes scaling as is used in Figure 30.



Although it was mentioned above that Unit Elements can be inserted to the left of the lumped elements, this is not the complete story. Access UEs are padded to the right of the ladder, and if no lumped elements should be present at all, a filter consisting of only UEs can be constructed.

```
>> nlp_ladder('vlach',3,0.1,[],[],[ 0 0 0 1 1 ]);
```




```
>> nlp_ladder('vlach',0,1,[],[],[ 1 1 1 1 1 ]);
```

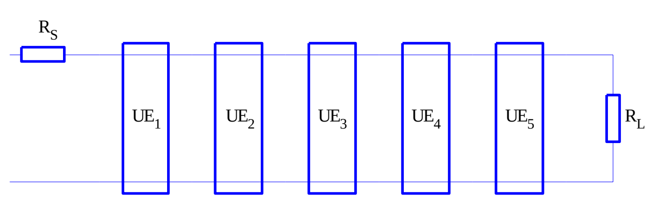


Figure 32. *Examples of Unit Element insertions.*

```
$ Filename:  vlach_5_1_4ue.i
$ see User's Guide, Figure ???
$ H.J. Lincklaen Arriens

Declare IVAR
+  C01  = 2.17298
+  RUE2 = 1.37838
+  C03  = 3.07531
+  RUE4 = 1.43977
+  C05  = 3.14506
+  RUE6 = 1.43977
+  C07  = 3.07531
+  RUE8 = 1.37838
+  C09  = 2.17298

$-----------------------------------------------------
$ Transmission lines
$ type - name - node connections - electrical length - ref freq in GHz – impedance
$-----------------------------------------------------
Tline line1  1  0  2  0  EL_LENGTH 90 FC 2  Z  1/C01
Tline line2  1  0  3  0  EL_LENGTH 90 FC 2  Z  RUE2
Tline line3  3  0  4  0  EL_LENGTH 90 FC 2  Z  1/C03
Tline line4  3  0  5  0  EL_LENGTH 90 FC 2  Z  RUE4
Tline line5  5  0  6  0  EL_LENGTH 90 FC 2  Z  1/C05
Tline line6  5  0  7  0  EL_LENGTH 90 FC 2  Z  RUE6
Tline line7  7  0  8  0  EL_LENGTH 90 FC 2  Z  1/C07
Tline line8  7  0  9  0  EL_LENGTH 90 FC 2  Z  RUE8
Tline line9  9  0 10  0  EL_LENGTH 90 FC 2  Z  1/C09

$-----------------------------------------------------
$ 2-port definition
$-----------------------------------------------------
DefNPort ideal_splane_filt 2 1 0 1.00000 9 0 1.00000

$-----------------------------------------------------
$ Output commands
$-----------------------------------------------------
Sweep "S-parameter Analyse"
+ LOOP 2000 FREQ LIN 0.0001 2
+ WINDOW 0
+ Y "|S21|" "(dB)" -60 10.0 GRID
+ WINDOW 1
+ X "f" "Hz" 0.0001 1.2
+ Y "|S21|" "(dB)" -1.5 0.5 GRID


Show
+ WINDOW 0 Y MagdB(S(2,1))
+ WINDOW 1 Y MagdB(S(2,1))
EndSweep
```

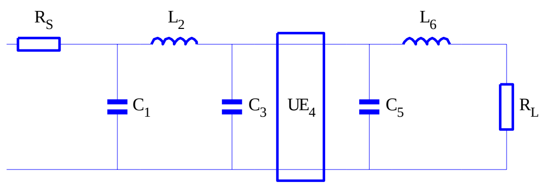
Figure 33. *Listing of a transmission line implementation of the low-pass Chebyshev(!) filter with 4 Unit Elements as shown in Figure 29.*

Although of no practical use, the strength of this toolbox function is also demonstrated with the following example in which just one Unit Element is inserted at an arbitrary location in the ladder circuit. The MATLAB result is again verified in APLAC (only a part of the .i-file is shown)

```
>> nlp_ladder('vlach',5,1,[],[],[ 0 0 0 1 0 0]);
```

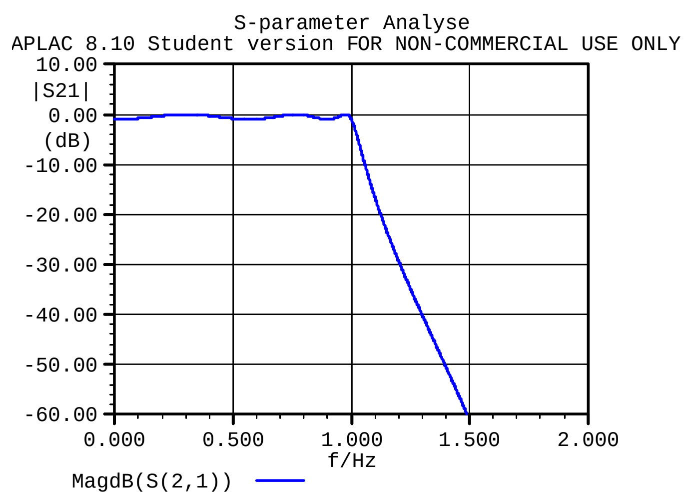
```
Declare IVAR
+ C01  = 2.14956
+ L02  = 1.10085
+ C03  = 3.04808
+ RUE4 = 1.41224
+ C05  = 2.92797
+ L06  = 0.80819
```

a)



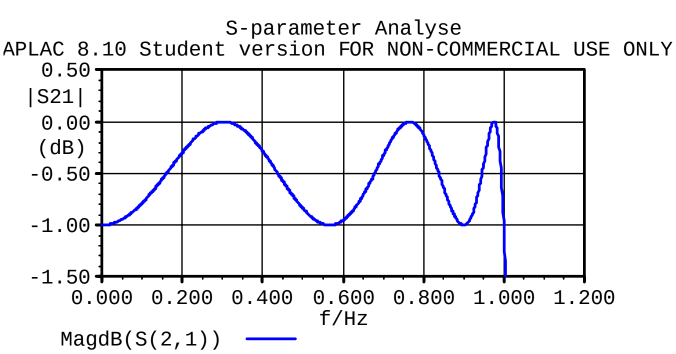

```
$--------------------------------------------------
$ Transmission lines
$--------------------------------------------------
Tline line1  1  0  2  0  EL_LENGTH 90 FC 2  Z  1/C01
Tline line2  1  4  3  3  EL_LENGTH 90 FC 2  Z  L02
Tline line3  4  0  5  0  EL_LENGTH 90 FC 2  Z  1/C03
Tline line4  4  0  6  0  EL_LENGTH 90 FC 2  Z  RUE4
Tline line5  6  0  7  0  EL_LENGTH 90 FC 2  Z  1/C05
Tline line6  6  9  8  8  EL_LENGTH 90 FC 2  Z  L06

$--------------------------------------------------
$ 2-port definition
$--------------------------------------------------
DefNPort ideal_splane_filt 2 1 0 1.00000 9 0 0.37598
```



b)

Figure 34. *APLAC plots showing the simulation of the ladder circuit of **a)** if written as a transmission line filter in **b)**.*



GUI's that cover all the above.

For an easy access of the majority of the Toolbox functions, the `wdf_GUI` (Graphical User Interface), shown below, and the more specific `bpVlach_GUI` (for LWDFs only) have been developed.
Given the examples and explanations in the previous sections, there shouldn't be any problems with the possibilities offered by and the parameters needed by the GUI's.
Parameter fields that are not needed for a particular design are grayed-out while those that are needed pop-up and need to be filled in. If a parameter may contain more than one value, like the 'Stopband Zero Frequencies' or the 'Unit Element Positions', just separate the individual values with a space (or spaces).
Output values are written to the MATLAB console window and can be used for further processing.

Specific design properties, e.g. whether a bireciprocal LWDF structure can be used, are automatically detected.

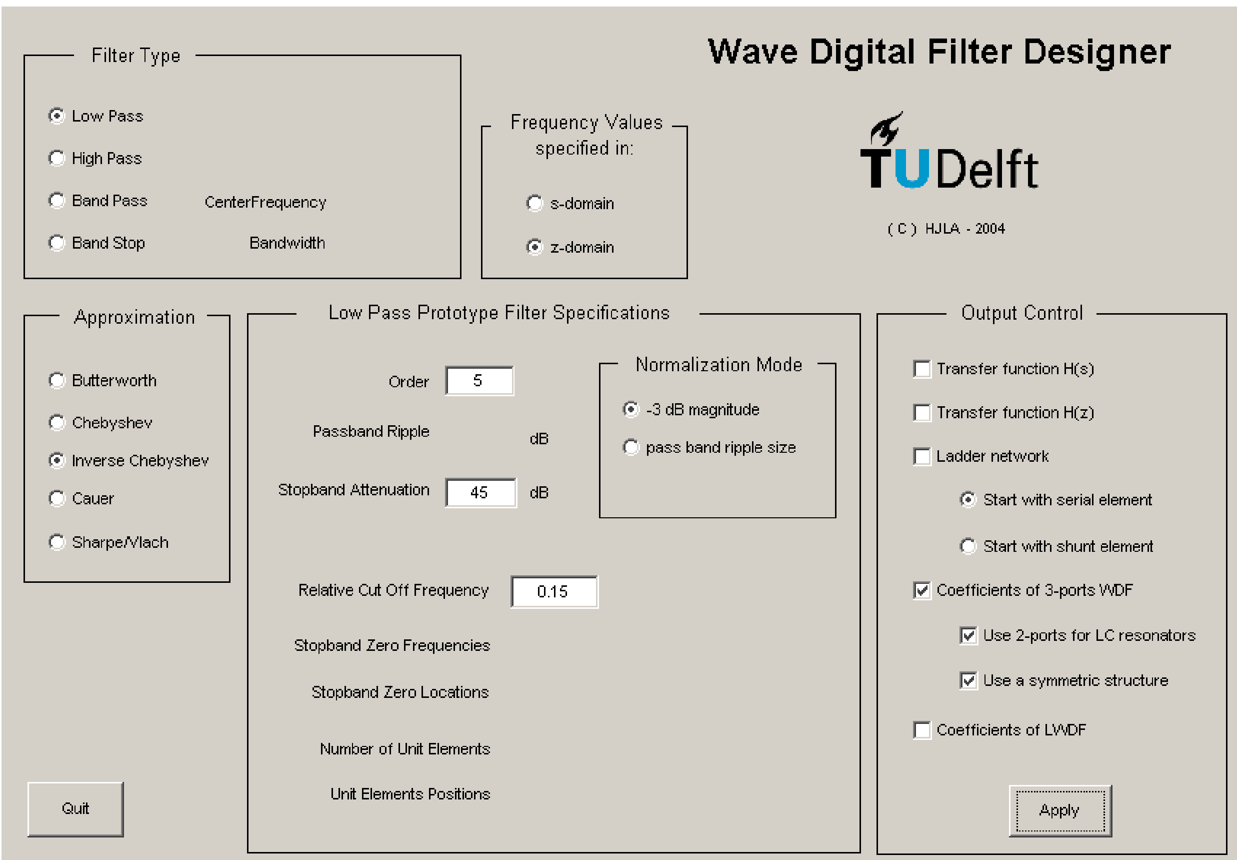


Figure 35. *Screen shot of the wdf_GUI.*

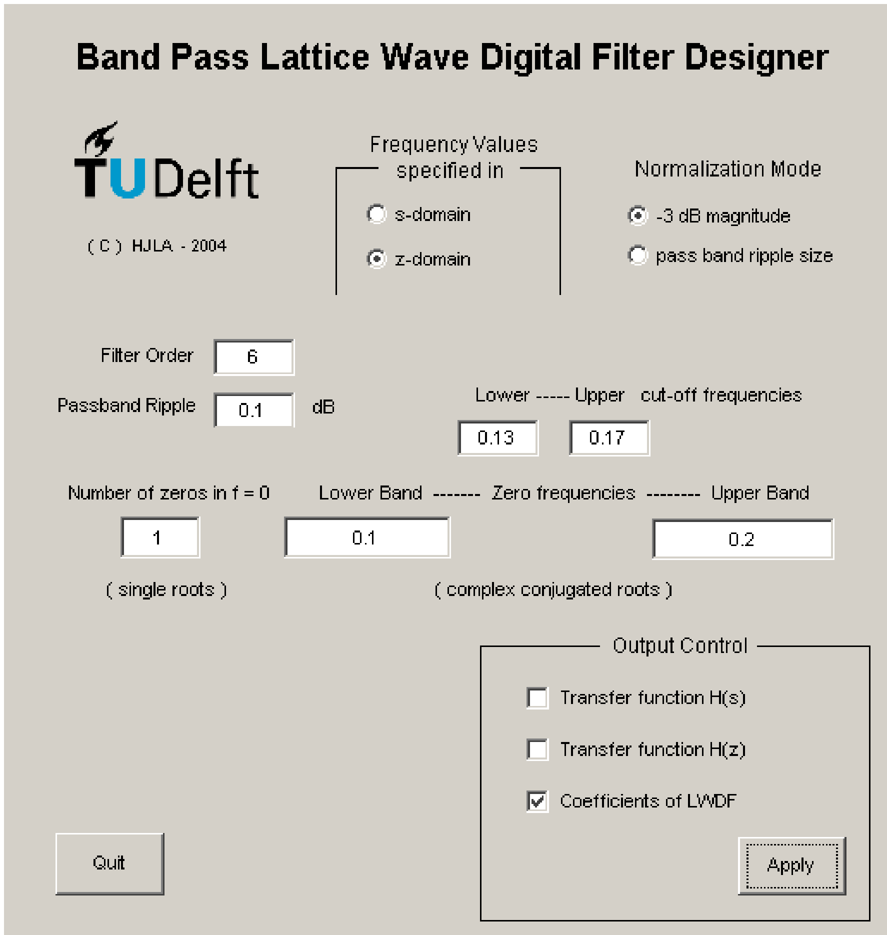


Figure 36. *Screen shot of the bpVlach_GUI for rapidly creating LWDFs based on the Vlach approximation.*

Interface to Scheduling Toolbox.

This toolbox contains two functions to translate the WDF and LWDF structures into text files in which all operations are listed in a way that it can be read by the Scheduling Toolbox, viz. `WDF2cir` and `LWDF2cir`.

Both function also return the component values in a structure that also can be recognized by the Scheduling Toolbox.

When WDFs using 2-port adaptors are to be scheduled, there is no guarantee that when the delay element between a 3-port and its corresponding 2-port adaptor is connected from port B2 of Adaptor(n) to port A1 of Adaptor(n–1) – as has been done in all WDF-figures up to now– will result in a faster circuit then when it should be connected from port B1 of Adaptor(n–1) to port A2 of Adaptor(n). Therefore, the position of the delay elements can be specified in `WDF2cir` (and in `showWDF`, see Figure 37).

```
NlpLadder = nlp_ladder('cauer',5,0.1,45,'a',1,'z');
WDF = ladder2WDF(NlpLadder,'2p_sym');
WDF2cir(WDF,'r');        % output to screen
showWDF(WDF,'r',2);      % plot in figure 2
```

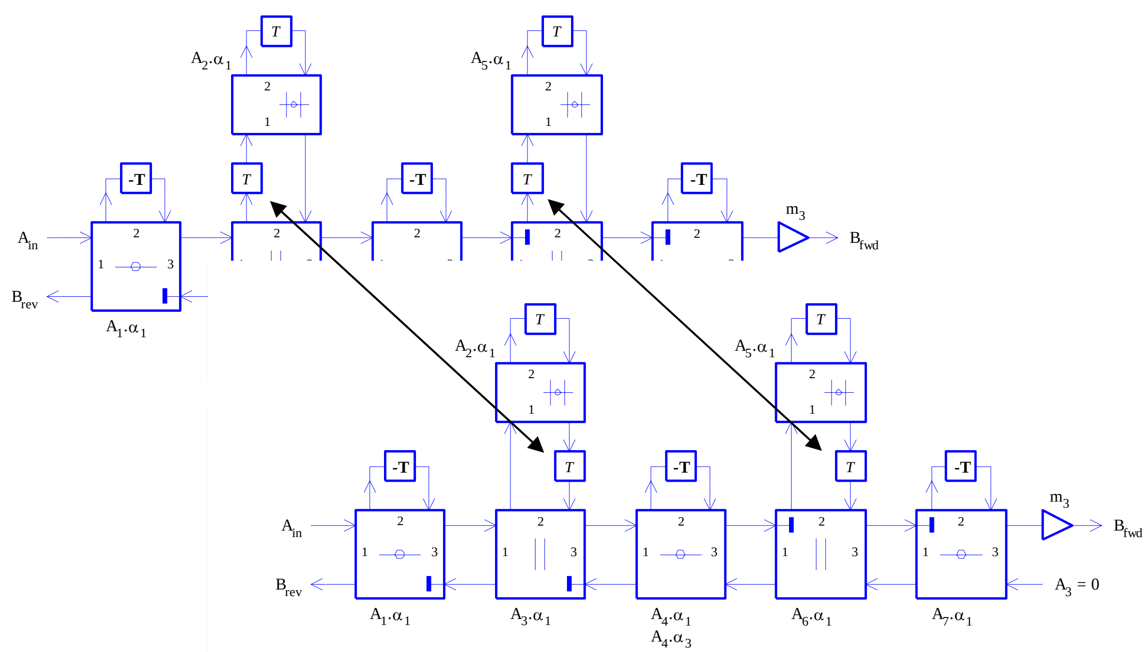


Figure 37. *The location of the delay elements connecting to 2-ports can be specified (default is the left-most connecting arm).*

In the LWDF structure, pipeline registers can be inserted easily with `LWDF_insRegs` before calling `LWDF2cir`:
supposed that the LWDF structure `LWDF` known here is the one from Figure 27a, then

```
>> LWDFp = LWDF_insRegs(LWDF,[ 1 1 ]);
>> showLWDF(LWDFp,'L')
```

will result in a structure like Figure 38 which allows more parallelism when scheduling.

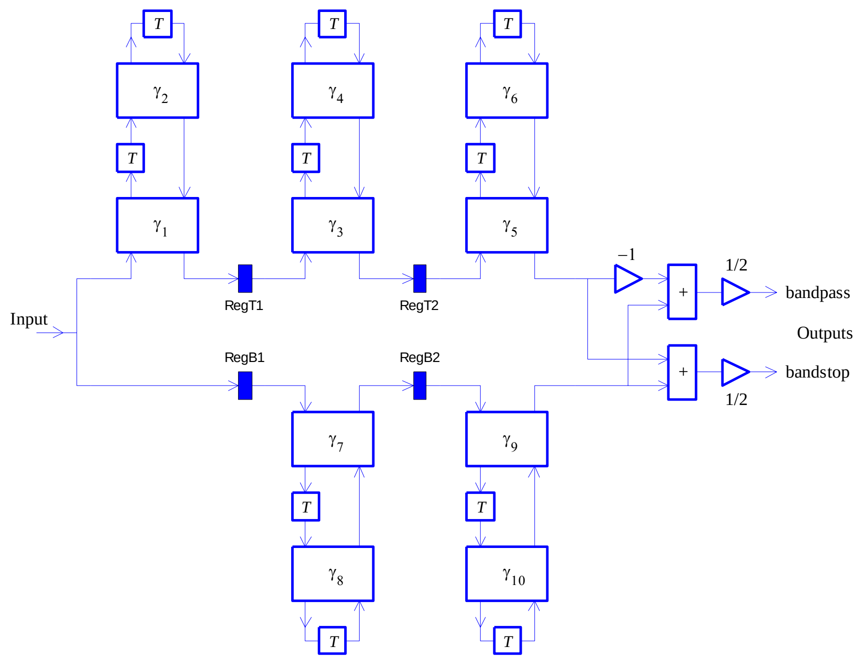


Figure 38. *Pipelined version of the LWDF of Figure 27a.*

Here it would also have been possible to position the delay elements between the adaptors in the right-most arm instead of in the left arm by using `showLWDF(LWDFp,'r')`.

Epilog.

It is by no means expected that this Toolbox should ever cover all kinds of design wishes.
When assembling this documentation –some time after the functions have actually been written– lots of omissions, improvements and extensions came to mind, but time goes on and other projects are waiting ….
Since – except for the p-code files – the functions are normal MATLAB code, everyone can tailor the code to his/hers own needs.
Intermediate results may be the starting point for implementations in specific techniques or technologies.
An automatic generation of DSP code, whether or not with intervention of the Scheduling Toolbox (which generates VHDL), is surely possible.
Also, the setup of the (MATLAB) WDF structure leaves enough possibilities to diverge from the simple ladder structures shown here.

The Toolbox has been tested on Windows XP PC's, and also with the R14 Student version on a Linux system (KDE on Slackware). As expected, the GUIs on Linux will need some visual adjustments.

Although the majority of the functions initially were written in MATLAB Release 13, some modifications and extensions were made using Release 14 features. Rewriting these parts of the code (particularly used for nested functions) to be executable with Release 13 again, including the GUIs, has not been considered and is left to the confirmed, for whatever reason, R13 user.

References.

[APLxx] APLAC, *a Powerfull and versatile analog and RF Simulator*
http://www.aplac.com/

[Abr72] "The Process of the Arithmetic-Geometric Mean."
from
Handbook of Mathematical Functions
with Formulas, Graphs, and Mathematical Tables
edited by M. Abramowitz & I.A. Stegun
Dover Publications, Inc. (1964-72)
9th edition, §17.6, pp. 571 and 598-599, 1972

[Ant79] A. Antoniou
Digital Filters: Analysis and Design
McGraw-Hill Book Company (1979)
ISBN 0-07-002117-1

[Bel68] V. Belevitch
Classical Network Theory
Holden-Day, San Francisco (1968)

[Bin64] J.A.C. Bingham
A New Method of Solving the Accuracy Problem in Filter Design
IEEE Trans. on Circuit Theory, **CT-11**,
No. 3, pp. 327-341 (September 1964)

[Chr66] E. Christian & E. Eisenmann
Filter Design Tables and Graphs
John Wiley & Sons, Inc. (1966)

[Fet86] A. Fettweis
Wave Digital Filters: Theory and Practice (Invited Paper)
Proc. of the IEEE, Vol. 74, No. 2, Februari 1986

[Gaz85] L. Gazsi
Explicit Formulas for Lattice Wave Digital Filters
IEEE Trans. on CAS, Vol. 32, pp. 68-88, Jan 1985

[Hel90] J. Helszajn
Synthesis of Lumped Element, Distributed and Planar Filters
McGraw-Hill Book Company (1990)
ISBN 0-07-707166-2

[Law90] S. Lawson & A.Mirzai
Wave Digital Filters
Ellis Horwood Ltd. (1990)
ISBN 0-13-946997-4

[Nou79] R. Nouta
Studies in Wave Digtal Filter Theory and Design
PhD Thesis, Delft University of Technology (1979)

[Rho76] J.D. Rhodes
Theory of Electrical Filters
J. Wiley & Sons Ltd. (1976)
ISBN 0-471-71806-8

[Sha54] Ch.B. Sharpe
A General Tchebycheff Rational Function
Proc. IRE, Vol. 42, pp. 454-457 (1954)

[Skw65] J.K. Skwirzynski
Design Theory and Data for Electrical Filters
Van Nostrand Company Ltd. (1965)

[Su_96] K.L. Su
Analog Filters
Chapman & Hall (1996)
ISBN 0 412 63840 1

[Tel52] B.D.H. Tellegen
A general network theorem, with applications
Philips Research Reports, Vol. 7, 1952, pp. 259-269

[Vla69] J. Vlach
Computerized Approximation and Synthesis of Linear Networks
John Wiley & Sons, Inc. (1969)
SBN 471 90870 3

[Wan99] L. Wanhammer
DSP Integrated Circuits
Academic Press, (1999)
ISBN 0-12-734530-2

[Zve67] A.I. Zverev
Handbook of Filter Synthesis
John Wiley & Sons, Inc. (1967)

This page intentionally left blank