

Filsyn

Program for the synthesis, design and analysis of electrical
filters

Version 1.70 for Windows®

2013

MANUAL

Dr. George Szentirmai

**Copyright © Dr. George Szentirmai 1975-2013
All rights reserved.**

No part of this manual or the related software may be copied or distributed in any form for any purpose without the explicit permission of the copyright holder.

No express or implied warranty is made with regard to this document or its related software. No warranty of fitfulness for a particular purpose is implied. No liability for incidental or consequential damages is assumed in connection with or arising out of the furnishing, use or performance of this document or its related software.

The name *Filsyn* is a registered trademark. All other names mentioned in this document are trademarks of their respective companies.

Programs used in the development of this software are:

Development:

- a. Intel Visual Fortran Compiler, Professional Edition**
- b. Realwin Library of Indowsway Software**
- c. Windows Interface Language of Wilson WindowWare, Inc.**
- d. Inno Setup Compiler by Jordan Russell**

Documentation:

- e. Microsoft Word**
- f. Snagit by TechSmith Corp.,**
- g. Foxit Reader**
- h. HelpScribble by JGsof**

Table of content

1. The ‘Filscript’ menu system (START menu)	
1.1 Introduction	1
1.2 Coupled resonators	4
1.3 Filters with finite zeros	12
1.4 MW (microwave) bandpass filters.	17
1.5 MW low- and highpasses	18
1.6 Miscellaneous filters	23
1.7 Time domain.	35
1.8 Other items	36
1.9 Help	38
2. Introduction to Filsyn (MAIN menu)	
2.1 Program scope	39
2.2 Starting the program	42
2.2.1 MAIN menu	42
2.2.2 New filter	46
2.2.3 New delay line	47
2.2.4 FIR filter design	48
2.2.5 Filter analysis	49
2.2.6 FIR filter analysis	49
2.2.7 Examples	50
3. Passive LC design	
3.1 Normal design	59
3.2 Placer design and lattice conversion	67
3.3 Linear phase lowpass	74
3.4 Functional input	82
Specified $F(s)$ example	83
Specified $E(s)$ example	87
3.5 Signs of the reflection coefficient zeros	89
3.6 Predistortion	91
Predistorted elliptic filter	92
3.7 Complex transmission zeros	95
Filter with complex transmission zero	96
3.8 Monotonic stopband	98
Microwave monotonic bandpass filter	98
3.9 Parametric bandpass	100
Parametric multiplexer	101
4. Passive LC/Microwave analysis	
4.1 ‘File’ menu options	104
4.2 ‘View’ menu options	105
4.3 ‘Analyze’ menu options	106
4.4 ‘Branch’ menu options	107
4.5 ‘Modify’ menu options	108

4.6 ‘Transform’ menu options	119
4.7 ‘Macro’ menu options	120
4.8 ‘Undo’ menu options	121
4.9 ‘Help’ menu options	121
4.10 Additional examples	122
5. Bandpass impedance matching	137
6. Active RC analysis	
6.1 Available structures	147
6.2 The ‘File’ menu options	151
6.3 The ‘View’ menu options	152
6.4 The ‘Analyze’ menu options	157
6.5 The ‘Modify’ menu options	157
6.6 The ‘Transform’ menu options	158
6.7 The ‘Undo’ menu option	165
6.8 The ‘Help’ menu option	165
7. IIR digital design	
7.1 Available transformations	166
7.2 Example 1: Bilinear bandpass filter	167
7.3 Example 2: Band elimination filter	169
7.4 Wave digital filters	173
7.5 Example 3: Wave digital lattice	176
7.6 Example 4: Wave digital ladder	180
7.7 Example 5: Wave digital ladder with unit elements	183
8. IIR digital analysis	
8.1 The ‘File’ menu option	185
8.2 The ‘View’ menu option	186
8.3 The ‘Analyze’ menu option	187
8.4 The ‘Implement’ menu option	189
8.5 The ‘Modify’ menu option	191
8.6 The ‘Transform’ menu option	192
8.7 The ‘Undo’ menu option	194
8.8 The ‘Help’ menu option	194
8.9 Example 1: Elliptic lowpass filter	194
8.10 Example 2: Digital bandpass filter	206
9. FIR digital filters	217
9.1 Linear phase equal-ripple design	217
9.2 Equal ripple design output	219
9.3 Example 1: Lowpass filter	219
9.4 Example 2: Band elimination filter	224
9.5 Example 3: Hilbert transformer.	226
9.6 Shaped passband	228
9.7 Windowed design	231

9.8 Example 4: Windowed bandpass	232
10. FIR digital analysis.	
10.1 Available operations	234
10.2 The 'File' menu option	235
10.3 The 'View' menu option	236
10.4 The 'Analyze' menu option	236
10.5 The 'Modify' menu option	237
10.6 The 'Transform' menu option	238
10.7 The 'Undo' menu option	240
10.8 The 'Help' menu option	240
11. Existing filters	
11.1 Passive LC filters	241
11.2 Active RC filters	244
11.3 IIR digital filters	248
11.4 Delay equalizers	252
11.5 FIR digital filters	257
11.6 Preparing a file to read	260
11.7 Switched capacitor example	261
12. Linear phase filters and delay lines	268
13. Combine postprocessor	
13.1 Introduction	293
13.2 Usage	293
13.3 Example	296
14. Edge-coupled MW postprocessor	
14.1 Introduction	305
14.2 Usage	305
14.3 Example	306
15. Optimizing preprocessor	
15.1 Introduction	314
15.2 Data input	314
15.3 Initial value selection	317
15.4 Program output	317
15.5 Definition of error	317
15.6 Optimization strategies	318
15.7 Examples	318
15.7.1 Example 1	318
15.7.2 Example 2	329
15.7.3 Example 3	336
15.8 Additional features	344
15.9 Summary	346

16. Macro files	347
Appendix A: References	353
Appendix B: Manual synthesis	
B.1 Introduction	357
B.2 Outline of the ladder synthesis procedure	357
B.3 Immittance function selection	360
B.4 Manual synthesis examples	362
B.4.1 Bandpass with quadruplet of zeros	362
B.4.2 Comments	373
B.4.3 Microwave bandpass example	373
Appendix C: Parametric filters	381
Appendix D: The ZS parameter	383
Appendix E: Reflection coefficient zeros	385
E.1 Bessel filter with Hurwitz $F(s)$	385
E.2 Default algorithm	387
E.3 User-specified $F(s)$ zero distribution	388
E.4 Modified Bessel filter	389
Appendix F: Predistortion	392
F.1 Elliptic lowpass	392
F.2 Predistorted elliptic lowpass	395
Appendix G: Functional relations	
G.1 Popular filter functions	400
G.2 Other useful relationships	405
G.3 Internal computations	408
Appl. Note 1: Diplexer and multiplexer design	
1. Introduction	409
2. Short-circuited filters	409
3. Exact constant resistance filter pairs	410
4. Approximately constant resistance filter pair	411
5. Example	413
6. Multiplexers	419
7. Summary	421
Appl. Note 2: Use of complex transmission zeros	
1. Introduction	422
2. Original lowpass	422
3. Combined design	427

Appl. Note 3: Lower sideband crystal filter	
1. Introduction	430
2. Lower sideband filter	430
Appl. Note 4: Narrow-band band-reject filters	
1. Introduction	441
2. Starting ladder design	441
Appl. Note 5: Cascaded lattice crystal filter	
1. Introduction	450
2. Filter specification	450
3. Additional comments	462
Appl. Note 6: Delay equalization of IIR and MW highpass filters	
1. Introduction	463
2. Digital example	463
3. Microwave filter case	472
Appl. Note 7: Linear-phase bandpass filters	
1. Introduction	481
2. Design example	481
3. Equal-minima design	485
4. Specified stopband	487
5. 3 dB bandwidth	490
Appl. Note 8: Active RC filter using biquads	
1. Introduction	494
2. Specification	494
3. Cascade design	497
4. Leapfrog design	503
5. Overall structure	507
Appl. Note 9: Design of switched-capacitor filters	
1. Introduction	511
2. Leapfrog bandpass filter	511
3. Cascaded design	519
4. Summary	527
Appl. Note 10: Lowpass-like bandpass filters	
1. Introduction	528
2. Impedance matching	528
3. Equal terminations	528
4. Example	529
5. Maximally-flat passband example	532
Appl. Note 11: Single sideband crystal filters	
1. Introduction	535

2. USB filter	535
3. LSB filter	538
Appl. Note 12: Minimum-phase FIR filters	
1. Introduction	544
2. Example	544
Appl. Note 13: Coupled shunt resonators	
1. Introduction	550
2. Example 1	550
3. Example 2	556
4. Example 3	561
5. Comments	570
Appl. Note 14: Active RC filter using FDNR	
1. Introduction	572
2. Filter design	572
Appl. Note 15: Microwave or digital low- or high-pass filters with flat delay and loss	
1. Introduction	582
2. Example 1	583
3. Example 2	587
Appl. Note 16: Analog lowpass filters with flat delay and loss	
1. Introduction	591
2. Data input	591
3. Comments	596
Index	597

THE "FILSCRIPT" MENU SYSTEM

1.1 Introduction

The **filscript.exe** executable file is a front end program for the **filesyn.exe** main **Filsyn** program and it performs two classes of functions:

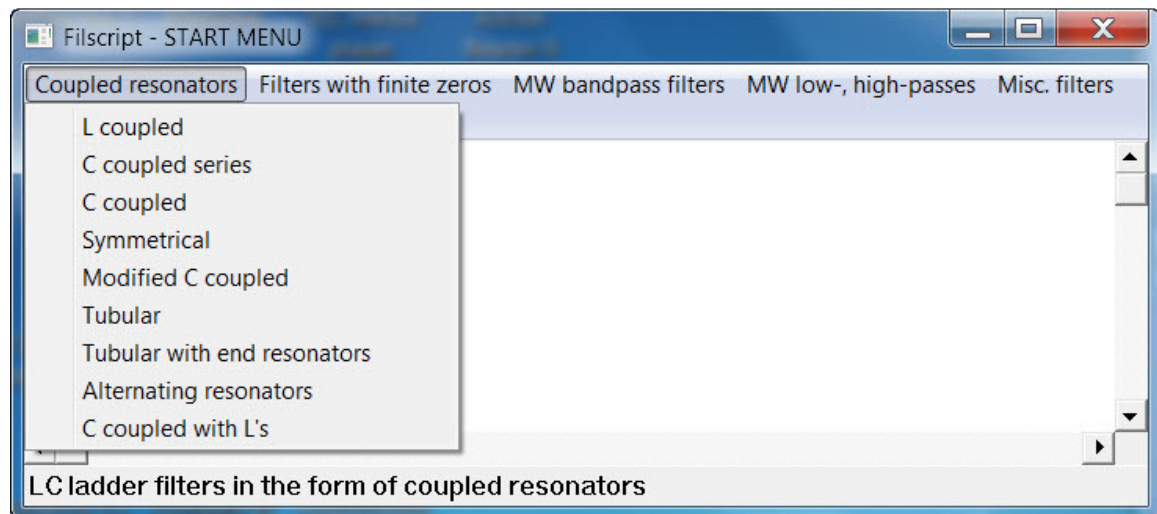
First, it automates a number of (mainly) passive LC filter designs that could be performed directly (manually), but this front end makes their design substantially easier. Second, it either stores or generates transfer functions that are not available in the main **Filsyn** program and forwards them to the main program for the synthesis of the filter.

This program was written in the Windows Interface Language of Wilson WindowWare Inc., which can be used to generate further script files for the automation of other specific designs.

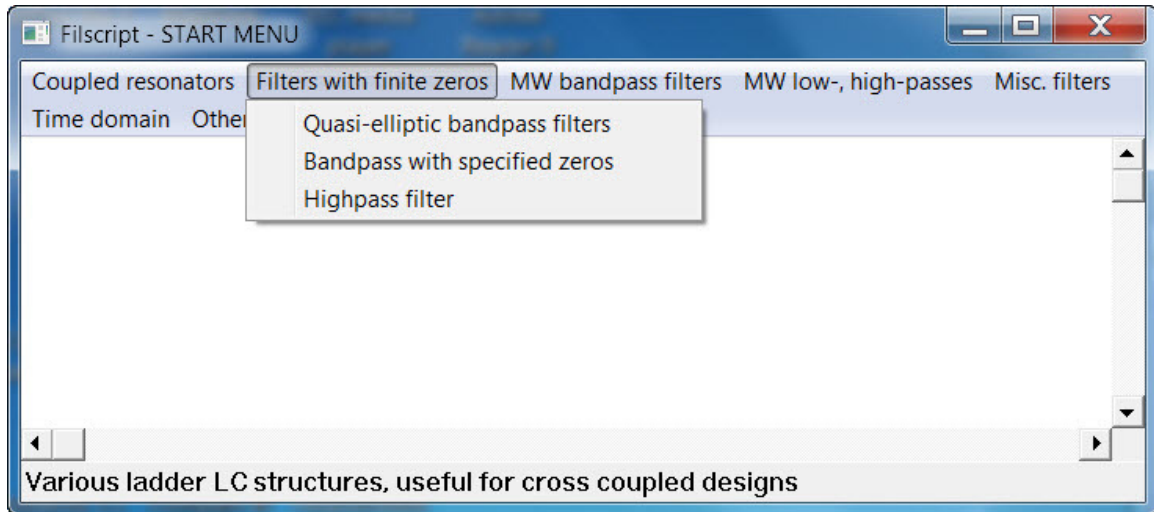
Upon calling **filscript.exe**, we first encounter a menu system, where we can select the type of design we wish to perform and subsequently enter the numerical and other data that specifies the particular filter.

This START MENU groups the options into the following categories:

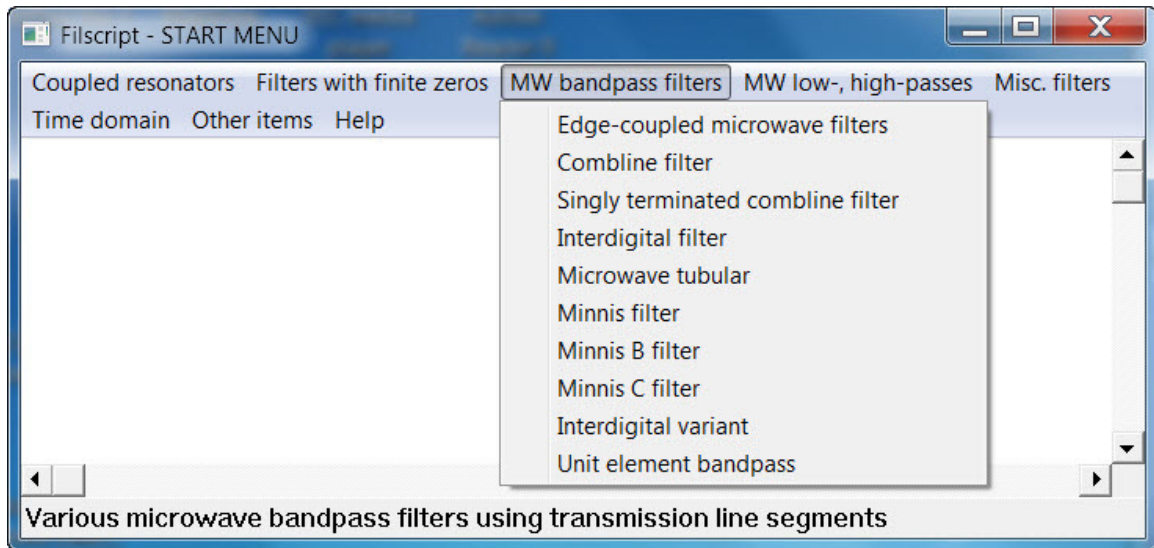
- a) Coupled resonators. These are passive LC bandpass filters containing (mostly parallel) resonators coupled by single inductors or capacitors.



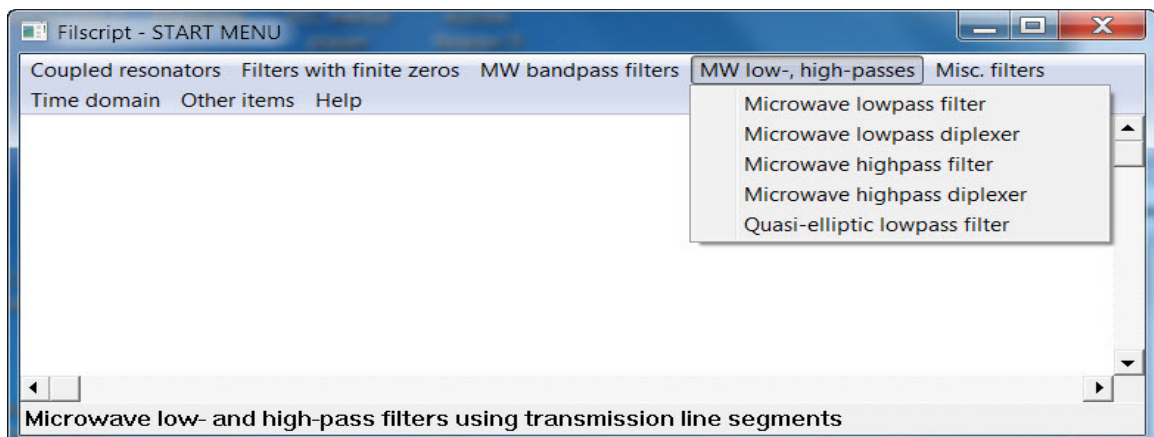
- b) Designs using the **Placer** routine to generate optimal highpass and bandpass LC filters, this time containing finite transmission zeros. The bandpass filters again have a structure of coupled resonators, but this time the coupling branches can be resonators themselves or couple non-adjacent resonators.



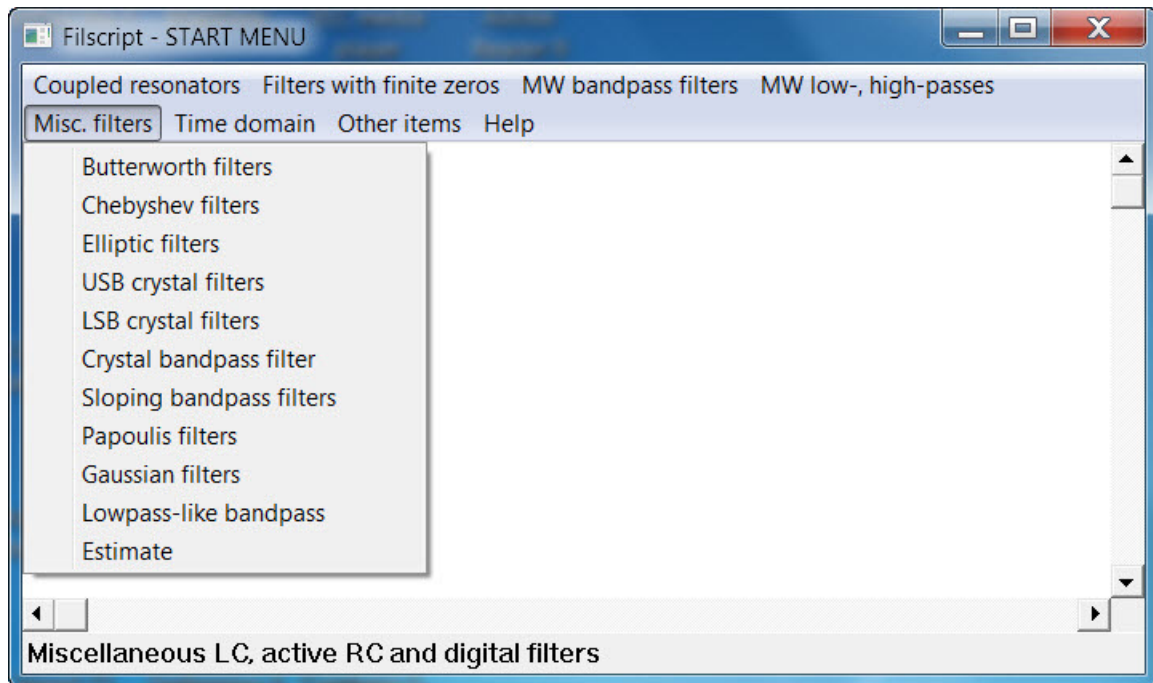
c) Microwave bandpass filters of various kinds.



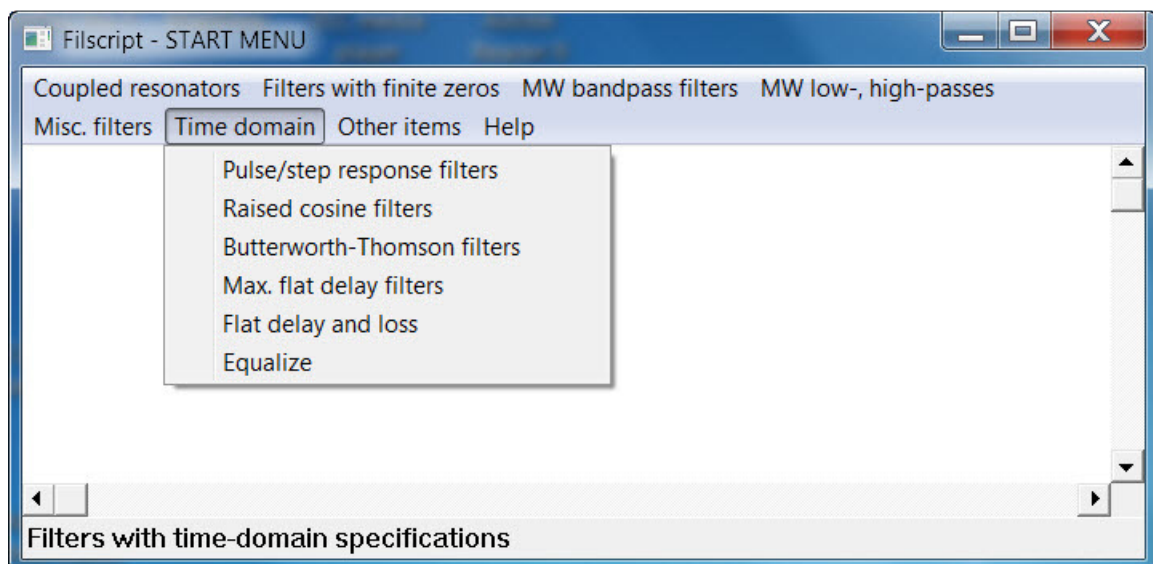
d) Microwave low- and high-pass filters of various kinds



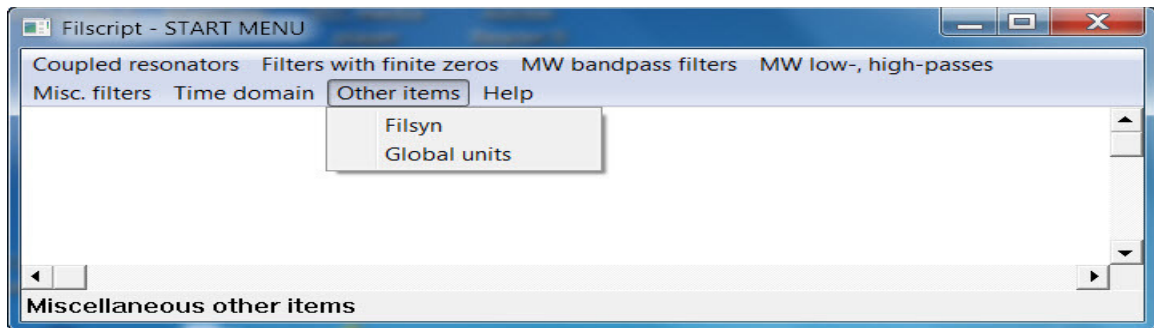
- e) Miscellaneous filters, like crystal, Gaussian, sloping passband and others.



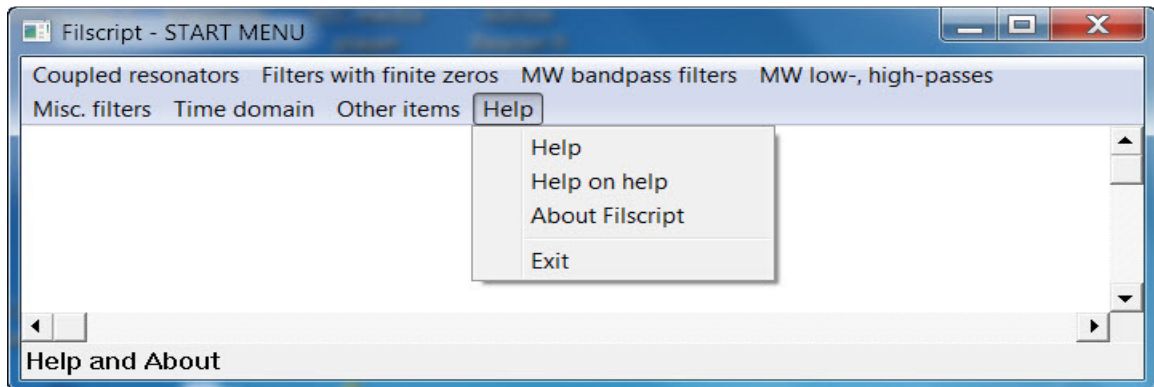
- f) Time domain designs, which include specified pulse- or step-response designs, Butterworth-Thomson filters and others.



- g) Under the header **Other items** we can call the main **Filsyn** program directly, or select other than the default global units:



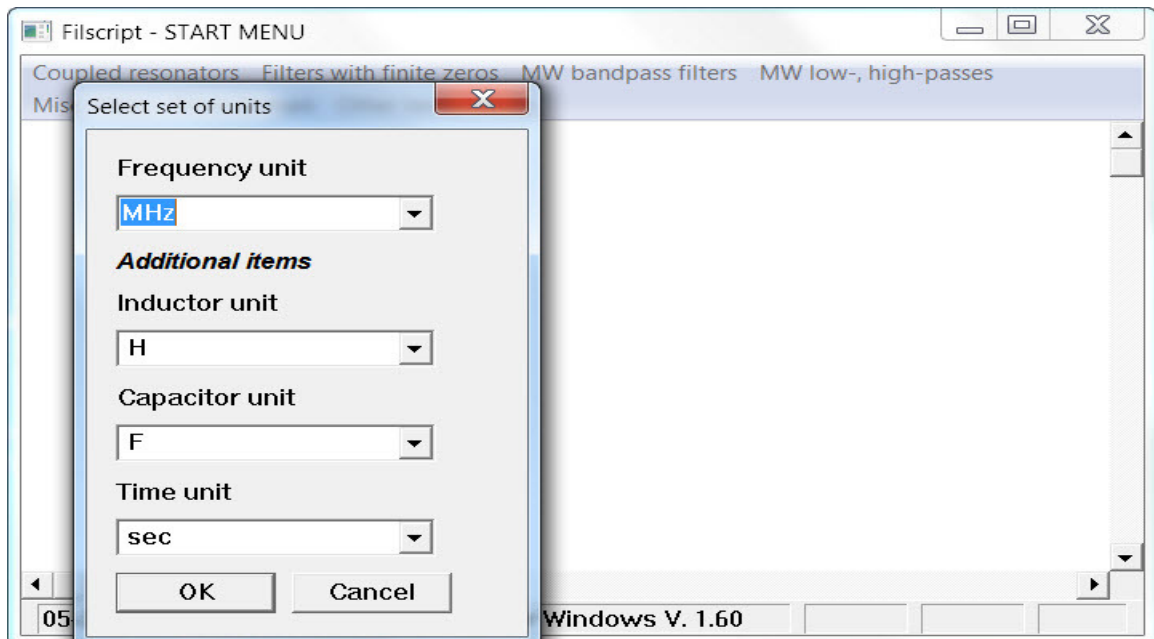
h) Finally under the header **Help** we find the usual items as well as the **Exit**:



Consider now the individual filter types available.

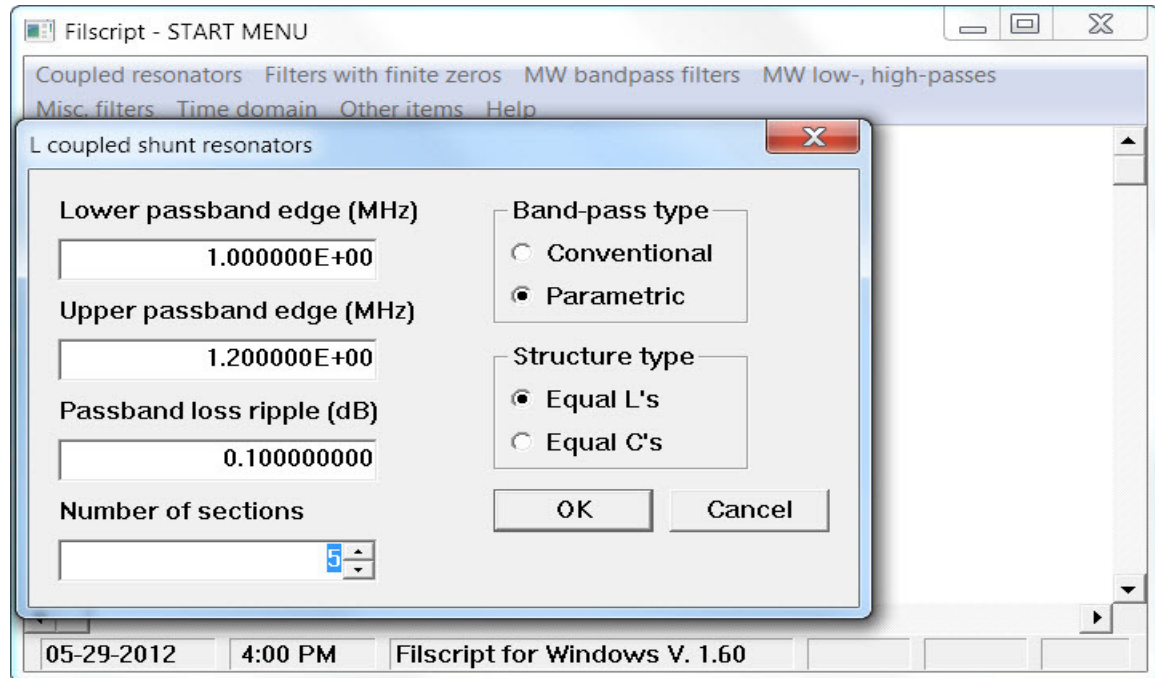
1.2 Coupled resonators

First we use the **Global units** option to make MHz to be the frequency unit of choice:

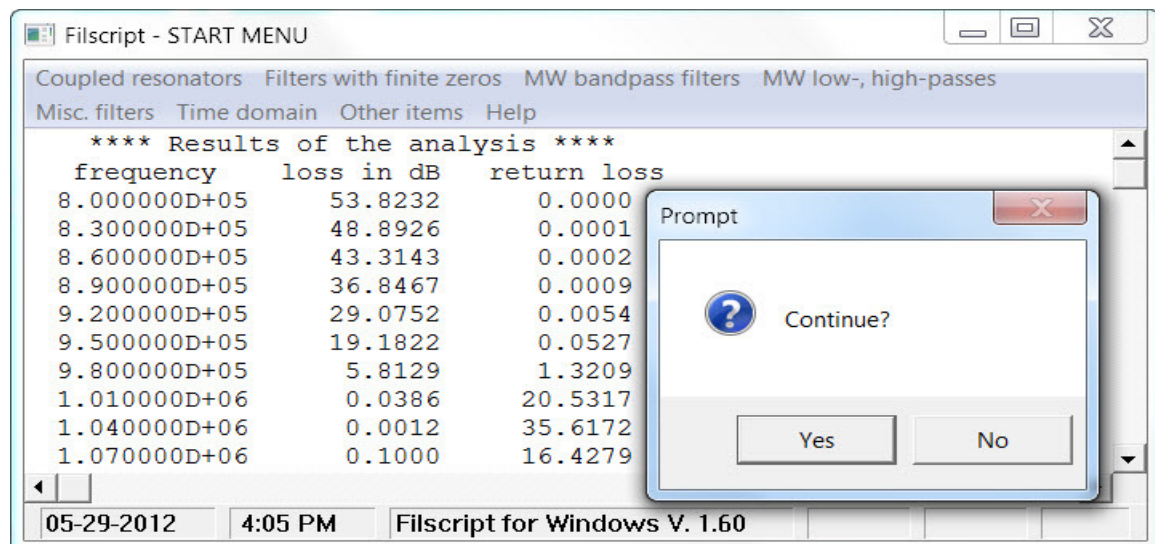


Under **Coupled resonators** the first type, **L coupled**, gives us inductively coupled shunt resonators, where all shunt inductors or capacitors are equal.

Clicking on this option brings up the data entry window:

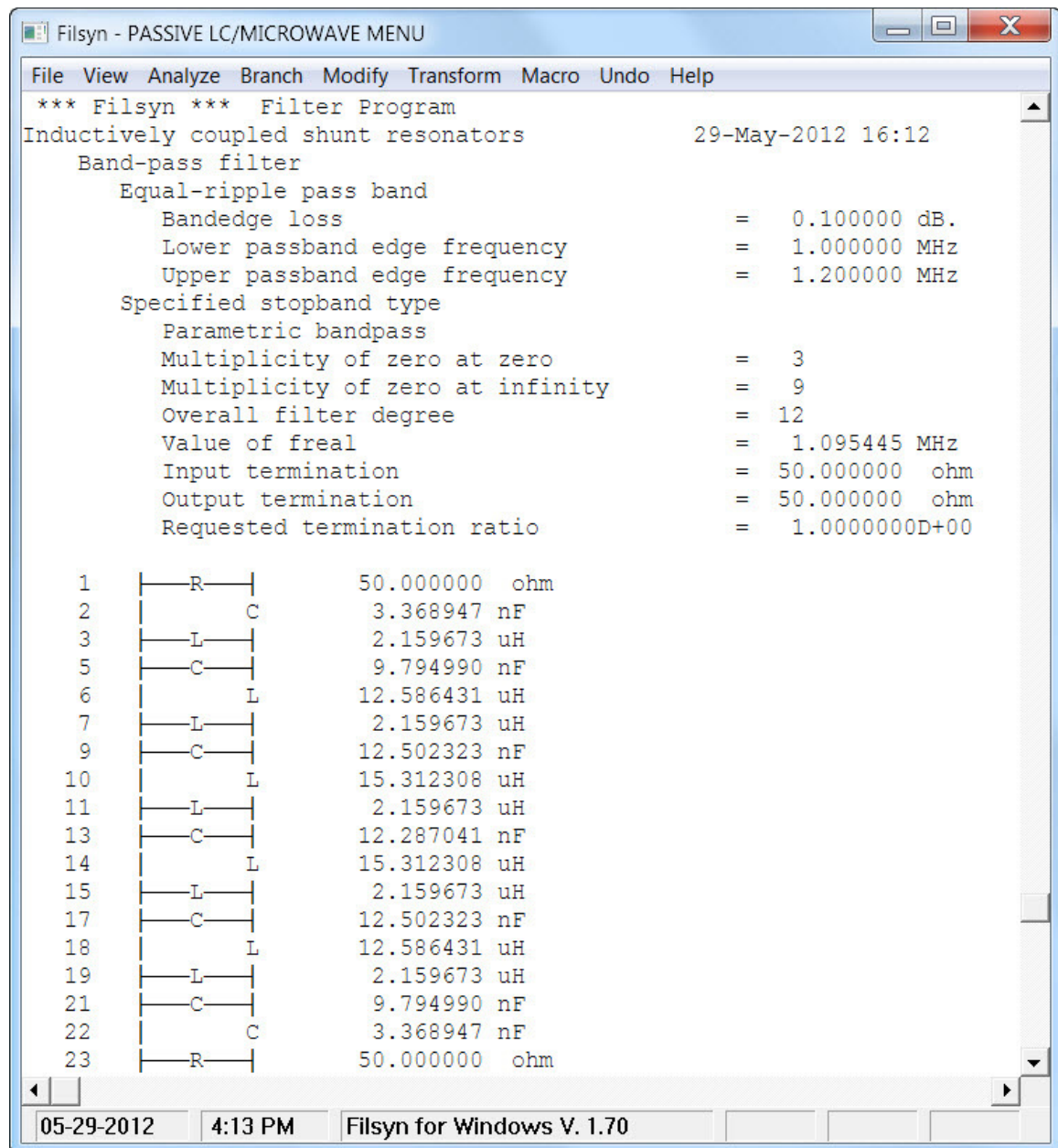


Here we specified a bandpass filter with parametric behavior and all equal shunt inductors. The number of sections (resonators) must be between 2 and 20. Clicking on OK, brings up the analysis entry window, which is needed since the loss behavior of the filter would not be known otherwise until later. Note also, that we have previously set the global frequency unit to MHz. We specified a set of frequencies and the results are shown when we click on the OK button. The columns are the frequency in Hz, the loss and the return loss, both in dB.

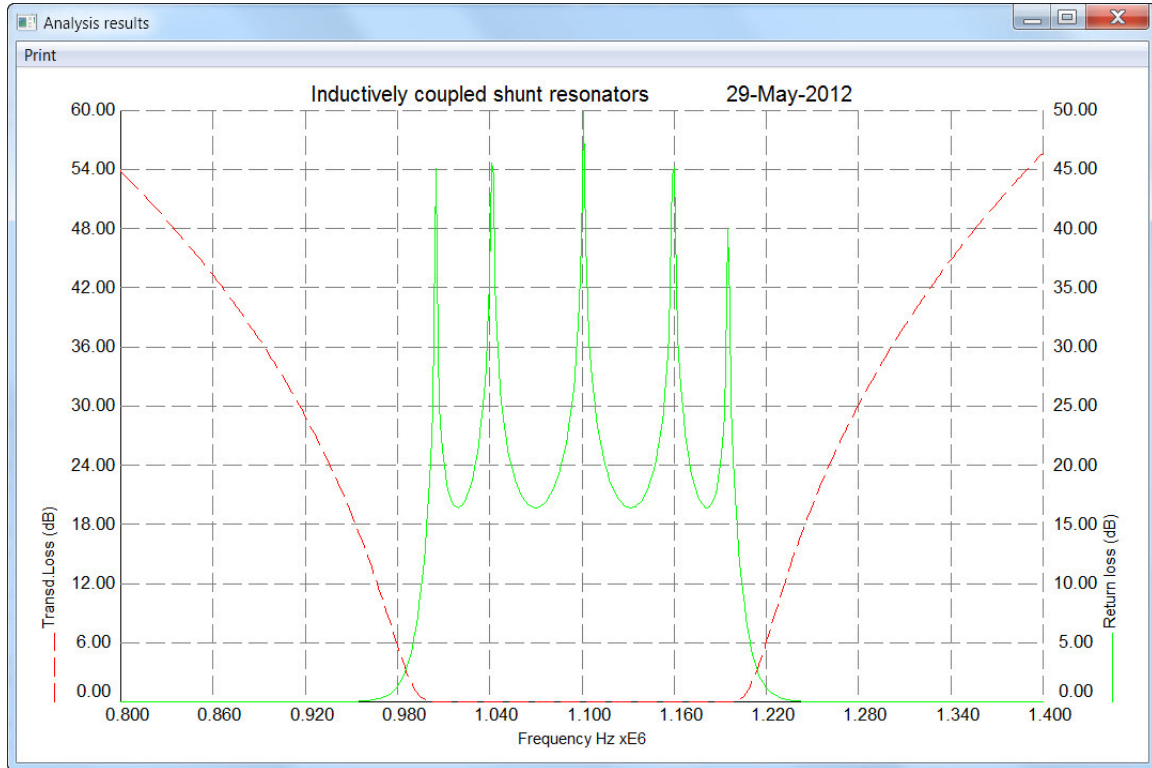


Note that while specifying the analysis frequencies uses the MHz unit but not the display.

There is a **Continue?** prompt that can be moved out of the way and the tabulation scrolled if necessary. Clicking on **Yes**, the program now calls the **filsyn.exe** executable, enters all the necessary information into its entry screen and performs the synthesis. This is followed by a number of ladder manipulation steps to get the circuit into the proper form. This process may take a minute or two and the program finally stops where the display is as shown (we have declined the offered option to increase the internal impedance level):



At this point we are in the passive LC analysis part of the program and can perform any operation we wish. The available menu options are described in detail later in this manual. The only thing we wish to show here is the results of a frequency domain analysis in graphical form to demonstrate the filter performance:

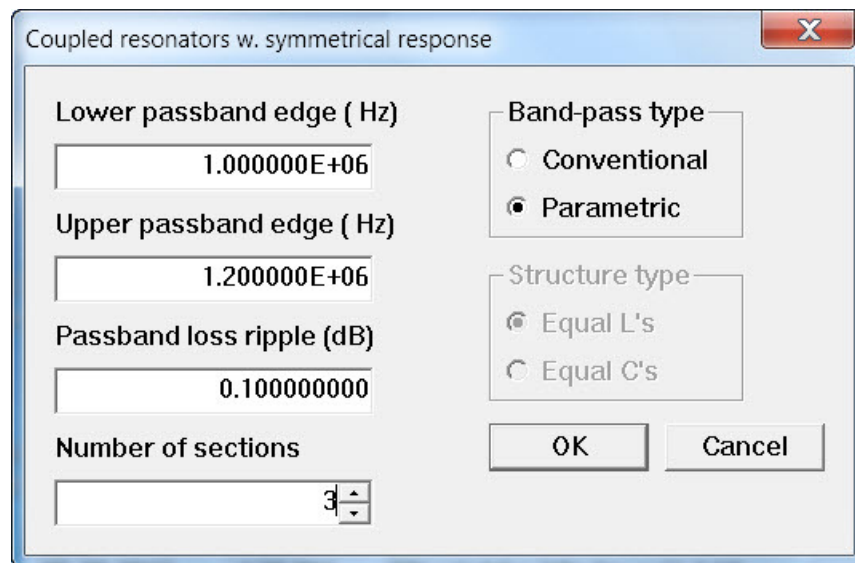


The next option, **C coupled series**, yields series LC resonators coupled by capacitors to ground. The data entry screen is nearly identical except here the equal L option is the only one available and the maximum number of sections is limited to 15.

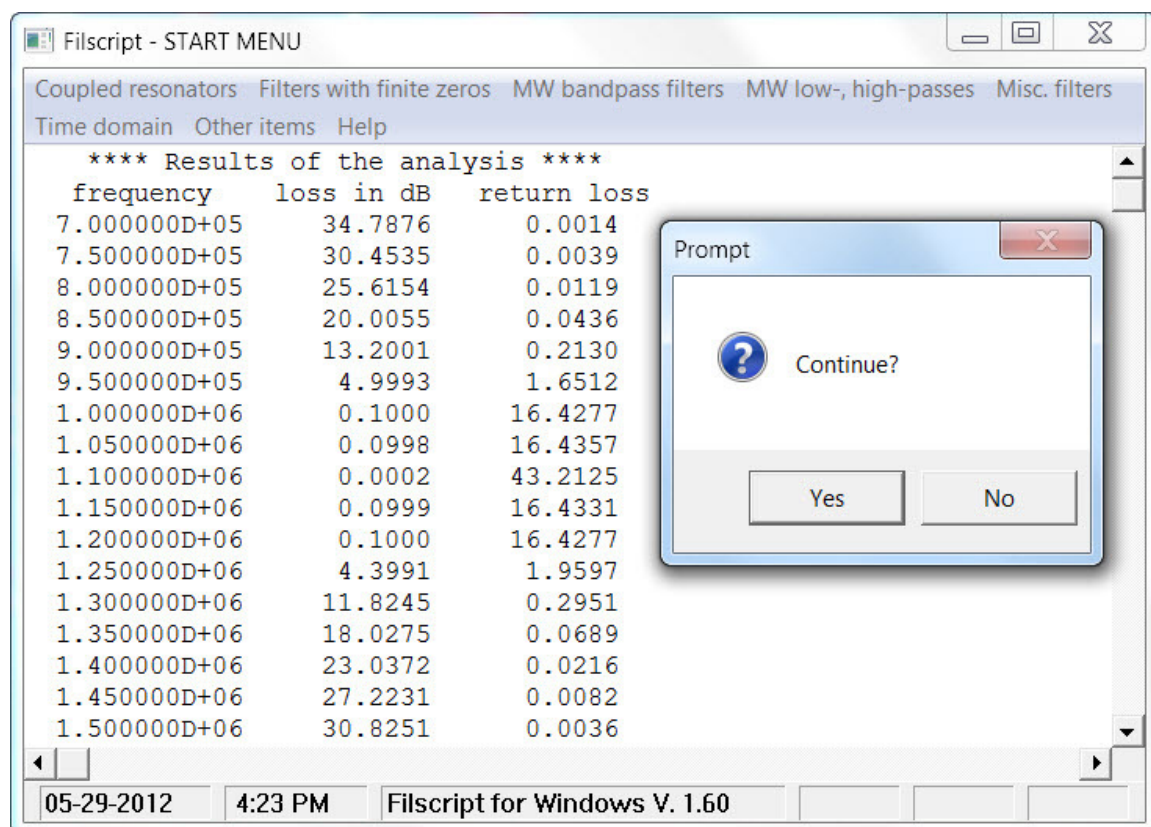
Next we can get capacitively coupled shunt resonators, **C coupled**, again all shunt inductors or all shunt capacitors being equal at the end of the design. The data entry screen and all its options are identical to that of the inductively coupled case treated above. Number of resonators is limited to a maximum of 20.

The next option, **Symmetrical**, offers designs with mixed inductive and capacitive couplings in order to achieve a frequency response that is as close to arithmetic symmetry as possible. Only the equal inductors case is available and the number of resonators is limited to 12. The data entry screen and its options are very similar to the other cases mentioned above.

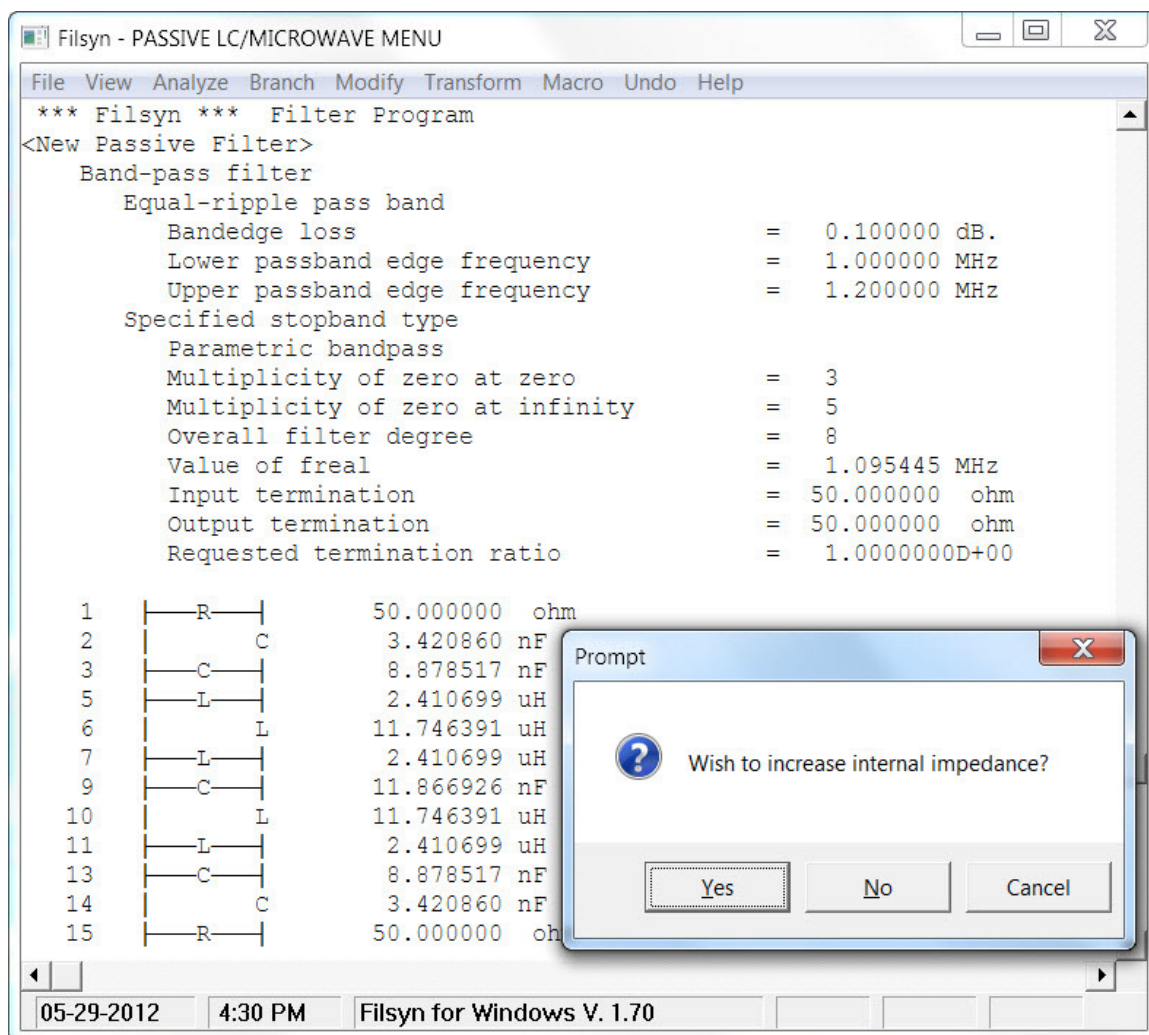
An example of this type starts with the data entry screen, where the global frequency unit was reset to Hz:



We analyze this design to see what the performance is:

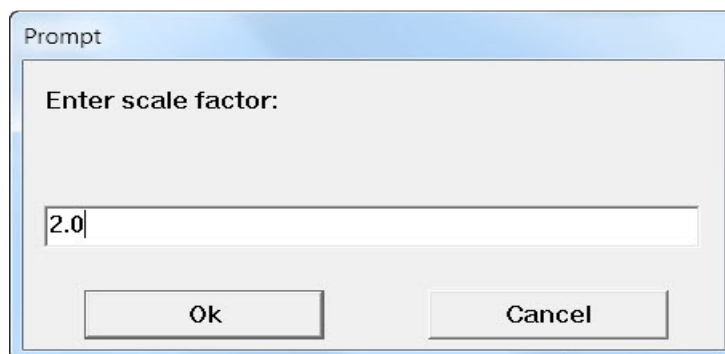


Clicking on the **YES** button, the program goes through a large number of operations, since it is using the manual synthesis, to be described later, followed by some additional circuit manipulation and ending up here:

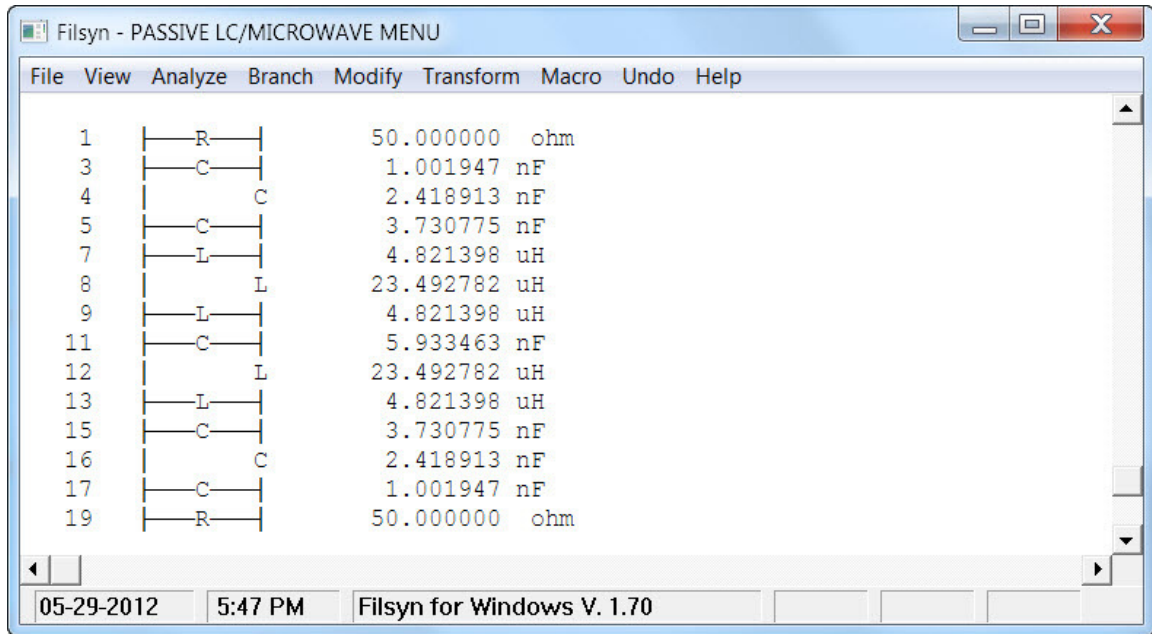


Before continuing, please note that some of the lower degree filters in this group may turn out to be identical to filters generated by other scripts in this set. That is just coincidence in that the filters satisfy the requirements in both scripts.

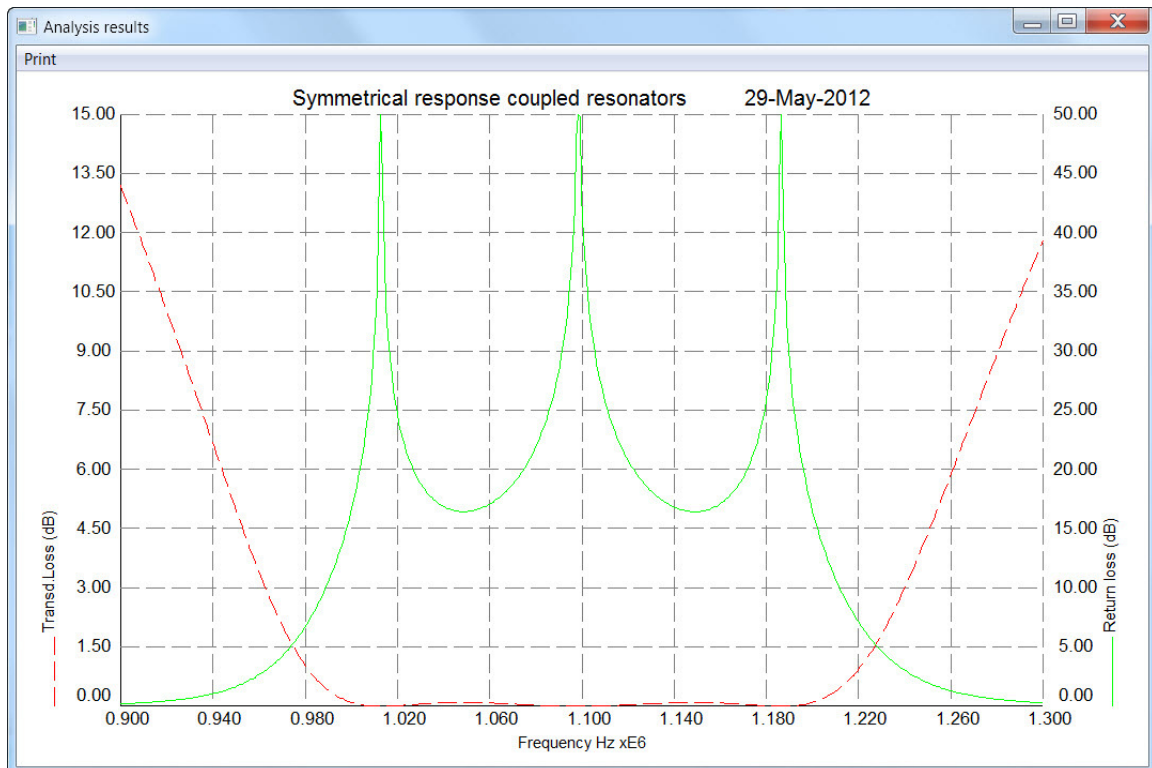
Let us increase the internal impedance level by a factor of two:



yielding the final circuit:



Indeed branches from 7 to 13 have twice the impedance they had before and the circuit starts and ends with pi sections of capacitors. The analysis of the circuit shows perfect results and we display the computed loss and return loss. The response shows a close to arithmetic symmetry:



The capacitively coupled resonator structure can be modified to have series end resonators, yielding the next option, **Modified C coupled**. Here the design is necessarily of

the “conventional” (i.e. non-parametric) form, all equal inductors is the only option available and the number of resonators is limited to 20. Otherwise the data entry screen and the general behavior of the program are similar to those above.

Tubular structures consist of series inductors separated by symmetrical capacitive pi sections, available under the **Tubular** heading. Here the design is necessarily of the “parametric” form and the equal L case is the only one possible. Hence the data entry screen is a bit different as shown below:

The filter may either end with a single capacitive branch, or a capacitive pi section. The inductors will be all equal if the pi section option is selected, otherwise the two end inductors will be different from those in the middle. Number of sections (here meaning number of inductors) is limited to 20 again. Otherwise the program behavior is very similar to all of those in this group.

Next we can modify these tubular structures to have end shunt resonators (**Tubular with end resonators**). Number of inductors may be between 3 and 20 and all but the end inductors will be equal. The structure may be either parametric or conventional. Data entry screen is again similar to those above.

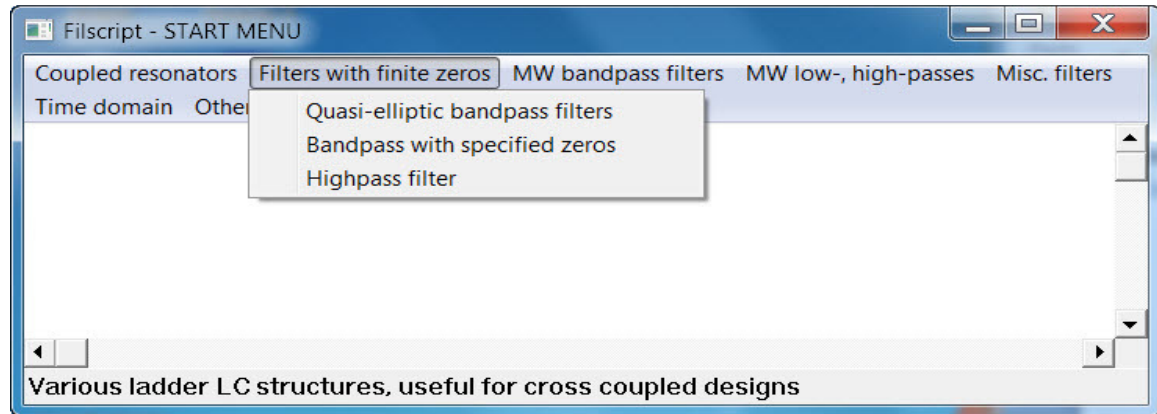
The next type, **Alternating resonator** structure, actually consists of alternating series and shunt inductors, separated by capacitive sections. All inductors again end up being equal. Number of inductors is to be between 3 and 11 and either conventional or parametric forms are available.

The last option, **C coupled with L's**, yields capacitively coupled shunt resonators, but with series inductors at both ends. All shunt inductors will have equal values and either parametric or conventional forms are again available. The number of resonators may be between 2 and 15.

While we have not mentioned it, frequency analysis is available at all these designs in order to see if they meet a given set of requirements.

1.3 Filters with finite zeros

In this second group of options there are only three:



Under **Quasi-elliptic bandpass filters**, we can design bandpass filters with equal-ripple passband and equal-minima type stopbands. The resulting structures are modified to have equal shunt inductors. If we replace the resonator couplings by branches coupling nonadjacent resonators, leading to the so-called coupled-triplet and coupled-quadruplet structures, sometimes these are replaced by all equal shunt capacitors.

The input data screen can be seen below for a particular design. The stopband requirements are automatically set to 50 dB minimum in both stopbands, while the transmission zero entries are grayed out here, since the program will determine their locations.

Quasi-elliptic band pass filters

Filter kind

☒ Passive LC
☐ Microwave

Quarter-wave frequency (Hz)
0.00000

Lower passband edge (Hz)
1.000000E+06

Upper passband edge (Hz)
1.200000E+06

Passband loss ripple (dB)
0.100000000

Band-pass type

☐ Conventional
☒ Parametric

Lower stopband edge (Hz)
940.000000E+03

Upper stopband edge (Hz)
1.240000E+06

No. of zeros at zero
1

No. of zeros at infinity
7

No. of zeros below passband
1

No. of zeros above passband
1

	Transm. zeros (Hz)
1	0.00000
2	0.00000
3	0.00000
4	0.00000
5	0.00000
6	0.00000
7	0.00000
8	0.00000

OK Cancel

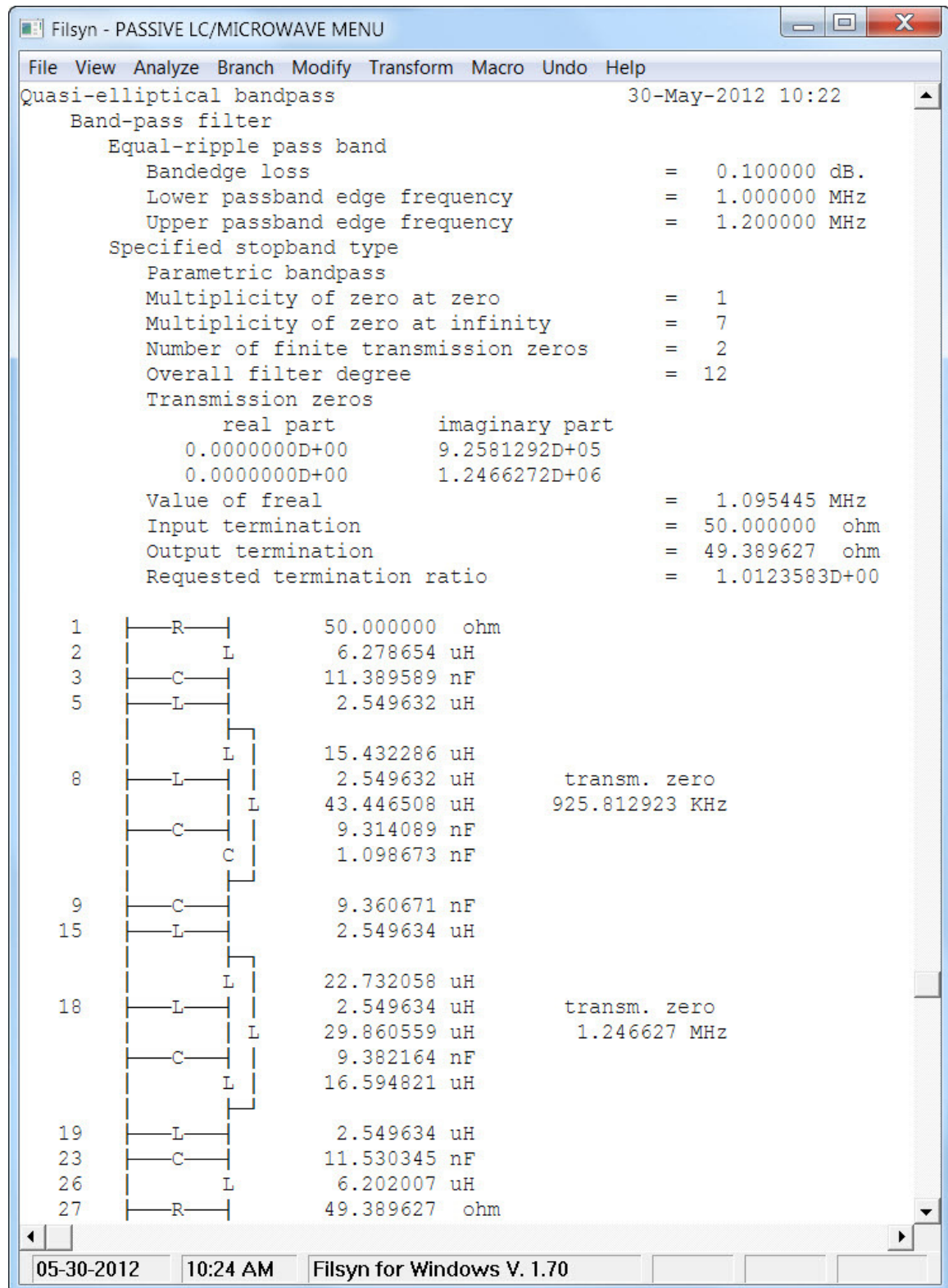
Once the **OK** button is activated, the design proceeds and towards the end we get a prompt as shown below:

Option

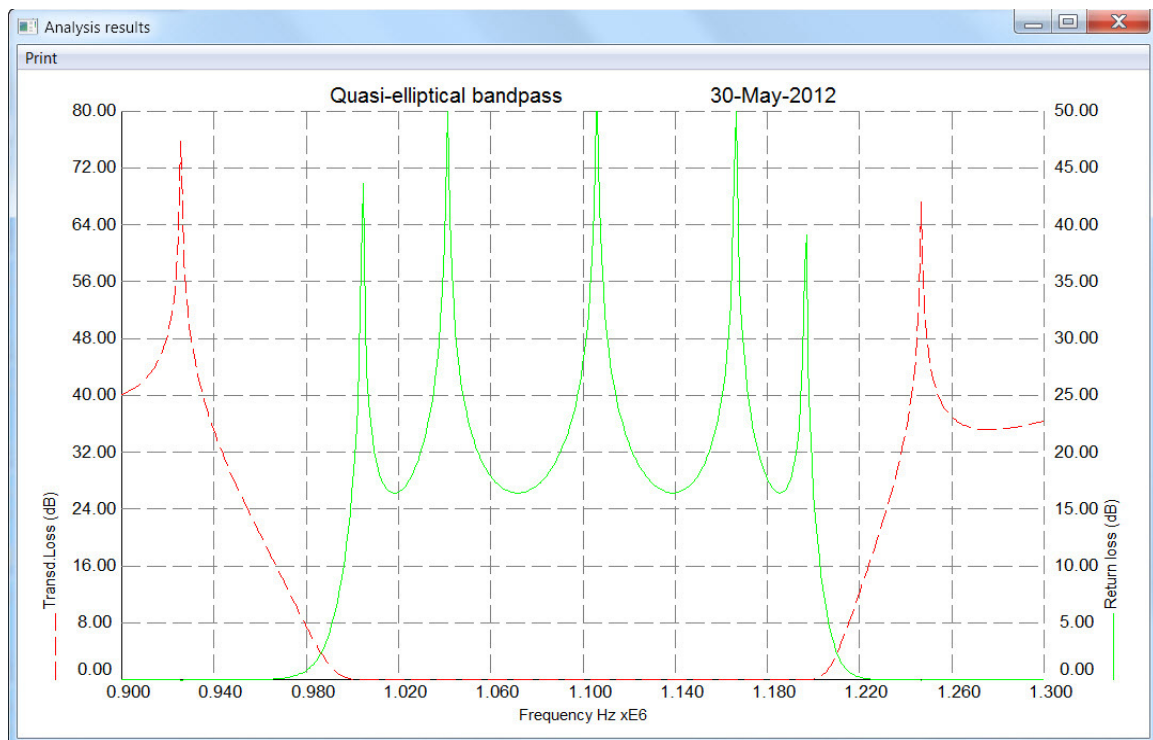
Wish to use triplets?

Yes No Cancel

Accepting the offer, we get the final circuit as shown:



Please note that all shunt inductors have the same value. Analyzing this circuit in the frequency domain, the results are shown in the plot below:



The next option in this column, **Bandpass with specified zeros**, is in fact identical to the first, except that we can specify the location of the transmission zeros ourselves. The data entry screen is the same as for the quasi-elliptic case above, but this time the frequency entry is active, but other items are grayed out instead; otherwise the design procedure is identical to that described above and will not be pursued here.

Bandpass w. specified transm. zeros

Filter kind

☒ Passive LC

☐ Microwave

Quarter-wave frequency (Hz)

0.00000

Lower passband edge (Hz)

0.00000

Upper passband edge (Hz)

0.00000

Passband loss ripple (dB)

0.100000000

Band-pass type

☐ Conventional

☒ Parametric

Lower stopband edge (Hz)

0.00000

Upper stopband edge (Hz)

0.00000

No. of zeros at zero

0

No. of zeros at infinity

0

No. of zeros below passband

0

No. of zeros above passband

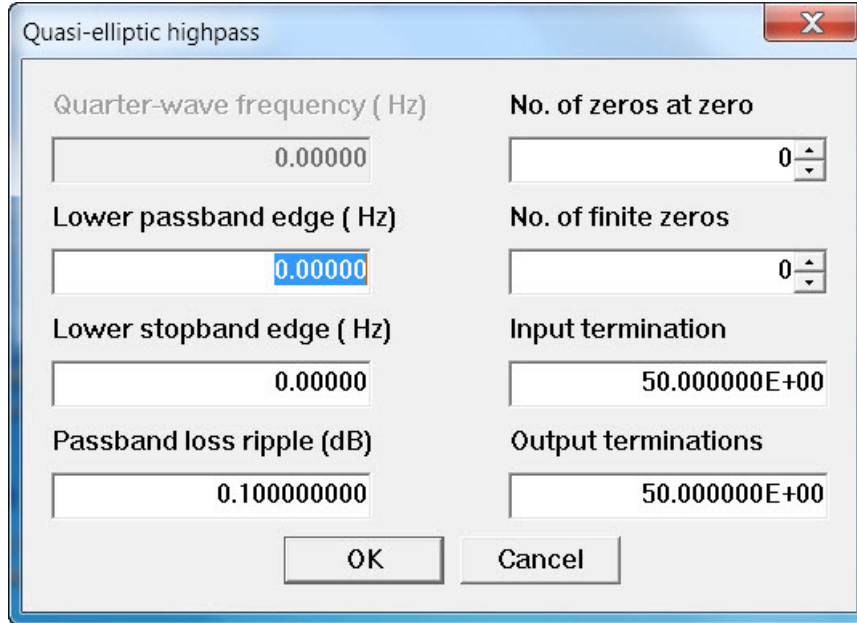
0

	Transm. zeros (Hz)
1	0.00000
2	0.00000
3	0.00000
4	0.00000
5	0.00000
6	0.00000
7	0.00000
8	0.00000

OK Cancel

These two script files use the, about 240, auxiliary files in the Script subfolder and the design can be completed only if the folder contains an appropriate file, that depends on the number and distribution of transmission zeros. If an appropriate file does not exist, a message to that effect is printed. In such a case please let us know the details and we will be glad to develop a corresponding script. Nevertheless, the design can be continued using the circuit configuration generated by the **Filsyn** program.

The last, **Highpass filter** option gives us the quasi-elliptic solution to highpass filter problems. The data entry screen is shown below. The stopband loss requirement is set to 50 dB again, while the **Quarter-wave frequency** entry is grayed out since this is a lumped LC design. The reason the item is there is that the same menu is reused later for microwave filters. We do not show a complete design, it does not demonstrate anything new.



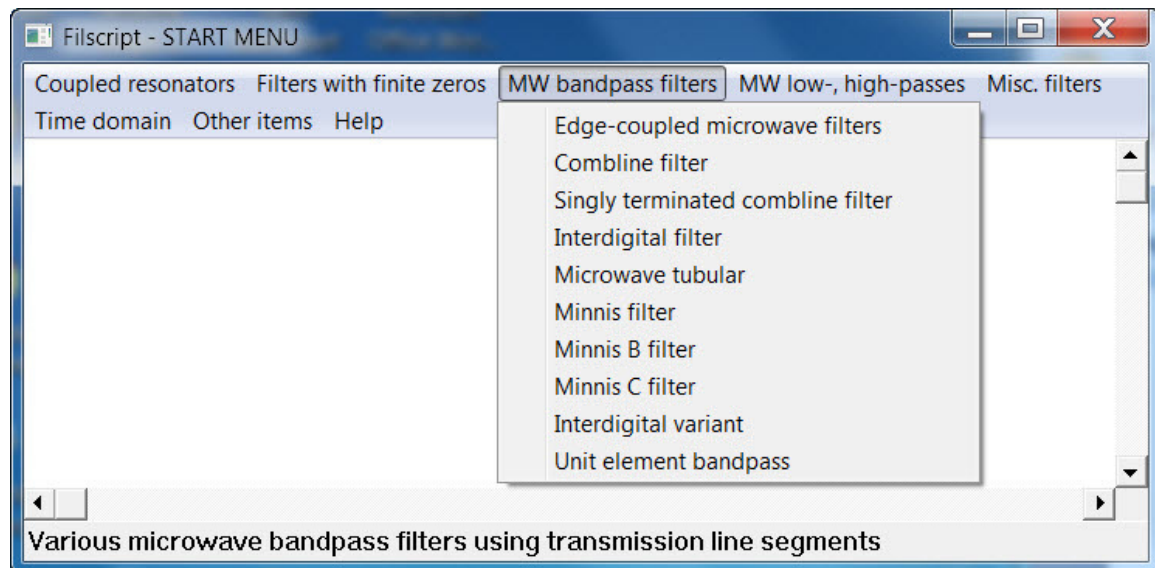
Quasi-elliptic highpass

Quarter-wave frequency (Hz)	No. of zeros at zero
<input type="text" value="0.00000"/>	<input type="text" value="0"/>
Lower passband edge (Hz)	No. of finite zeros
<input type="text" value="0.00000"/>	<input type="text" value="0"/>
Lower stopband edge (Hz)	Input termination
<input type="text" value="0.00000"/>	<input type="text" value="50.000000E+00"/>
Passband loss ripple (dB)	Output terminations
<input type="text" value="0.10000000"/>	<input type="text" value="50.000000E+00"/>

OK Cancel

1.4 Microwave bandpass filters

Next we get to the microwave bandpass filters:



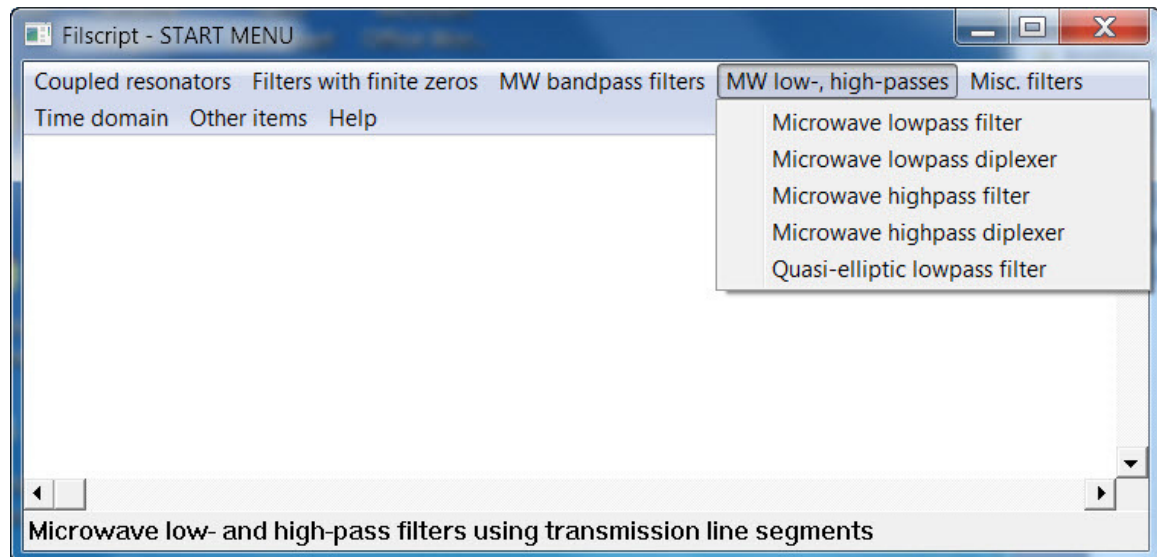
- **Edge-coupled microwave filters.** This script will generate the equivalent circuit for edge coupled microwave filters having one transmission zero at zero frequency and from two to nine unit elements (corresponding to from 3 to 10 sections)
- **Comblin filter.**
- **Singly terminated comblin filter.**
- **Interdigital filter.** This script file is for the design of short-circuited input interdigital filters.
- **Microwave tubular filters.** These are the microwave versions of the tubular filters,

consisting of eighth-wavelength unit elements (transmission-line segments) flanked by pi-sections of commensurate open-circuited stubs (microwave capacitors)

- **Minnis filter.** This script file designs microwave filters of class A as described by B.J. Minnis (see ref[32])
- **Minnis B filter.** This script file designs microwave filters of class B as described by B.J. Minnis (see ref[32])
- **Minnis C filter.** This is a generalization of the Minnis filter which has two transmission zeros that are triple zeros instead of the usual double zeros. The resulting filter is more complex, but strictly structurally symmetrical, as usual.
- **Interdigital variant.**
- **Unit element bandpass filters.** The topology of these filters is restricted to unit elements separated by shunt short-circuited stubs. For bandwidths of less than 100%, there are stubs between every pair of unit elements. For bandwidths of greater than 100%, there are only two stubs, placed symmetrically. These can be moved or split later to adjust impedance levels as desired. When the script stops all stub impedances are set to be equal.

1.5 Microwave low- and highpasses

The next group contains design files for microwave low- and high-pass filters:



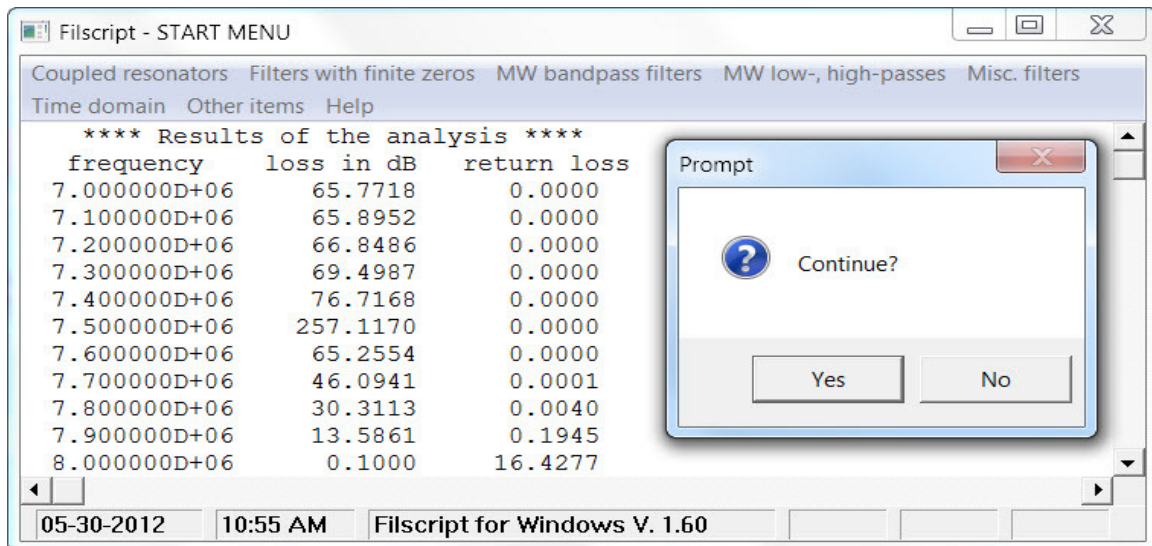
- **Microwave lowpass filter.** These lowpass filters contain contributing unit elements and both finite zeros and zeros at the quarter-wave frequency. The limits are from one to four finite transmission zeros (all at the same frequency), and from one to four zeros at the quarter-wave frequency. The number of unit elements necessary to eliminate all series inductors, is computed by the program. Only odd overall degree is possible.

- **Microwave lowpass diplexer.** These filters are very similar to the ones described above, except that one end is short-circuit driven. This end will be paralleled.
- **Microwave highpass filter.** The topology of these filters is limited to from one to three transmission zeros at zero frequency and from one to four finite transmission zeros at one or two frequencies. The filter will have a structural symmetry. The number of contributing unit elements is restricted to a finite number of combinations.

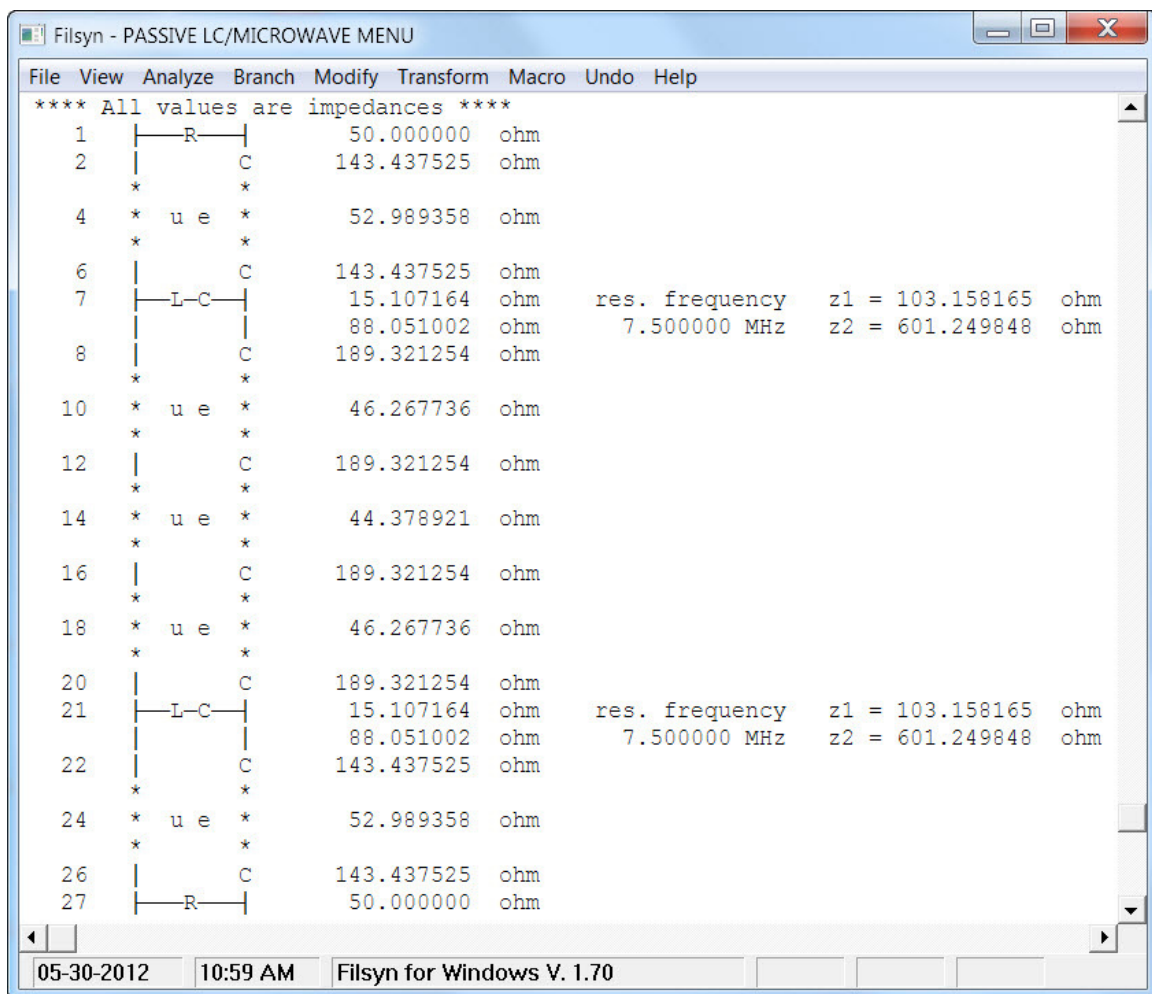
A simple design of this kind starts with a data input screen:

Parameter	Value
Lower passband edge (MHz)	8.000000E+00
Quarter-wave frequency (MHz)	10.000000E+00
Passband loss ripple (dB)	0.100000000
No. of zeros at zero frequency	1
No. of unit elements	4
No. of finite zeros	2
Finite zero (MHz)	7.5
The other zero (MHz)	0.00000
Multiplicity	1

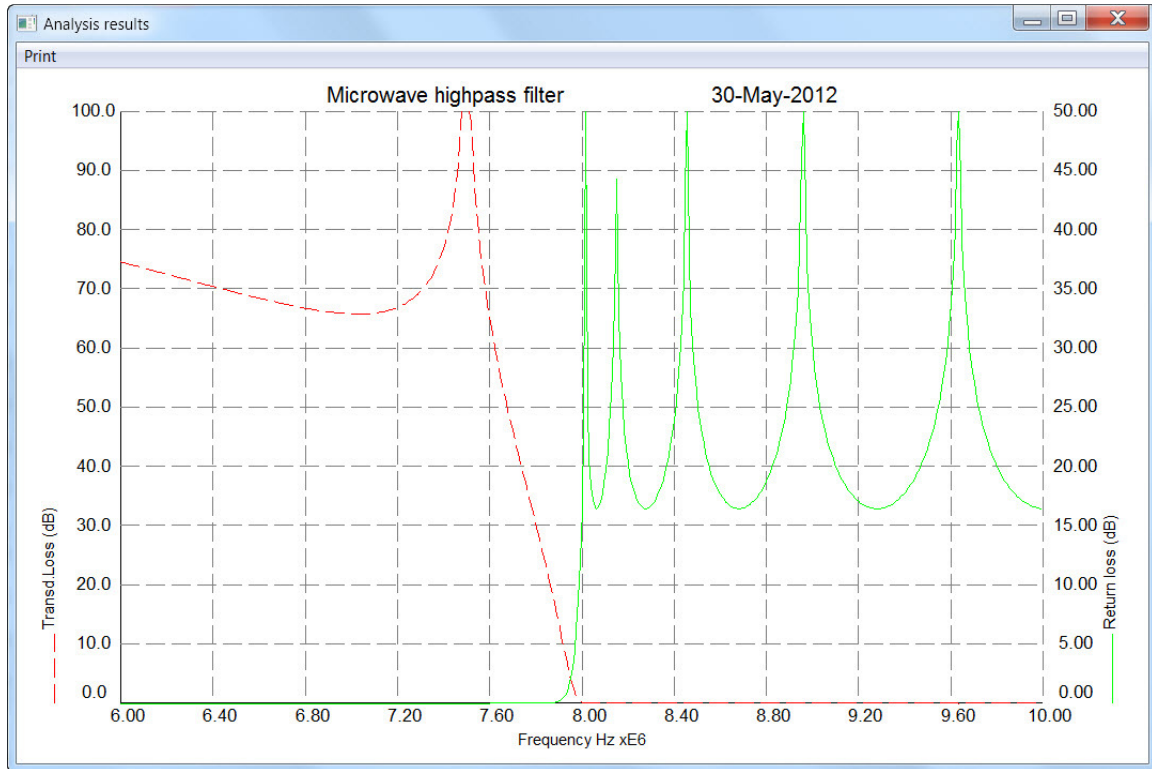
Clicking on the **OK** button brings up an advisory, saying that the number of unit elements is incorrect. The same data entry screen comes back with most of the data still in it and we change the number of unit elements to 5, which is now acceptable. This is due to the fact that the number of unit elements and finite transmission zeros need to satisfy certain rules. We can also perform a frequency domain analysis here, since all necessary information is known:



Next comes the actual synthesis (using manual synthesis again, but performed by the script file) and when that stops, we get the final circuit:



Element values are all quite reasonable. Note that the circuit is also structurally symmetrical. The z_1 and z_2 values listed together with the resonant branch offer an alternative implementation to the series connected shorted and open stubs ($L - C$ branch). This alternative is the cascade connection of two quarter-wave long segments (unit elements) of impedances z_1 and z_2 , with the second segment left open-circuited. Analyzing the circuit, we get the plotted response:



Microwave highpass diplexer. These filters can have from one to two transmission zeros at zero frequency and from one to four finite transmission zeros at a single frequency. All series lines are made up of unit elements and there may be either one or three unit elements in each series position.

Microwave Quasi-elliptic lowpass filter. These are filters with equal-ripple passband and equal-minima stopband behavior and have contributing unit elements; their number is selected to eliminate all series branches. The number of finite transmission zeros is limited to 10.

Unit elements in microwave lowpass filters may be used to replace the series branches of shorted stubs (series L 's) by shunt branches of open stubs (shunt C 's) and they also make the interconnection of the branches substantially simpler. These unit elements may be non-contributing or contributing to the filtering characteristics. Incorporating the contributing kind into the design, this script file uses the **Placer** procedure to yield quasi-elliptic type of performance, illustrated by the following example:

Quasi-elliptic MW lowpass

Quarter-wave frequency (MHz)

Upper passband edge (MHz)

Passband loss ripple (dB)

Upper stopband zero (MHz)

No. of finite transmission zeros

OK Cancel

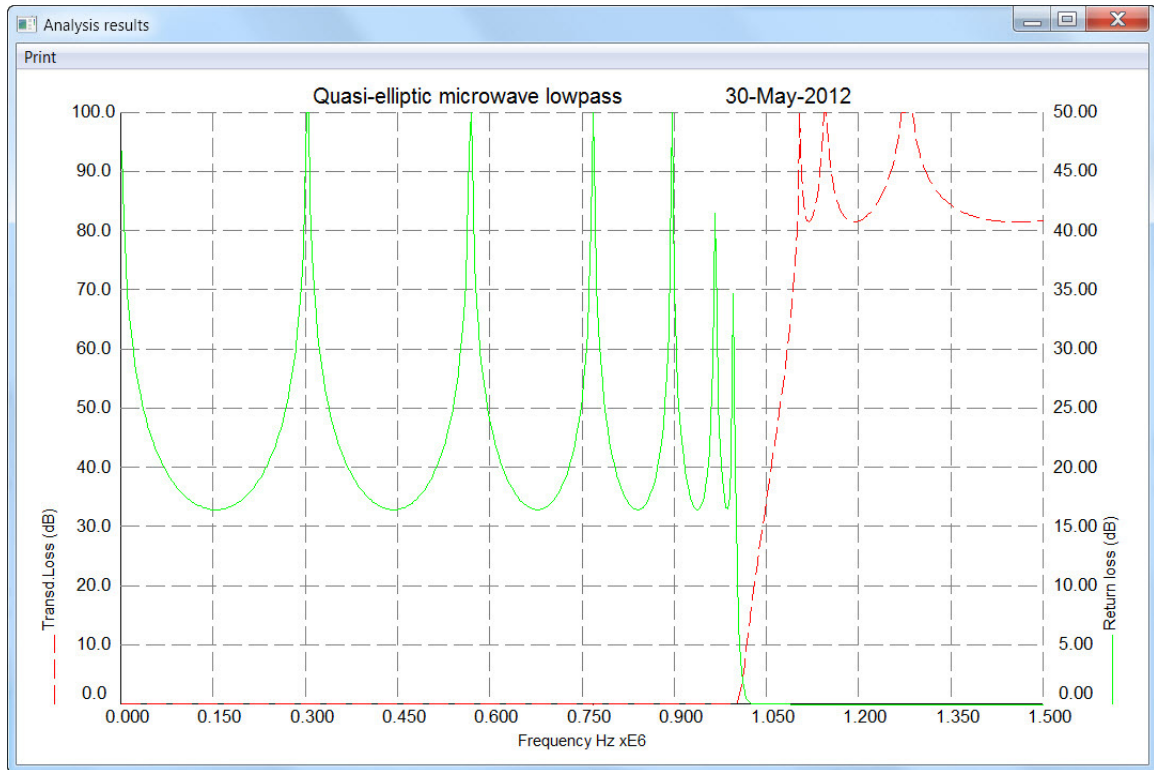
After a fairly large number of automated steps the procedure comes to a halt at:

Filsyn - PASSIVE LC/MICROWAVE MENU

File	View	Analyze	Branch	Modify	Transform	Macro	Undo	Help
1	—R—		50.000000	ohm				
3	—C—		6.584414	ohm				
	* * *							
5	* u e *		288.013456	ohm				
	* * *							
7	—L—C—		339.494540	ohm	res. frequency	z1 = 349.919150	ohm	
			10.424610	ohm	1.104349 MHz	z2 = 10.744711	ohm	
	* * *							
9	* u e *		378.122218	ohm				
	* * *							
11	—C—		3.467687	ohm				
	* * *							
13	* u e *		397.677612	ohm				
	* * *							
15	—L—C—		220.910227	ohm	res. frequency	z1 = 228.217862	ohm	
			7.307635	ohm	1.145352 MHz	z2 = 7.549369	ohm	
	* * *							
17	* u e *		396.129897	ohm				
	* * *							
19	—C—		3.476234	ohm				
	* * *							
21	* u e *		427.124635	ohm				
	* * *							
23	—L—C—		151.959779	ohm	res. frequency	z1 = 158.250356	ohm	
			6.290578	ohm	1.277830 MHz	z2 = 6.550984	ohm	
	* * *							
25	* u e *		354.281731	ohm				
	* * *							
27	—C—		6.556377	ohm				
29	—R—		50.000000	ohm				

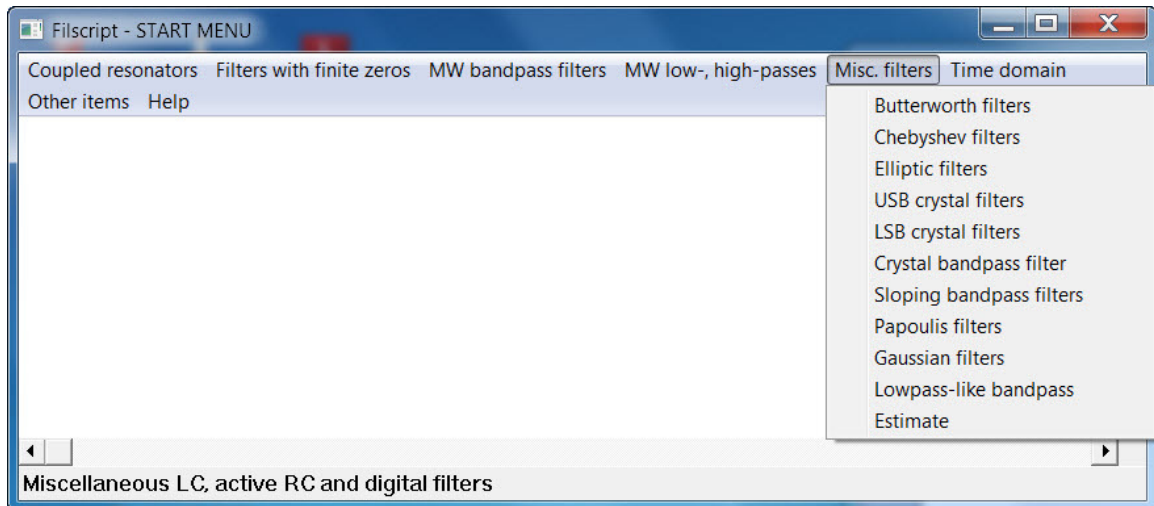
05-30-2012 11:12 AM Filsyn for Windows V. 1.70

where we only show the circuit itself. The performance is what we expect from a quasi-elliptic design:



1.6 Miscellaneous filters

A group of miscellaneous filter design scripts are grouped under the next option. All of these filters can be implemented in passive LC, active RC or IIR digital forms. Note also, that the first three of these menu items permits us the direct design of band-reject filters of the proper type, hence there is no separate menu option for band-reject filters.



- **Butterworth filters.** This script file simplifies the design of Butterworth lowpass, highpass, bandpass and band-reject filters of degree between 3 and 20.
- **Chebyshev filters.** This file does the same for Chebyshev filters.
- **Elliptic filters.** This file does the same for elliptic filters. We shall demonstrate this case with a band-reject filter:

Elliptic filters

Filter type

- ☐ Low pass
- ☐ High pass
- ☐ Band pass
- ☒ Band reject

Stopband min. loss (dB)

45

Filter degree

0

Implementation

- ☒ Passive LC
- ☐ Active RC
- ☐ IIR digital

Lower passband edge (Hz)

1.500000E+03

Upper passband edge (Hz)

3.500000E+03

Lower stopband edge (Hz)

1.550000E+03

Upper stopband edge (Hz)

3.400000E+03

Passband edge loss (dB)

0.100000000

Sampling frequency (Hz)

0.00000

Input termination

50.0000000

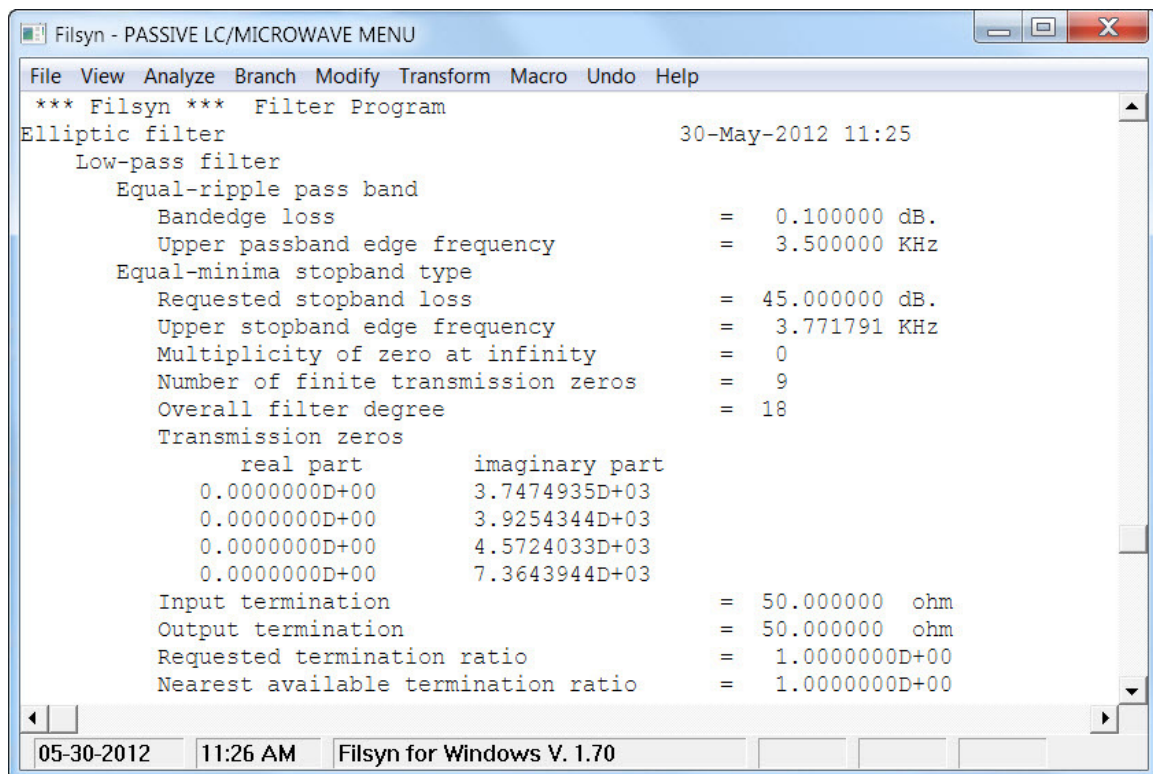
Output terminations

50.0000000

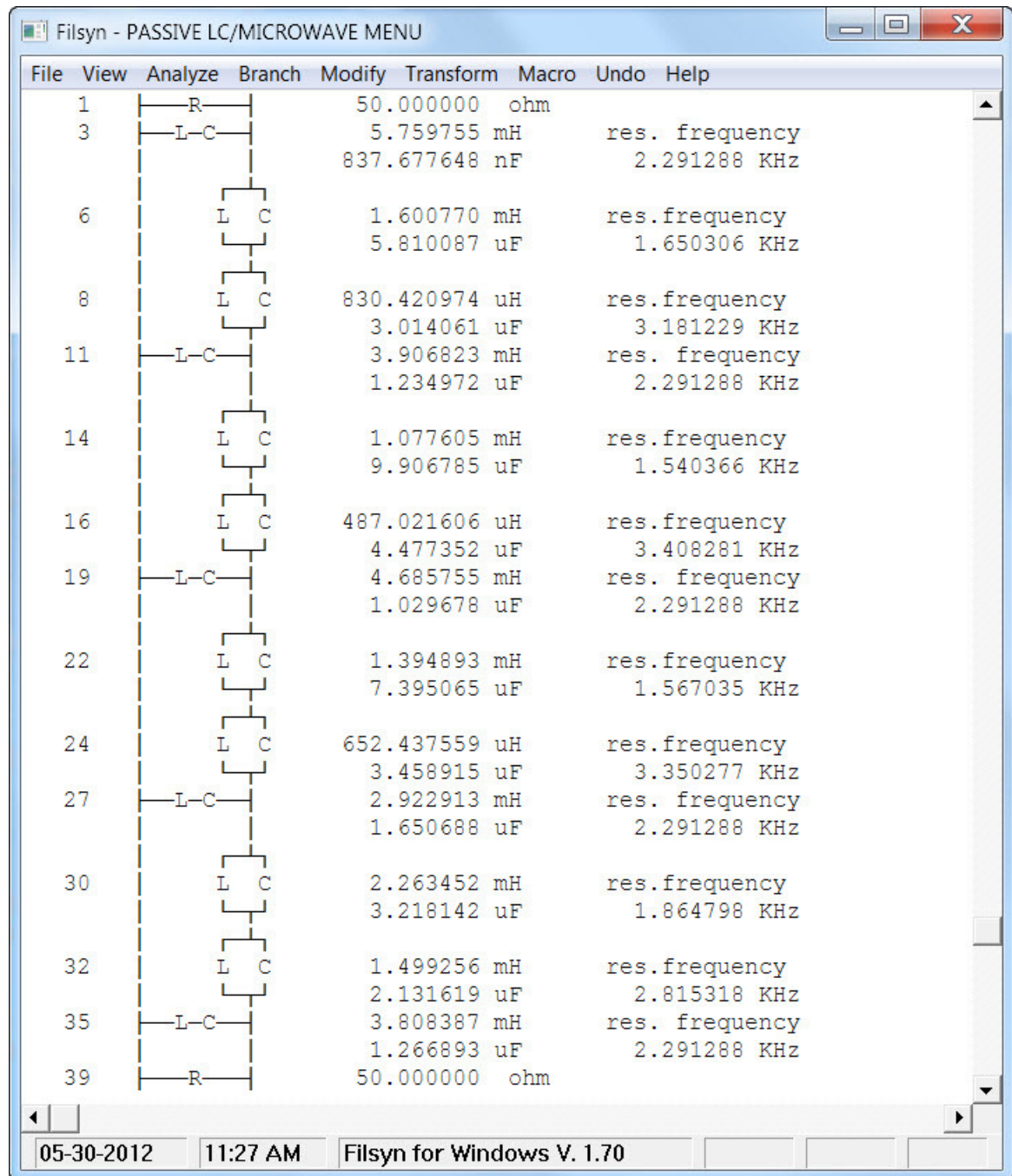
OK Cancel

Later on we will use the exact same specification in the **Estimate** menu option, to find the minimum filter degree needed to meet these specifications. It turns out, that the starting lowpass filter degree needed is eight, but an even degree lowpass can only be implemented in passive LC form if at least two of the zeros are at infinite frequency. For those familiar with the theory of elliptic filters, this means that a type ‘a’ filter is replaced by a type ‘c’. However, if two zeros are shifted to infinity, the requirements are not met anymore, so that the degree is automatically increased to 9.

The resulting filter summary indicates this. Note that the filter is initially designed as a lowpass and is converted into a band-reject at the last step, already inside the passive LC analysis module:



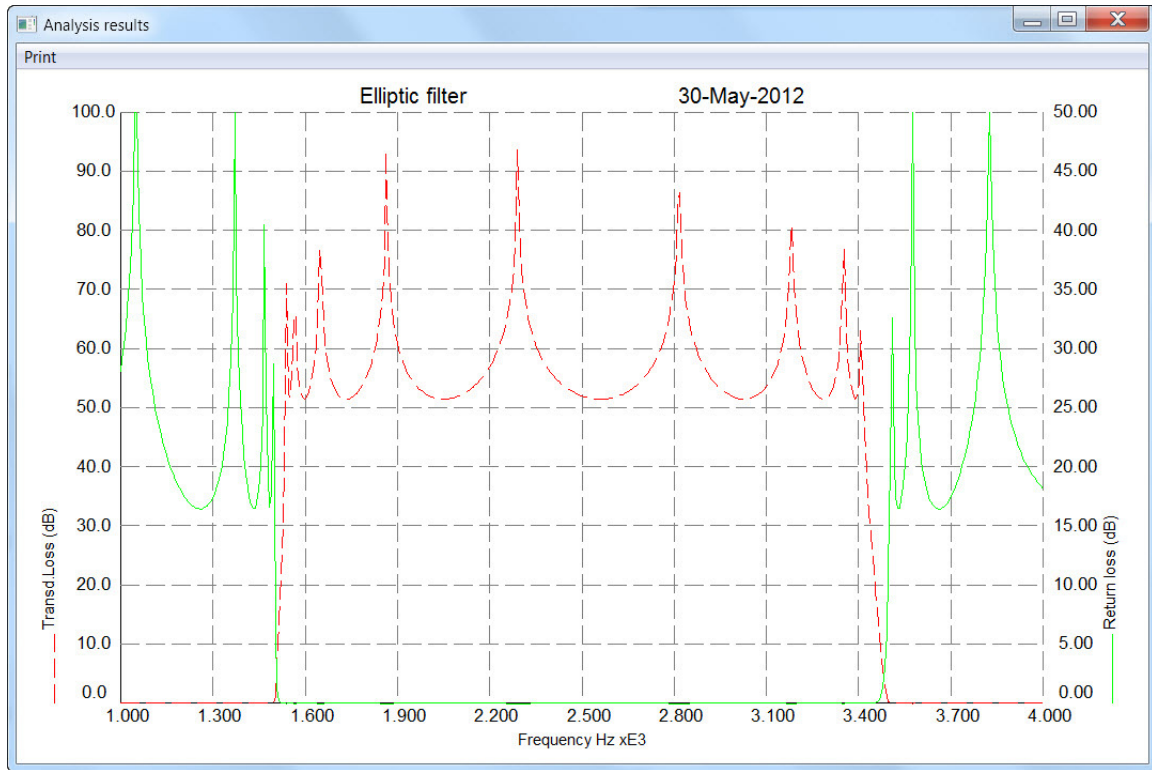
The actual circuit is shown below:



File	View	Analyze	Branch	Modify	Transform	Macro	Undo	Help
1		R		50.000000	ohm			
3		L-C		5.759755 mH		res. frequency		
				837.677648 nF		2.291288 KHz		
6		L C		1.600770 mH		res. frequency		
				5.810087 uF		1.650306 KHz		
8		L C		830.420974 uH		res. frequency		
				3.014061 uF		3.181229 KHz		
11		L-C		3.906823 mH		res. frequency		
				1.234972 uF		2.291288 KHz		
14		L C		1.077605 mH		res. frequency		
				9.906785 uF		1.540366 KHz		
16		L C		487.021606 uH		res. frequency		
				4.477352 uF		3.408281 KHz		
19		L-C		4.685755 mH		res. frequency		
				1.029678 uF		2.291288 KHz		
22		L C		1.394893 mH		res. frequency		
				7.395065 uF		1.567035 KHz		
24		L C		652.437559 uH		res. frequency		
				3.458915 uF		3.350277 KHz		
27		L-C		2.922913 mH		res. frequency		
				1.650688 uF		2.291288 KHz		
30		L C		2.263452 mH		res. frequency		
				3.218142 uF		1.864798 KHz		
32		L C		1.499256 mH		res. frequency		
				2.131619 uF		2.815318 KHz		
35		L-C		3.808387 mH		res. frequency		
				1.266893 uF		2.291288 KHz		
39		R		50.000000	ohm			

05-30-2012 11:27 AM Filsyn for Windows V. 1.70

while the results of a frequency domain analysis show a very satisfactory performance:



Other filter kinds available in this group are:

- **Upper side-band USB crystal filters** and **lower side-band LSB crystal filters**. These files simplify the design of ladder crystal filters with all shunt crystal branches or all series crystal branches respectively. The stopbands may be either of the equal-minima type or the transmission zeros may be specified individually. Details may be found in *Application Note 11*.
- **Crystal bandpass filter**. This is about the design of a specific bandpass filter with exactly four transmission zeros and implemented in the form of two cascaded lattice sections.
- **Sloping bandpass filters**. This script helps to design a bandpass interstage network with a specified loss slope in the passband and a specified additional flat loss. The circuit may also be used to match complex impedances by absorbing reactive elements of the terminating impedances. This may be controlled by the amount of flat loss used.

Let us consider an interstage network with a slope of -4.5 dB/oct from 2 MHz to 4 MHz. In addition, we need to specify the number of transmission zeros at extreme frequencies, and a value of 3 suffices for both:

Sloping bandpass filter

Lower passband edge (Hz)	No. of zeros at zero
<input type="text" value="2.000000E+06"/>	<input type="text" value="3"/>
Upper passband edge (Hz)	No. of zeros at infinity
<input type="text" value="4.000000E+06"/>	<input type="text" value="3"/>
Passband loss ripple (dB)	Input termination
<input type="text" value="0.100000000"/>	<input type="text" value="50.0000000"/>
Loss slope in dB/oct	Output termination
<input type="text" value="-4.50000000"/>	<input type="text" value="50.0000000"/>
Additional flat loss (dB)	<input type="button" value="OK"/> <input type="button" value="Cancel"/>
<input type="text" value="0.00000000"/>	

The resulting circuit, after correcting the output termination to the required 50 ohms:

Filsyn - PASSIVE LC/MICROWAVE MENU

File View Analyze Branch Modify Transform Macro Undo Help

*** Filsyn *** Filter Program

Sloping bandpass filter 30-May-2012 14:47

Band-pass filter

Equal-ripple pass band

Bandpass loss slope	= -4.500000 dB/octave
Bandedge loss	= 0.100000 dB.
Lower passband edge frequency	= 2.000000 MHz
Upper passband edge frequency	= 4.000000 MHz

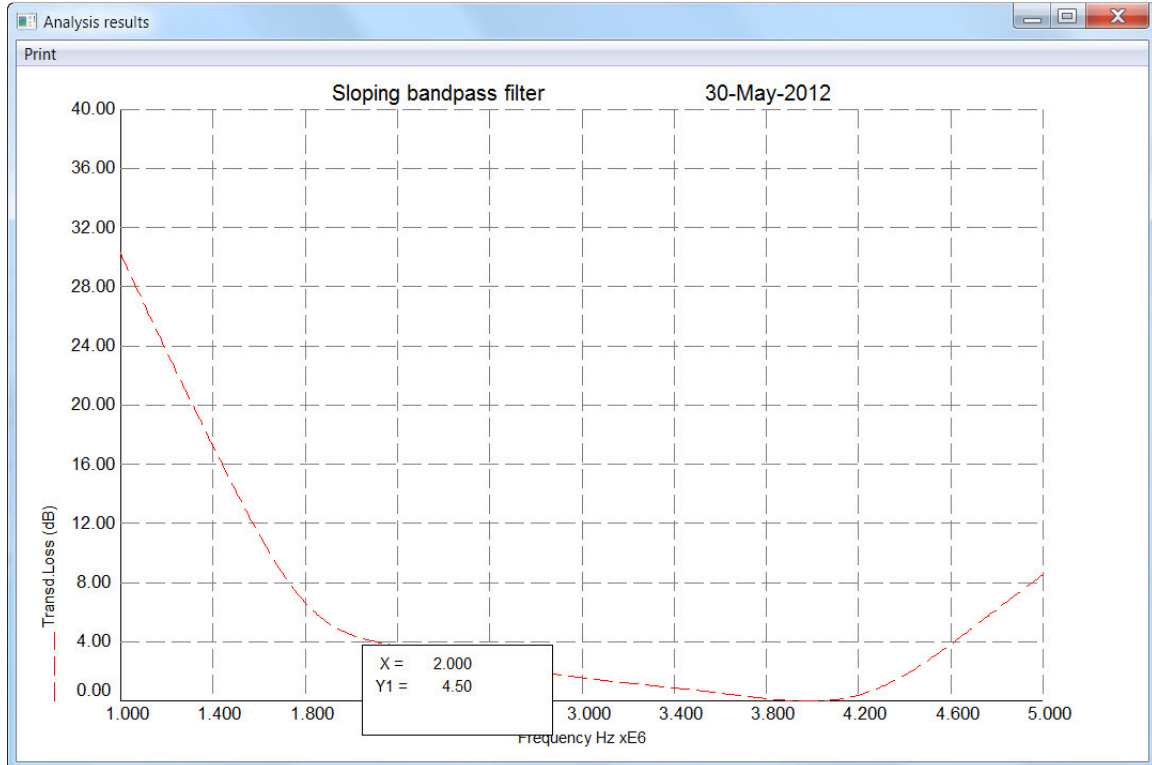
Equal-minima stopband type

Multiplicity of zero at zero	= 3
Multiplicity of zero at infinity	= 3
Overall filter degree	= 6
Input termination	= 50.000000 ohm
Output termination	= 50.000000 ohm
Requested termination ratio	= 1.000000D+00

1	—R—	50.000000 ohm
2	—C—	853.929351 pF
4	—L—	5.321760 uH
5	—L—	2.147961 uH
7	—C—	1.026721 nF
8	—C—	429.160313 pF
9	—C—	309.198634 pF
10	—L—	3.340237 uH
11	—R—	50.000000 ohm

05-30-2012 2:47 PM Filsyn for Windows V. 1.70

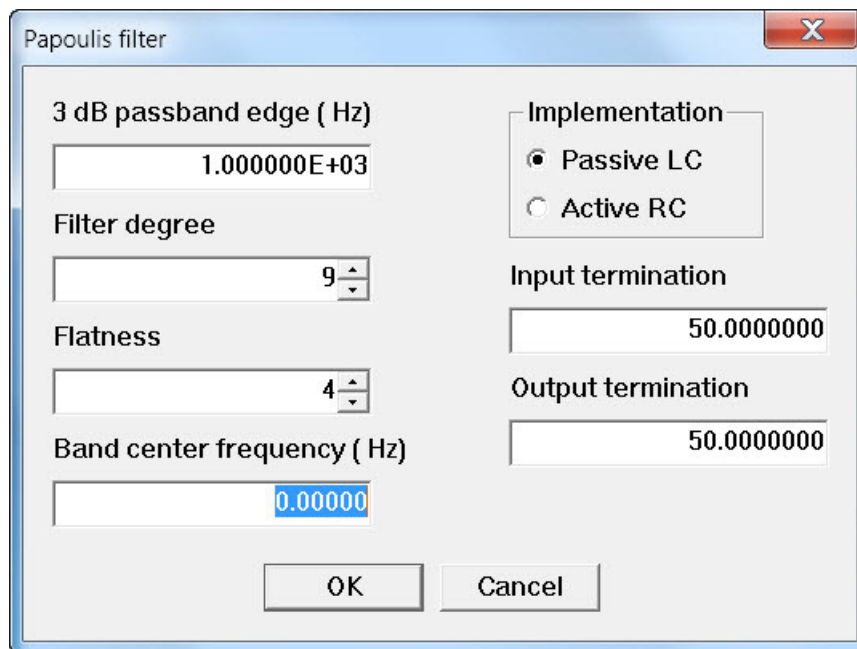
The analysis results show a perfect agreement, the loss at 2 MHz is exactly 4.5 dB:



Another feature shown on this display is the following. If we move the mouse pointer over the graph, it turns into a crossed pair of lines. If we follow (any of the) curves in the graph and click on the right mouse button, the display shows the coordinate values of the nearest point on the curve.

- Papoulis filters.** These are lowpass filters with monotonic loss characteristics and the steepest loss curve at the 3 dB point. The group also contains transitional filters between the Papoulis group and the Butterworth group. This is selected by specifying an additional integer “flatness” parameter. This must be less than the overall degree and of opposite polarity. The value 0 (or 1) specifies the Papoulis group, while flatness = degree – 3 corresponds to the Butterworth filters and therefore is not available here. Filter degrees from 2 to 15 are available. See ref[36], [17] and [13].

A sample design is shown below for a transitional filter between Butterworth and Papoulis designs:



Papoulis filter

3 dB passband edge (Hz)
1.000000E+03

Filter degree
9

Flatness
4

Band center frequency (Hz)
0.00000

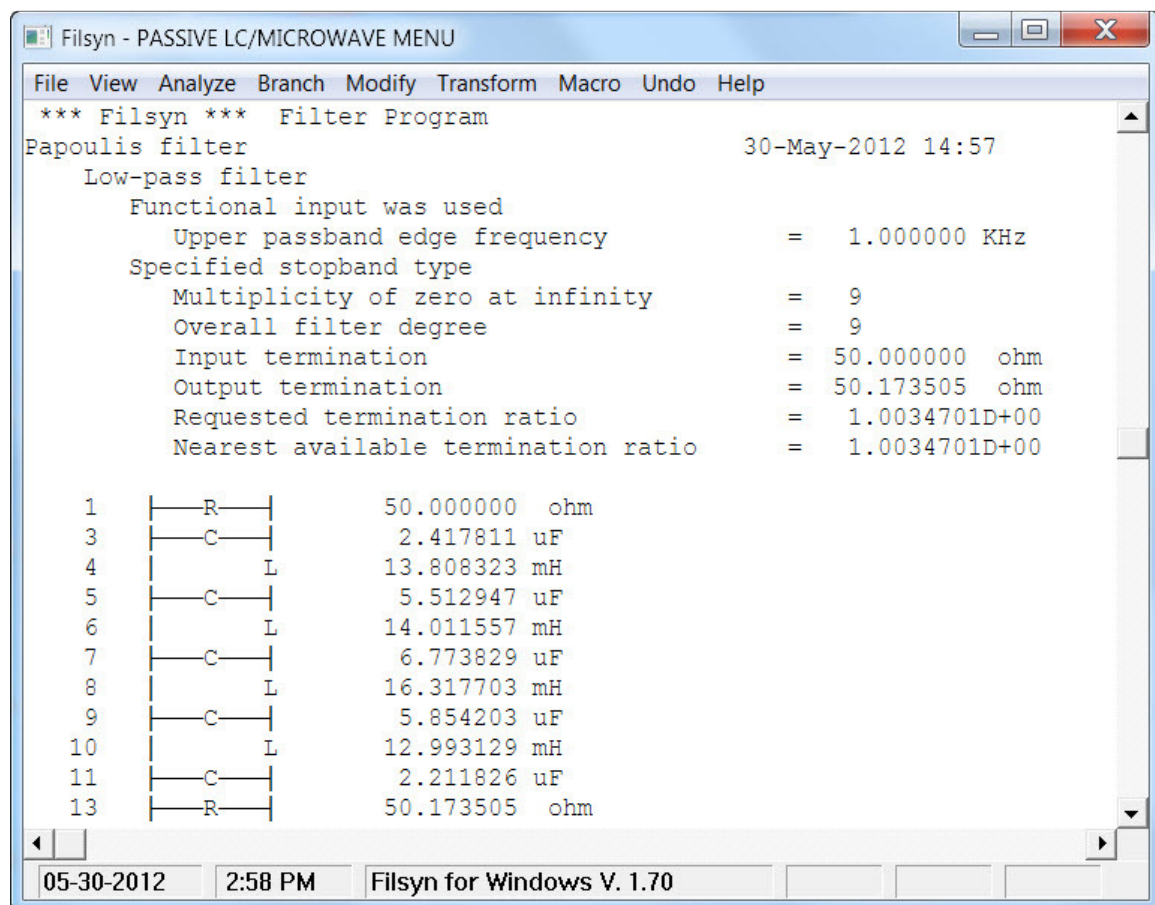
Implementation
☒ Passive LC
☐ Active RC

Input termination
50.0000000

Output termination
50.0000000

OK Cancel

The resulting structure is shown on the next screen:



Filsyn - PASSIVE LC/MICROWAVE MENU

File View Analyze Branch Modify Transform Macro Undo Help

*** Filsyn *** Filter Program

Papoulis filter 30-May-2012 14:57

Low-pass filter

Functional input was used

Upper passband edge frequency = 1.000000 KHz

Specified stopband type

Multiplicity of zero at infinity = 9

Overall filter degree = 9

Input termination = 50.000000 ohm

Output termination = 50.173505 ohm

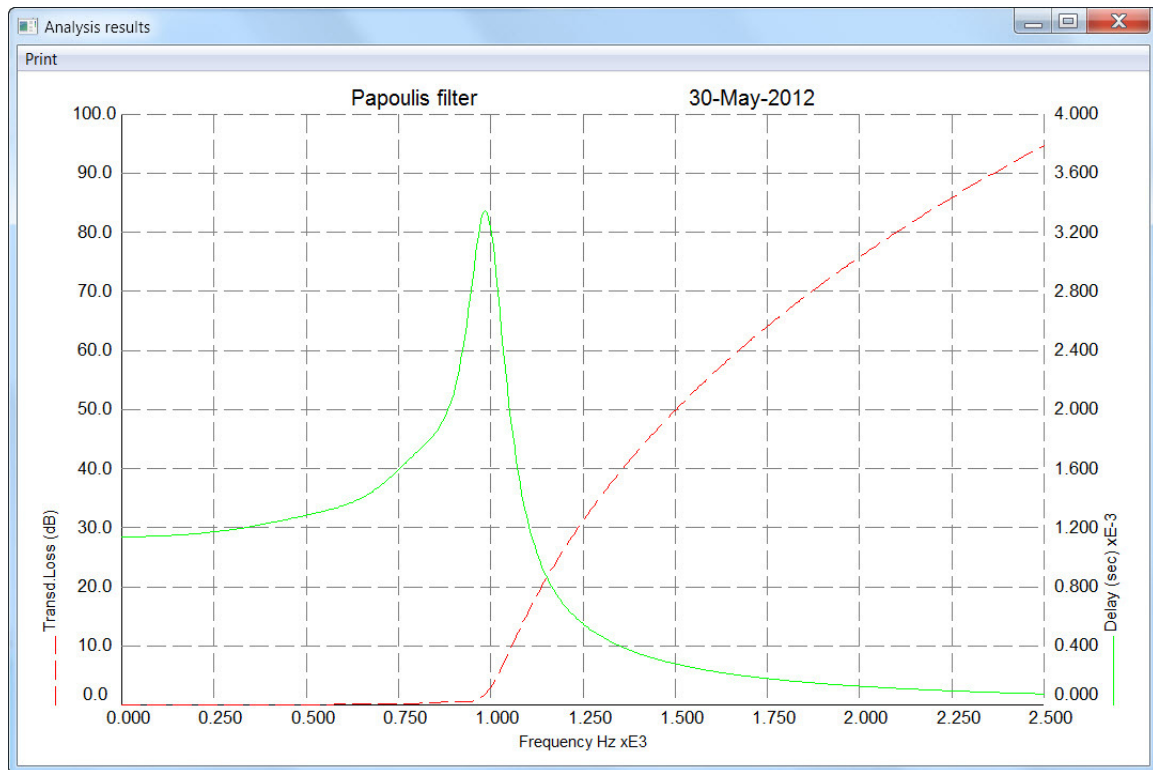
Requested termination ratio = 1.0034701D+00

Nearest available termination ratio = 1.0034701D+00

1	—R—	50.000000 ohm
3	—C—	2.417811 uF
4	—L—	13.808323 mH
5	—C—	5.512947 uF
6	—L—	14.011557 mH
7	—C—	6.773829 uF
8	—L—	16.317703 mH
9	—C—	5.854203 uF
10	—L—	12.993129 mH
11	—C—	2.211826 uF
13	—R—	50.173505 ohm

05-30-2012 2:58 PM Filsyn for Windows V. 1.70

Its performance is shown next:



- **Gaussian filters.** These filters approximate the Gaussian loss characteristics either in the maximally flat manner or in the equal-ripple manner up to the 6 dB or 12 dB points with a ± 0.05 dB ripple. Filter degrees from 2 to 15 are available for the maximally flat case and from 2 to 12 for the equal-ripple cases
- **Estimate.** This script does not perform any design, but it gives us an estimate of the required filter degree for a set of loss specifications with Butterworth, Chebyshev or elliptic characteristics. You enter the pass- and stop-band data and the required pass and stopband loss and clicking on the **OK** button gives you the required degree (including the fractional value). Alternatively (or additionally), you can enter the degree and leave either the passband or the stopband loss value at zero. Clicking on the **OK** button again computes the missing value for the passband or stopband loss. Note that the inverse Chebyshev filters would give results identical to the Chebyshev case.

Let us demonstrate this with a band-reject filter specification as shown:

Estimate necessary filter degree

Filter kind

- ☐ Low pass
- ☐ High pass
- ☐ Band pass
- ☒ Band reject

Filter type

- ☐ Butterworth
- ☐ Chebyshev
- ☒ Elliptic

Lower passband edge (Hz)

1.500000E+03

Upper passband edge (Hz)

3.500000E+03

Lower stopband edge (Hz)

1.550000E+03

Upper stopband edge (Hz)

3.400000E+03

Passband edge loss (dB)

0.100000000

Stopband min. loss (dB)

45

Transition ratio

0.00000000

Filter degree

0.00000000

0

OK Cancel

The transition ratio and the first (fractional) degree boxes are for display only. When we click on the **OK** button, we get:

Estimate necessary filter degree

Filter kind

- ☐ Low pass
- ☐ High pass
- ☐ Band pass
- ☒ Band reject

Filter type

- ☐ Butterworth
- ☐ Chebyshev
- ☒ Elliptic

Lower passband edge (Hz)

1.500000E+03

Upper passband edge (Hz)

3.500000E+03

Lower stopband edge (Hz)

1.550000E+03

Upper stopband edge (Hz)

3.400000E+03

Passband edge loss (dB)

0.100000000

Stopband min. loss (dB)

45.0000000

Transition ratio

1.07765452

Filter degree

7.99820493

16

OK Cancel

Now we could set either loss requirements to zero and click on the **OK** button again, to see how much spare we have in them. However, since the fractional degree is so close to the integer 8, we would see practically no change at all. At the same time, precisely because this number is so close to the *even* number 8, if we wish to implement this filter in passive-LC form, the lowpass degree will automatically be raised to 9 and the band-reject degree to 18 as explained above as well as in *Appendix C*.

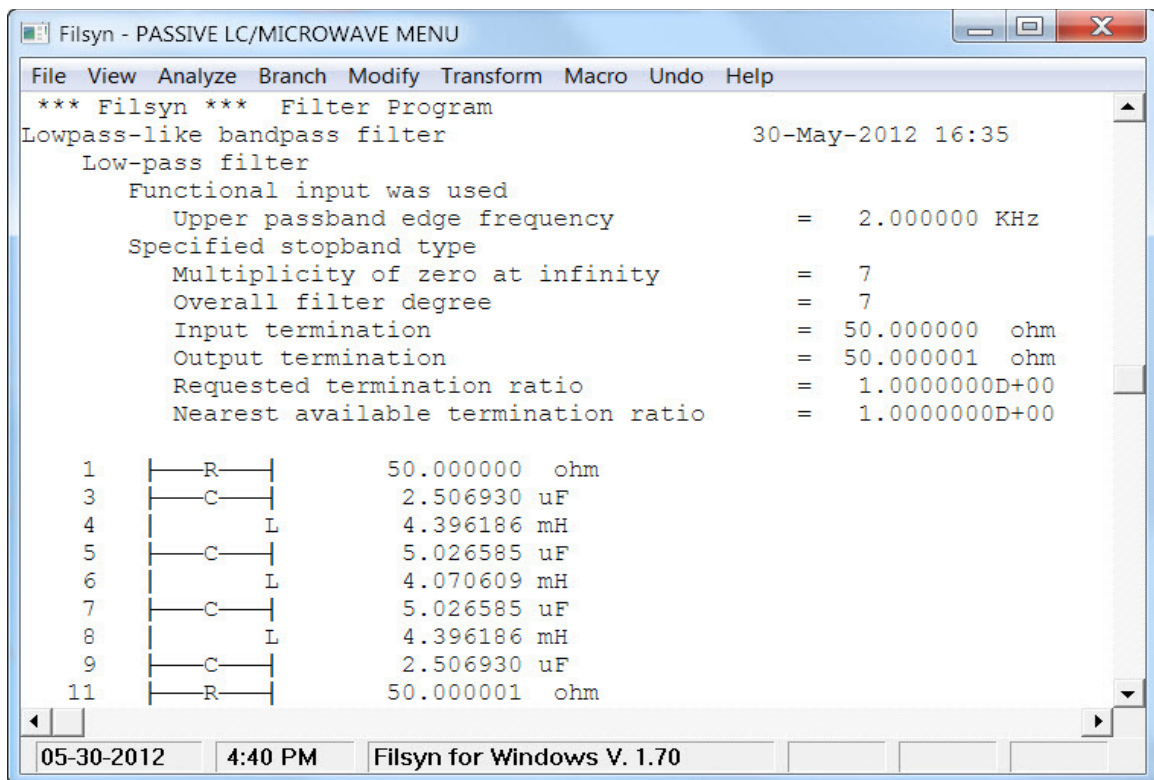
If, instead of using the **filscript.exe**, we enter the **filsyn.exe** directly, the transition ratio will be required for the actual design, which must start as a lowpass filter. We specify this filter with the correct loss specifications, then select an arbitrary passband edge frequency and a stopband edge that is this passband edge times the transition ratio. This transition ratio needs to be computed manually, or alternatively, we may use the **Estimate** script above to do that for us.

When the lowpass design is completed, we can then convert that into the required band-reject filter. The simpler solution is, of course, to use the batch processor **filscript.exe** as we have done above.

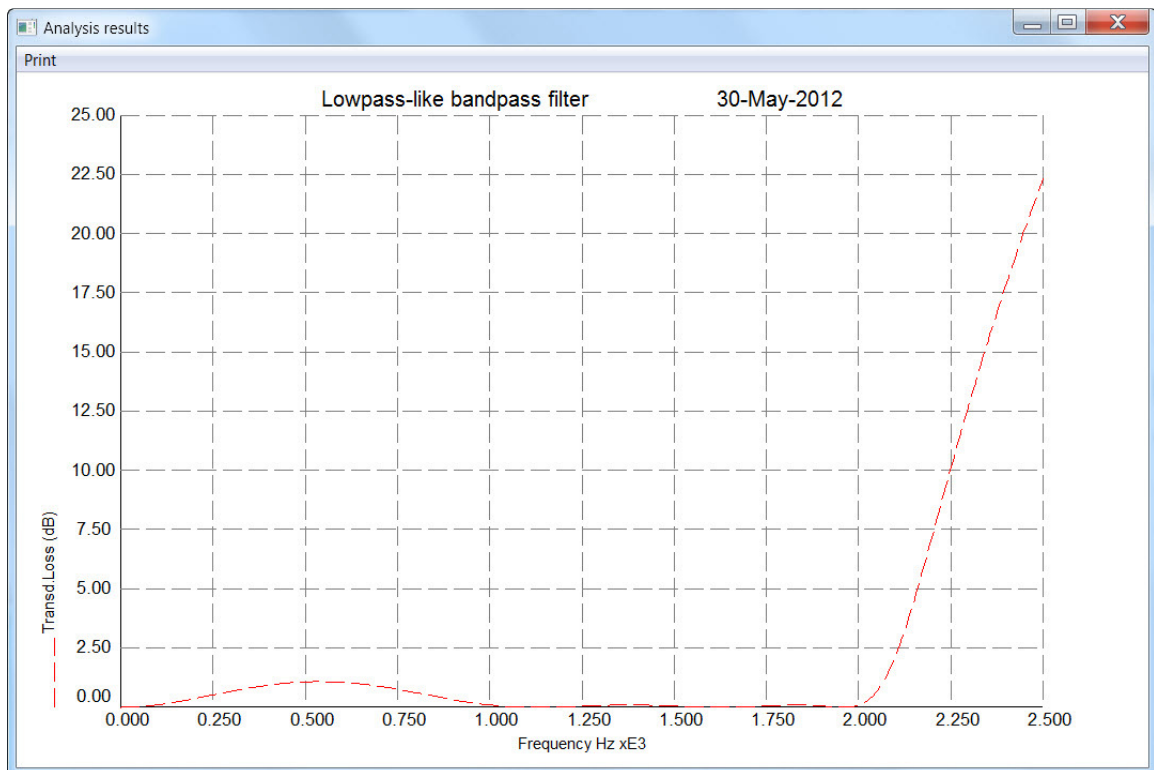
- **Lowpass-like bandpass.** This option generates filters that look like lowpasses, but behave like bandpasses and still have equal terminations. The passband can be maximally-flat or equal-ripple type. The lower stopband usually has not much of an attenuation, but the upper one is higher than a simple lowpass would have. The data entry window is as follows:

Parameter	Value
Filter degree	7
Passband edge loss (dB)	0.10000000
Lower passband edge (Hz)	1.000000E+03
Upper passband edge (Hz)	2.000000E+03
Input termination	50.0000000
Output termination	50.0000000
Filter type	Equal ripple

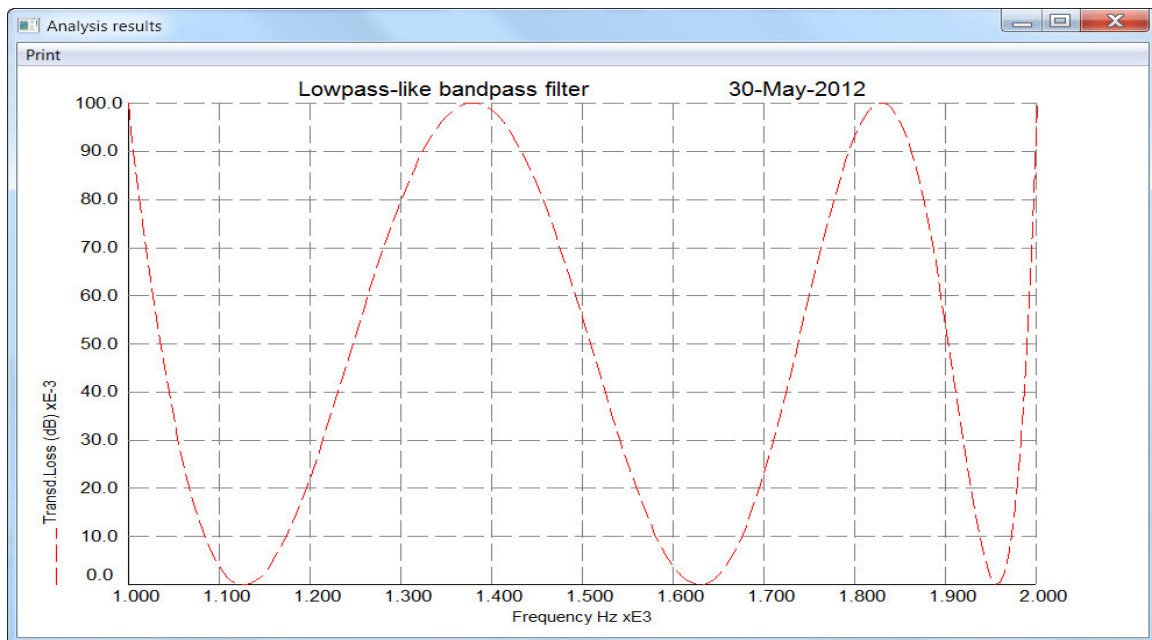
We have selected a seventh order filter (it can be of degree between 5 and 19) and a passband of 0.1 dB ripple from 1 KHz to 2 KHz. The resulting filter is shown next:



which does, in fact, look like a lowpass. The analysis shows the bandpass behavior:

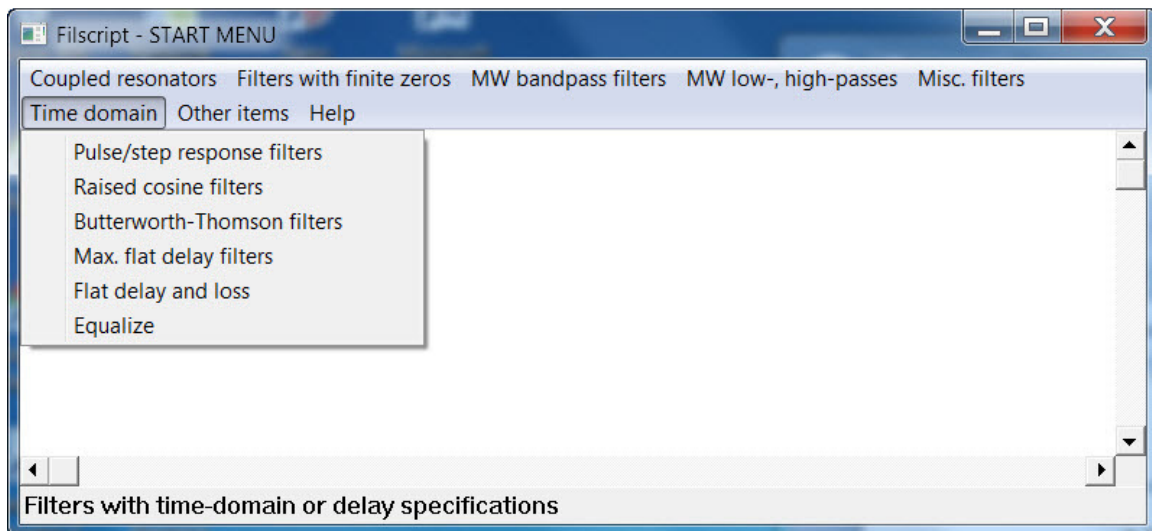


while the passband detail shows the equal-ripple behavior:



1.7 Time domain

The next main option gives us certain filter characteristics related to time domain or time-delay specifications. Some of these can be implemented in passive LC and active RC forms, others in microwave or IIR digital forms. The filters in this group are all essentially lowpass type which sometimes can be converted to highpass or bandpass.

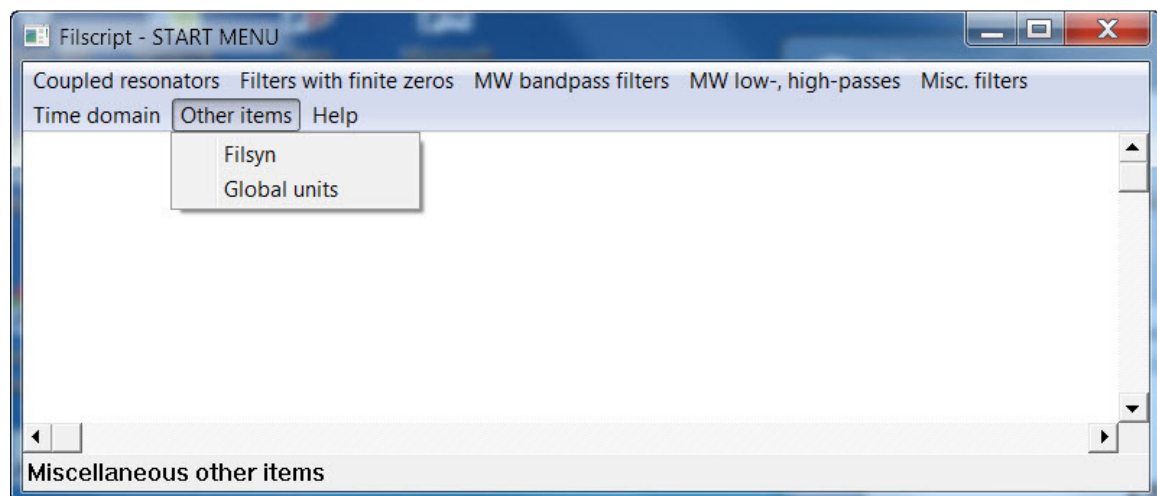


- Specified **Pulse/step response filters**. These are filters with pulse (or step) responses that approximate the ideal response to within 0.5%, 1% or 2% ripple and either a minimum of 40 dB or 60 dB stopband loss. Not all combinations are available, so you must try different ones until you find one that is satisfactory

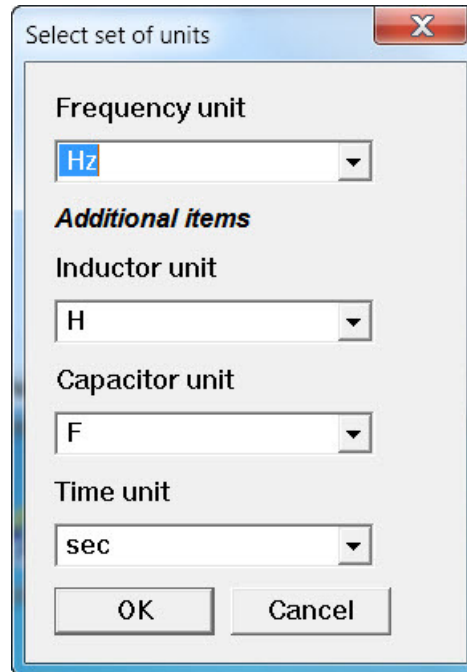
- **Raised cosine filters.** These filters approximate either a raised cosine or the square root of a raised cosine shape over a specified transition bandwidth. The cosine width may vary from 10% to 100% for the raised cosine and from 20% to 100% for the square-root shapes, in 10% steps for both. Passband are close to equal-ripple and stopbands have a minimum of 41 dB guaranteed loss.
- **Butterworth-Thomson filters.** These are the generalizations of the well-known Butterworth-Thomson design. Instead of the Butterworth design, we can also find the transitional design between a Chebyshev and the Thomson (also called Bessel) filters. This is selected simply by specifying a nonzero passband ripple-value. A zero value gives the old Butterworth-Thomson design. The filter degree may be between 2 and 20 and the lowpass filter may be converted to a bandpass using the **Shifted-bandpass** transformation.
- **Max. flat delay filters** These are microwave or IIR (Infinite Impulse Response) digital lowpass or highpass filters with maximally flat delay. The details of this design method are found in *Application Note 15*.
- **Flat delay and loss** These are analog (passive LC or active RC) lowpass filters with flat delay and flat loss behavior. Details are available in *Application Note 16*.
- **Delay equalizer** design. This option is not implemented here because it is directly available from the **Filsyn** program, using the **Design->Delayline/equ** menu item and selecting the **Equalizer** option.

1.8 Other items

Finally, the ‘Other items’ option contains two items, one is the **Filsyn** item, that just calls the main **filsyn.exe** executable program directly, while the new **Global units** item lets us specify units we wish to use.



In order to understand this option, let us remember that for data entry purposes we can generally use either scientific notation (using exponents) or engineering units (using engineering prefixes). Now however we are providing a third method and that is to preselect the units. If we select this menu item, we get the following data input window:



The down arrows provide us with alternative units possible. If for instance, we select the MHz unit for the frequencies here, then when we need to enter 1.2MHz later, we do not enter 1.2e6 or 1.2M, but just 1.2. This then carries over automatically into the **Filsyn** program as well as the postprocessors, if used. All frequency entries and displays will also be in MHz units.

Considering data display, we wish to point out that the program automatically selects units for all elements in such a manner that the displayed value should be between the limits:

$$1.0 \leq \text{value} < 1,000.0$$

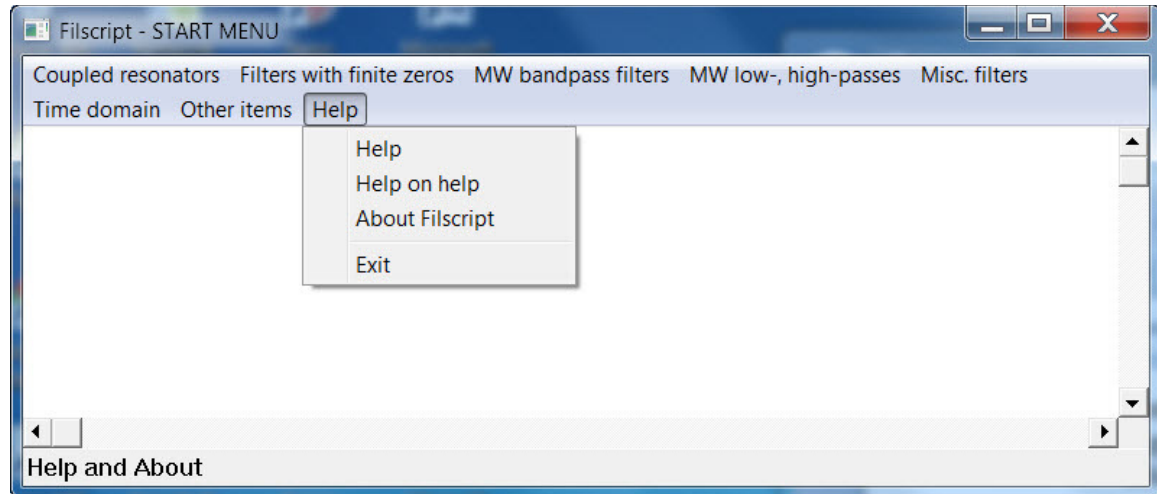
Naturally, this means that the values of different components may be displayed in different units. If we wish to use instead, the same units everywhere, we may select these preferred units under the **Additional items** heading in the above window. We must be careful in selecting alternative units here, since selecting the wrong values may result in values that are either too large or too small and may not be possible to display. In that case the simplest solution is to save the filter in a file and then reset the units and recall the filter. The units can be reset by invoking the prompt again and selecting Hz for the frequency unit. This will reset all other values as well. Another way to reset the units is by deleting the file(s) “__qx” and “__qz” in the Filsyn folder, that contain the selection data for the **Filscrip** and **Filsyn** programs respectively.

The unit selection in the **Filscrip** program is automatically transferred to the **Filsyn** program and they are saved in both. That is to say they are in effect the next time the programs are invoked.

Note that all of this refers to passive LC filters, active RC filters, microwave and IIR digital filters also. However, for microwave and IIR digital filters the **Additional items** selection is immaterial and is ignored.

1.9 Help

The **Help** menu item (below) contains the usual options with the **Exit** option added at the end. Of course, one may exit the program also by either clicking on the **X** at the top right corner or touching the Alt-F4 key combination.



Note that if you have used this menu system to perform a specific design, calling the program again and then aborting the call without calling any of the submenus, will perform the same design again. This is by design in order to enable us to repeat a design without having to reenter all the necessary data. The information is transmitted from the **START MENU** to the main **Filsyn** program through a text file with the name **_xxx.tmp** and this repeat performance can be easily prevented by deleting this file. Alternatively, you can call any of the submenus, then abort the design, which will also delete this file. As another option, you can save the file for later reuse or for documentation purposes under a different name. Even though it is a text file, please do not edit it, because we supply no documentation for its content.

Three more file names are used for data transfer between programs in some of these designs, **xxx.dat**, **xxx.tab**, **xxx.pz** and **_xxx.pz**. Please do not use these file names for any other purpose.

2. INTRODUCTION TO FILSYN

Filsyn is a very powerful filter design and analysis tool that has been available for 35 years in various formats and is now available in a Windows format for personal computers. It runs under any of the 32 or 64 bit Windows operating systems: Win95, Win98, WinME, Win2000, WinXP, Vista and Windows 7 or 8. It requires only about 150 Mbytes of disk storage space and has no further special requirements. Its installation does not modify the registry or any other system files, and therefore can be uninstalled simply by deleting the folder it is installed in. Your display should be set at least to an 800 x 600 screen resolution or better, otherwise you will have difficulty in maneuvering around some of the data input menus.

2.1 Program scope

A complete description of **Filsyn**'s capabilities follows.

a) *Filter Families*

- Passive LC (including crystal)
- Microwave (using commensurate length line segments)
- Active RC (including switched-capacitor)
- Digital IIR (infinite impulse response)
- Digital FIR (finite impulse response)

b) *Filter Types*

- Low-pass
- High-pass
- Linear-phase low-pass
- Band-pass
- Band-reject
- Delay lines and equalizers
- Matching networks

c) *Passband Behavior*

- Maximally flat (loss or delay)
- Equal ripple (loss or delay)
- Sloping pass band in bandpass
- Specified transfer or characteristic function (*functional* input)

d) *Stopband Behavior*

- Monotonic stop band (degree unknown)
- Equal minima
- Specified (known transmission zeros)
- General (specified stop band requirements)

In the last case, the **Placer** option is used to determine the number and locations of the transmission zeros needed to meet the stop band requirements, assuming a maximally flat or equal-ripple type pass band behavior.

e) Available Structures

Passive LC - Ladder (may include bridged T's, twin T's and/or cross-coupling)
- Lattice (single section)
- Ladder-lattice combinations

Active RC - Cascaded 2nd order sections
- Leapfrog feedback structure
- Follow-the-leader feedback structure

Digital IIR - Cascaded 2nd order sections
- Parallel 2nd order sections
- Direct form
- Gray-Markel lattice (2-multiplier)
- Differential allpass
- Various lattice and ladder wave-digital structures

Digital FIR - Single structure
- Cascaded structure

Microwave - Ladder (may include bridged T's, twin T's and/or cross-coupling)

Alternative active-RC filter structures may be derived from the passive ladder equivalents.

f) Terminations

(The following information does not apply to active RC and digital filters.)

Passive and microwave filters may operate between any pair of specified resistive terminations, including an open- or a short-circuit at one end (this is used mainly for multiplexer design). If unequal terminations are requested for a low pass or a lumped LC high-pass, additional flat pass band loss results. In band-pass cases (and microwave high-pass cases containing unit elements) an arbitrary termination ratio can usually be accommodated, regardless of specifications, with no additional flat loss.

Filters can also be designed to match two unequal resistive terminations with no reference to stop band requirements (matching networks). This can be accomplished using any of the following:

- a lumped LC band-pass that looks like a low pass,
- a microwave high-pass consisting of cascaded, quarter-wave line segments (unit elements),
- a microwave band-pass that looks like a low pass.

g) Other Options

A parametric design is available for all band-pass filter types (see *Appendix C* for the explanation of parametric filters). Such a design is useful primarily in passive LC realizations, because it can provide a structure with the minimal number of inductors.

Predistortion is available to pre-compensate for (assumed uniform) component dissipation in lumped passive LC realizations.

A parameter (the integer *ZS*) is available to control the behavior at low and high frequencies of even degree low- and high-pass filters respectively (see *Appendix D*).

Complex transmission zeros can be specified if the user enters the locations of all transmission zeros. This may improve the pass band delay characteristics of the filter but will lead to the presence of bridged-T or twin-T sections in the structure. Some active RC realizations may not be used with complex zeros (quadruplets).

Transmission zeros with arbitrary multiplicities may be specified.

If the reflection coefficient zeros are complex, they may, in certain cases, be placed arbitrarily in either the right or the left half of the *s*-plane (see *Appendix E*).

Functional input allows us to enter the transfer function or the characteristic function directly into the program. This is useful if a specified non constant pass band shape or a specified phase or delay characteristics is required and an approximating function has been obtained from another source.

For band-pass filters, we may specify a slope in the pass band loss shape; this is useful for interstage network design.

Nearly all combinations of these performance characteristics are available, hence the number of variations and combinations is staggering and cannot possibly be illustrated.

With the help of script files that are now built into the **filscrip.exe** executable, we can also design Gaussian filters, Papoulis polynomial filters, transitional Butterworth-Thomson filters, raised-cosine filters, filters meeting various specifications in both the time- and frequency-domains and many others.

Table 1.1 below shows how some of the popular filter types can be obtained from **Filsyn**. Filter degrees are restricted to 50 or less, except for FIR digital filters, where the limit currently is 512 taps.

The **Placer** option is needed in general stop band specifications. It can also be used for equal-minima type stop-bands, regardless of whether a closed form solution exists or not. If a closed form solution (like an elliptic type low pass) exists, the use of the **Placer** option will yield the same results within ± 1 degree, when the same requirements are specified. However, in a number of cases, closed form solutions do *not* exist, even for

equal-minima type stop bands, and even when they exist, they may not yield the optimal (simplest) solution. Parametric band-pass filters and microwave filters containing unit elements fall into the category of not having explicit solutions available and **Placer** must be used to obtain a solution. In most band-pass filter problems, including cases of uniform stop band requirements, the solution supplied by the **Placer** option will generally be simpler and more economical, than the one obtained without it.

SPECIFY:

NAME:	PASSBAND:	STOPBAND:
Butterworth	Maximally flat loss	Monotonic
Chebyshev	Equal ripple loss	Monotonic
Inverted Chebyshev	Maximally flat loss	Equal minima
Elliptic (Cauer)	Equal ripple loss	Equal minima
Bessel	Max. flat delay	Specified/monotonic
Modified Bessel	Max. flat delay	Equal minima
Ulbrich-Piloty	Equal ripple delay	Specified/Equal minima
General parameter	Maximally flat loss	Specified
	Maximally flat loss	Placer
	Equal ripple loss	Specified
	Equal ripple loss	Placer

Table 1.1. How to Specify Your Favorite Filter Type

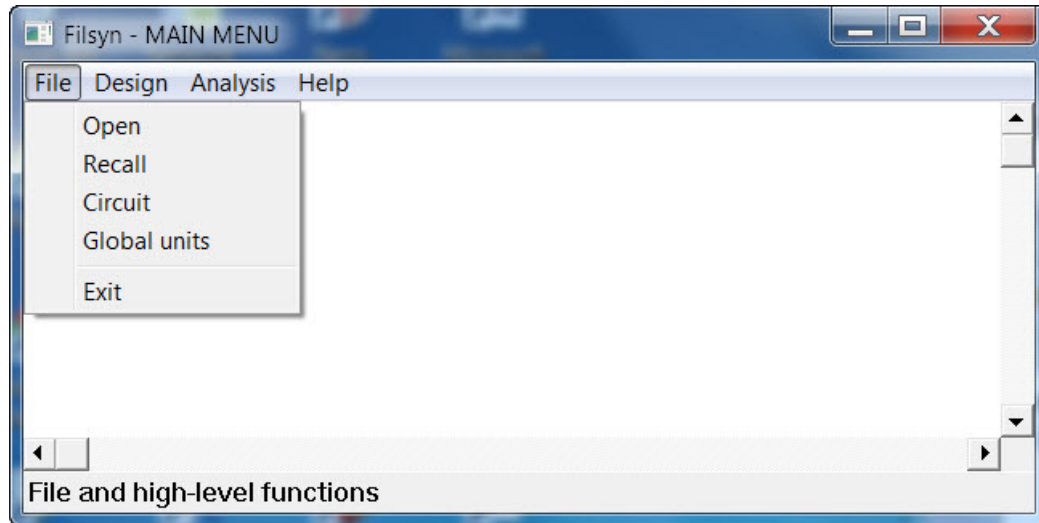
h) *Files*

Filsyn reads and writes a number of files, for saving various data sets and possibly rereading them as well as for communication between executable programs. When writing a file, you will be able to specify the name and the location of the file *except* the last 3-character extension; that will be specified by the program. This is done to enable the program to detect what the file contains and what it should do with it. Most of the files the program can read are binary files, which cannot and should not be edited. These have extensions beginning with the letter 'q'. Some temporary files are text files, but even then do not edit them if the program can read them. Files that will not be read back are always text files and used for purposes of documentation and naturally may be edited as desired. Files that we write to be read into the program are all text files and should have either .dat or .req extensions. If tabulated data is to be entered into the program, it can usually be entered either from the keyboard or read in from a file.

2.2 Starting the program

2.2.1 Main menu

Upon calling the program (double clicking the **filsyn.exe** file or the corresponding shortcut on your desktop) we face a very simple starting menu screen:

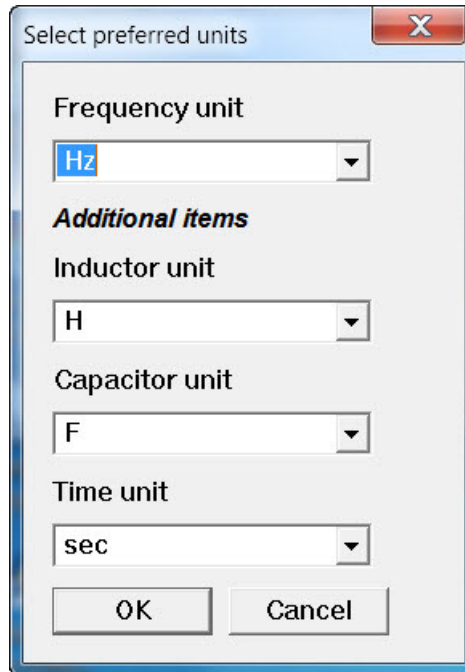


The bottom toolbar displays some information, including date and time. When we click on a menu item, the toolbar displays a brief explanation as to the role of the selected item.

The first, **File** item has five submenu items:

- **Open** will open any text file and display its contents (but does not edit it).
- **Recall** will recall a previous filter design that we have saved in a file. Both of these items use a standard Window file selection menu to select an appropriate file anywhere on your system. Note that using the **Recall** menu item, you will be able to select one of only those files that contain a design saved in **Filsyn**.
- **Circuit** please ignore this item.
- **Global units** allows us to select preferred units. As you can observe from this manual, we may specify data to be entered into the program either in scientific format, using exponents, or in engineering formats, using engineering prefixes.

However, there is a third way and that is to select the preferred units at the start. If we click on this option, we open the following data entry menu:



Clicking on the down arrows shows the available alternative units. For instance, we may select here the MHz frequency unit. This means that in the rest of the program all frequencies will be interpreted as in MHz units. Hence if we wish to enter the frequency 1.2 MHz, we do not enter it as 1.2e6 or as 1.2M, but just as 1.2. The frequency display will also be in MHz units.

Considering data display, the program automatically selects units for all elements in such a manner that the displayed value should be between the limits:

$$1.0 \leq \text{value} < 1,000.0$$

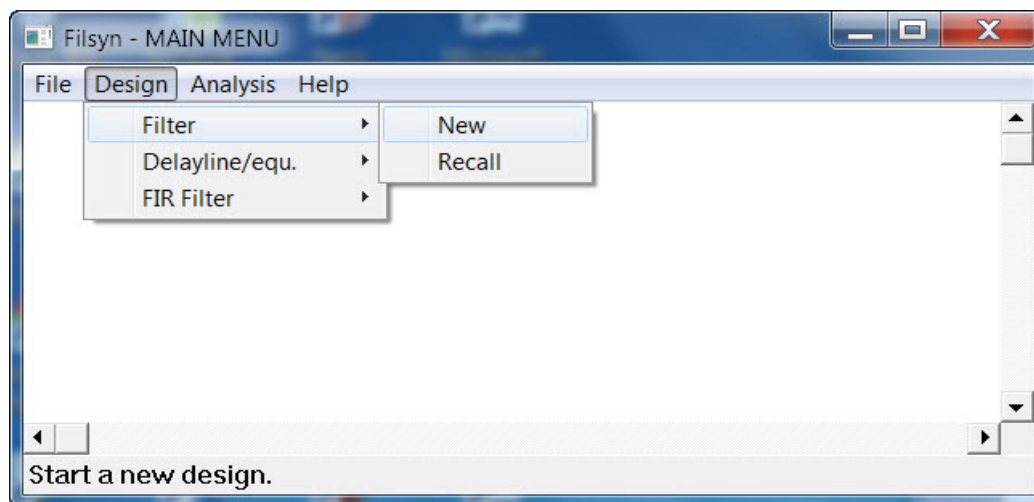
Naturally, this means that different elements may be displayed in different units. If we wish to use instead, the same units everywhere, we may select these preferred units under the 'Additional items' heading in the above window. We must be careful in selecting units here, since selecting the wrong values may result in values that are either too large or too small and may not be possible to display. In that case the simplest solution is to save the filter in a file and then reset the units and recall the filter. The units can be reset by invoking the prompt again and selecting Hz for the frequency unit. This will reset all other values as well. Another way to reset the units is by deleting the file(s) “__.qxx” and “__.qxz” in the **Filsyn** folder, that contain the selection data for the **Filscript** and **Filsyn** programs respectively.

The unit selection in the **Filscript** program is automatically transferred to the **Filsyn** program and they are saved in both. That is to say, they are in effect the next time the programs are invoked.

Note that all of this refers to passive LC filters, active RC filters, microwave and IIR digital filters also. However, for microwave and IIR digital filters the **Additional items** selection is immaterial and is ignored.

- Finally **Exit** terminates program execution.

The **Design** menu option is to be used to start a new design:

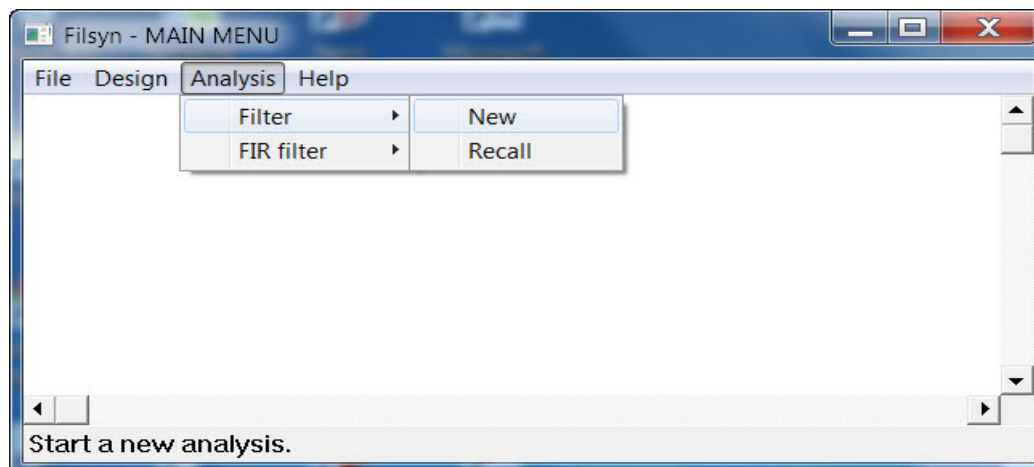


Here we have three submenu items:

- **Filter** that starts all filter designs (except FIR filters)
- **Delay line/equ.** that starts the design of a delay line, a delay equalizer or a linear-phase filter and lastly
- **FIR filter** will start the design of a new FIR digital filter.

In all of these cases, we have the further option of selecting either **New**, or **Recall**. The second option is to be used, if a previous filter specification has been saved before and we wish to rerun the design with possibly some modifications of the specifications. Otherwise we use the **New** menu item.

The **Analysis** top menu option has a similar set of submenu items:



These are to be used when a filter has been obtained from some other source and we wish to enter it into **Filsyn** for analysis and/or possible modifications. Again we may start a new data entry procedure or recall a previously entered data set.

Finally, the **Help** menu item contains the usual submenus.

2.2.2 New filter

Selecting the **Design->Filter->New** option is where most of our designs start and brings up the large <New Filter> data entry screen:

Here is where we enter the specification for our new design.

Most of the items on this screen will be familiar for filter designers. Data may be entered either in scientific or engineering notation where appropriate. However, please pay attention to cases; 'p' (for pico), 'n' (for nano), 'u' (for micro), and 'm' (for milli) must all be lower case, while 'K' (for kilo), 'M' (for mega), 'G' (for giga) and 'T' (for tera) must all be capital. Also the **Enter** key will activate the **OK** button immediately, so be careful. If frequency unit was selected through the **Global units** option, units or exponents are not needed for frequencies any more.

Also, please enter the data in the sequence it is asking for with only a few exceptions. Normally the earlier entries will modify the later ones; some initially grayed out entries will become active and vice versa. The most significant item is the initially grayed out **Detail parameters** button, that becomes active, if we need to enter transmission zeros or functional data. Clicking on that button then will bring up the appropriate data entry

screen. Entering data in this tabular data entry screen is a bit different, in that the **Enter** key will *not* activate the **OK** key, it will just move you around the screen.

The **Shifted bandpass trans.** option is activated, when you enter a nonzero center frequency into the editbox below it. This operation is active only for low-pass filters and will be discussed in detail later. Finally the **Save design data** checkbox enables us to save the specification in a file. We can then repeat the design by recalling and possibly modifying the specification without having to reenter all the information. If the box is checked, a filename selection menu will open before the design proceeds.

If we select **Design->Filter->Recall** menu sequence, we can select a file containing the saved set of specifications and the same **<New Filter>** window opens with the data already entered and ready for modification and/or a rerun.

2.2.3 New delay line

Selecting the **Design->Delay line/equ.->New** menu sequence leads to the next menu below. Again the **Recall** option is no different, except for the fact that the saved specification will show in the window, instead of the default settings:

<New Delay-Filter/-Line/-Equalizer>

Delay type

- ☒ Max. flat
- ☐ Equal ripple
- ☐ Equalizer

Implementation

- ☒ LC elements
- ☐ Active RC
- ☐ Microwave
- ☐ IIR digital

Data

FQ or FS (Hz) 0.00000

Normalization freq (Hz) 0.00000

Network type

- ☒ Filter
- ☐ Delay line

Delay value (sec) 0.00000

Stopband

- ☐ Equal min.
- ☒ Specified

End of passband (Hz) 0.00000

Start of stopband (Hz) 0.00000

Degree 0

Zeros at infinity 0

Transm. zeros

Terminations 50.000000E+00 50.000000E+00

Q-value 0.00000000

Hurwitz type

- ☒ Hurwitz
- ☐ Non Hurwitz

Selection

- ☒ Auto
- ☐ Manual

Indicators

Shifted bandpass trans.

Center frequency (Hz) 0.00000

☐ Odd parametric

☐ Save design data

OK Cancel

Most of the items here will be familiar again to designers. The **Filter** type will yield a linear-phase low-pass (or band-pass, if the **Shifted bandpass trans.** option is activated), otherwise we get a low-pass (or band-pass) delay line. The **Hurwitz type** radio button and additional associated data should normally be left at their default settings; they are there for the infrequent case when a realizable passive LC implementation cannot be found otherwise. Both filters and delay lines may be implemented only in passive LC or active RC forms. This will also be dealt with in detail later.

The last significant option is the **Equalizer**, which permits us to design a delay equalizer either to equalize a tabulated set of delay versus frequency values, or to implement a set of given equalizer poles in any of the four possible implementations indicated. The data again may be entered from the keyboard or read in from a text file. The file must have a .tab extension if it contains tabulated delay versus frequency data, .dat or .pz if it contains other data or the natural modes (poles) of the delay equalizer.

2.2.4 FIR filter design

The last design option is **Design->FIR filter->New** (or **Recall** as before). This leads to the following data entry screen:

Again, filter designers will be familiar with all the items available here. The **Equal ripple** option is, in fact, the familiar Remez technique of FIR filter design, and the **Details** button will open a window for entering the specification for up to ten bands.

The **Windowed** option, on the other hand, activates the choice of more than 20 types of windows.

2.2.5 Filter analysis

Selecting next the **Analysis->Filter->New** menu, we get to the following window, where we are able to enter the information of an existing filter:

Entering filter data

Filter kind

- ☒ Passive LC
- ☐ Microwave
- ☐ Active RC
- ☐ Switched capacitor
- ☐ IIR digital

Filter type

- ☒ Lowpass
- ☐ Highpass
- ☐ Bandpass
- ☐ Band reject
- ☐ Delay equalizer

Structure

- ☒ Cascade
- ☐ Follow-the-leader
- ☐ Leapfrog

Form

- ☒ Roots
- ☐ Factors

Sampling/Q.W. freq (Hz)

0.00000

Lower passband freq (Hz)

0.00000

(Upper) passband/Norm. (Hz)

0.00000

Poles Zeros

Numerator Denominator

Feedback coeff.

No. of zeros at infinity

0

☐ Save analysis data

OK Cancel

Note that the complete filter data can be entered here for all but the **Passive LC/Microwave** filter kind. For those, we only enter here a few pieces of information, the rest (the actual configuration and element values) will be entered in the analysis segment of the program, which we shall describe later.

2.2.6 FIR filter analysis

If we need to analyze an existing FIR digital filter, we use the **Analysis->FIR filter>New** (or **Recall**) menu item:

Entering FIR filter data

Sampling freq. (Hz)

0.00000

Filter data

☒ Coefficients

☐ Quadratics

Filter symmetry

☒ Even

☐ Odd

☐ None

Filter length

0

Data

☐ Save analysis data

OK Cancel

The coefficients or zeros are entered through the **Data** button, which opens the appropriate data input screen.

2.2.7 Examples

As the first example, consider an analog (LC or active RC) low-pass filter with an equal-ripple passband of 0.1 dB loss ripple up to 1.5 KHz and an equal-minima type stopband beginning at 1.55 KHz and having at least a 45 dB loss. In addition, we must have a triple zero at infinite frequency. Because of this last item, we cannot use the standard equal-minima stopband specification, we must use instead the **Placer** option for the design. The complete data entry screen will look as follows:

<New Filter>

Filter kind

- ☒ LC
- ☐ Microwave
- ☐ Active RC
- ☐ IIR Digital

Quarter-wave/Sampling freq(KHz)

0.00000

Filter type

- ☒ Low pass
- ☐ High pass
- ☐ Band pass

Lower passband freq (KHz)

0.00000

Upper passband freq (KHz)

1.50000E+00

Passband type

- ☐ Max. flat
- ☒ Equal ripple
- ☐ Functional
- ☐ Sloping

Band-edge loss/return loss (dB)

0.100000000

Loss Slope (dB/oct)

6.00000000

Flat loss (dB)

0.00000000

Function Type

- ☒ E
- ☐ F

Multiplier

0.00000000

Stopband

- ☐ Monotonic
- ☐ Equal min
- ☒ Placer
- ☐ Specified

Lower stopband freq (KHz)

0.00000

Loss (dB)

50.00000000

Upper stopband freq (KHz)

0.00000

Loss (dB)

50.00000000

Detail Parameters

of zeros at zero

0

of zeros at infinity

3

of unit elements

0

Parametric

- ☒ Conventional
- ☐ Parametric
- ☐ Matching

R1

50.000000E+00

R2

50.000000E+00

ZS parameter

-1

Q for predistortion

0.00000000

Predistortion

- ☒ Passband only
- ☐ Pass/stop bands

Shifted bandpass trans.

Center frequency (KHz)

0.00000

☐ Odd parametric

☒ Save design data

OK Cancel

Note that even though we used engineering notation for entering the data, once you leave the editbox, the program displays the data in scientific notation.

The **Band-edge loss/return loss** window entry is interpreted as insertion loss if the value is between 0.0 and 10.0, as return loss if it is greater than 10.0 and as percentage reflection coefficient if it is negative. The last two interpretations are valid only for passive LC/microwave filters.

The **Detail parameters** box contains the following data:

The Filter Parameters dialog box contains three tables for specifying filter requirements and zeros.

Requirements	
Frequency (KHz)	Loss (dB)
1	1.550000E+00
2	0.000000
3	0.000000
4	0.000000
5	0.000000
6	0.000000
7	0.000000
8	0.000000
9	0.000000
10	0.000000
11	0.000000

Fixed zeros	
Frequency (KHz)	Multiplicity
1	0.000000
2	0.000000
3	0.000000
4	0.000000
5	0.000000
6	0.000000
7	0.000000
8	0.000000
9	0.000000
10	0.000000

Movable zeros	
Frequency (KHz)	Multiplicity
1	0.000000
2	0.000000
3	0.000000
4	0.000000
5	0.000000
6	0.000000
7	0.000000
8	0.000000
9	0.000000
10	0.000000
11	0.000000

Below the tables, there are two input fields for the number of zeros:

or no. of zeros below passband:

and/or no. of zeros above passband:

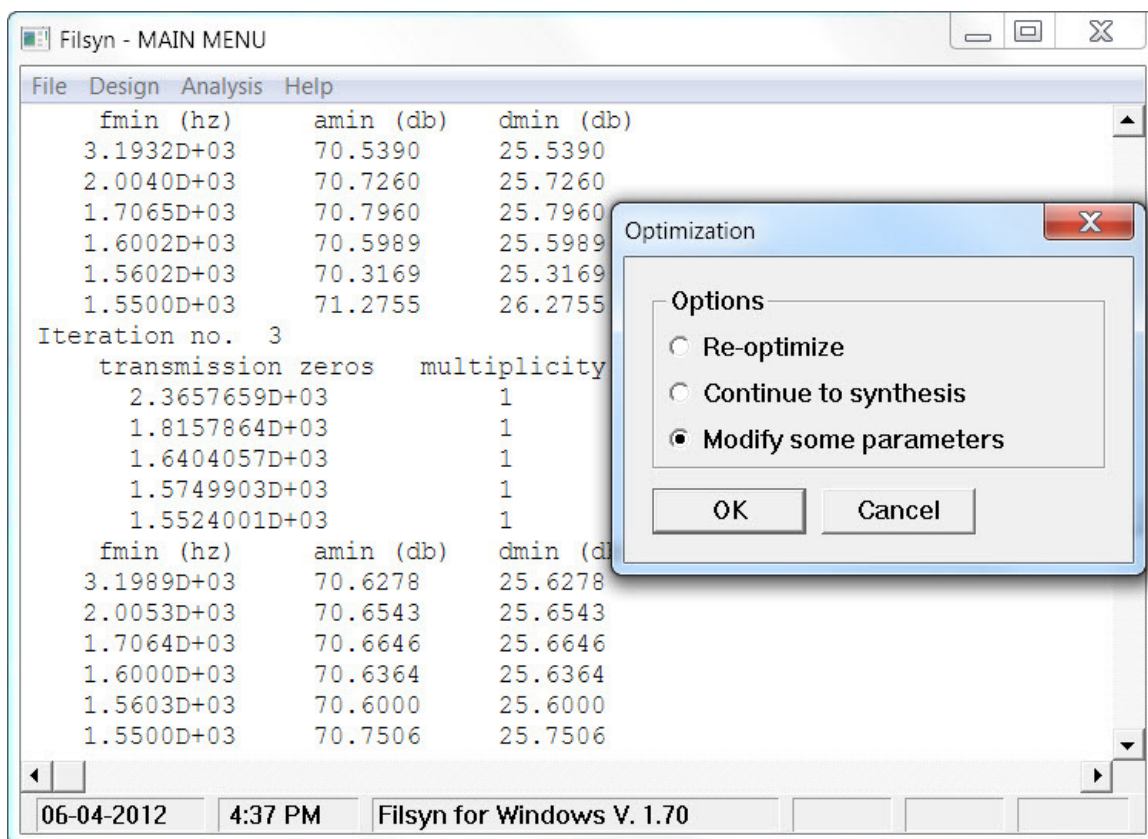
Buttons: OK, Cancel

We saved this specification in a file called *Example 1*, to which the program appended the '.qfi' extension:

The File Explorer window shows the 'Data' folder on the 'Second disk (D:)'. The file list includes 'Documents', 'Tables', 'Text files', and 'jpeg.zip'. The 'File name' field contains 'Example 1' and the file type is set to 'All Files (*.*)'. The 'Open' button is highlighted.

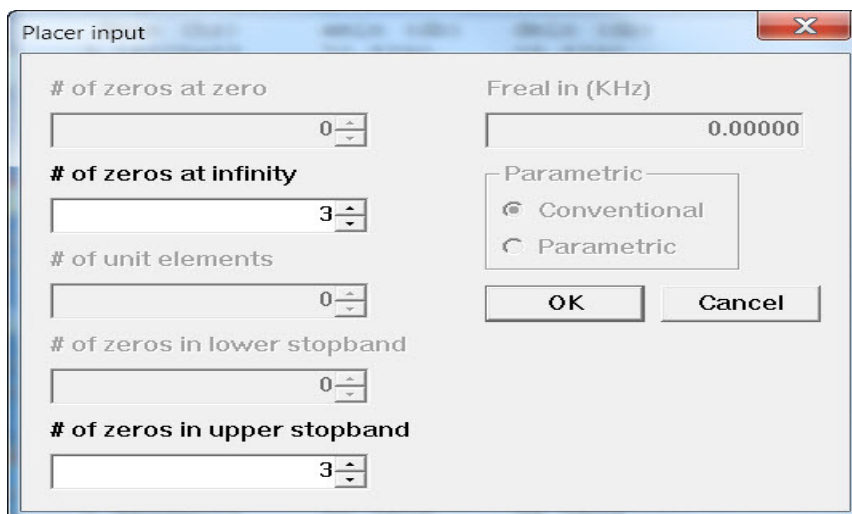
Note that the number of finite transmission zeros below and above the passband need not be entered. If we leave the zeros, the program will estimate the number of zeros needed.

Once the **OK** button is clicked, the program starts the computations and stops in the middle of the **Placer** run to ask:



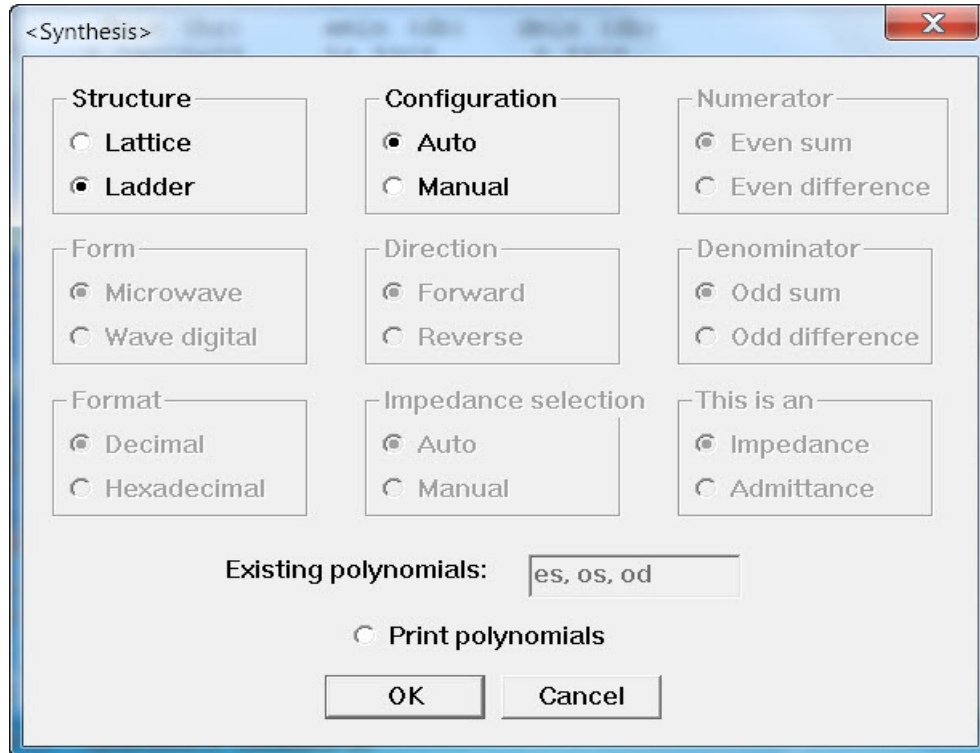
The numbers in the first column are the locations of the loss minima in the stopband(s), the second column contains the values of the loss minima, while the last column shows us the surplus loss.

The program has selected five movable transmission zeros, but as the last 6 lines of the tabulation indicates, the loss is 25 dB higher than what we need. We should therefore reduce the number of transmission zeros by selecting **Modify some parameters** in this submenu. Incidentally, if the submenu obscures some of the data, it can be readily moved out of the way. Our next step is to specify only three movable zeros:



Continuing, the program finds that three zeros are not enough and hence increased that (automatically) to four, which is just fine. We again see the **Optimization** prompt window and selecting the **Continue to synthesis** radio button, we get to the **<Synthesis>** window shown below.

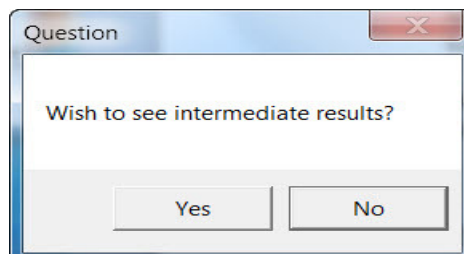
This is quite an elaborate screen, but the default option is basically the most often used, leading to a ladder implementation with the configuration selected by the program itself.



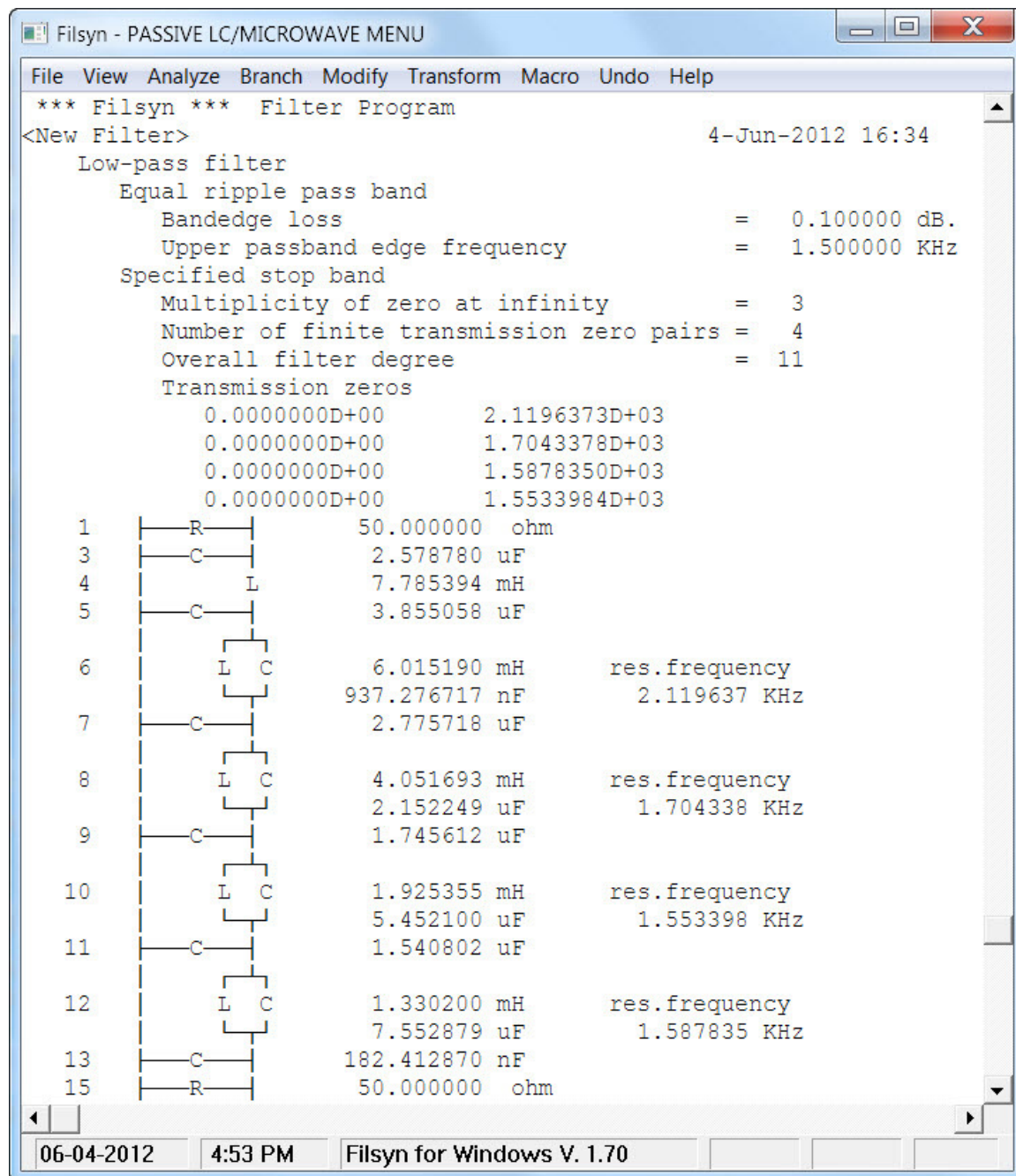
The items at the bottom of the screen make it possible to look at the immittance functions used for the manual synthesis. The manual configuration selection is quite complex and will need a substantial amount of explanation (see *Appendix B*). This synthesis method is seldom used and only when the computer-generated implementation is not realizable.

Please also note that due to the large number of ways we are able to change ladder elements, any starting ladder structure can be changed into any other possible ladder implementation.

Selecting the default settings of a ladder structure and letting the computer specify the exact configuration, the next prompt is:



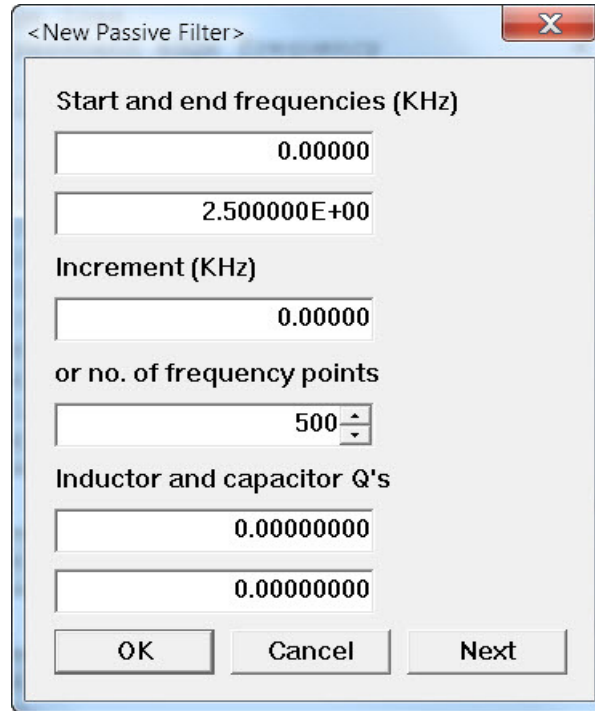
These intermediate results are also rarely needed. Declining them (the default) finally leads us to the results:



We are now in the passive-LC/microwave filter analysis segment of the program with an elaborate menu system. At this time we shall demonstrate only a couple of items, the first is the frequency domain analysis, selected by the **Analysis->Frequency** option, leading to the data input screen on the next page.

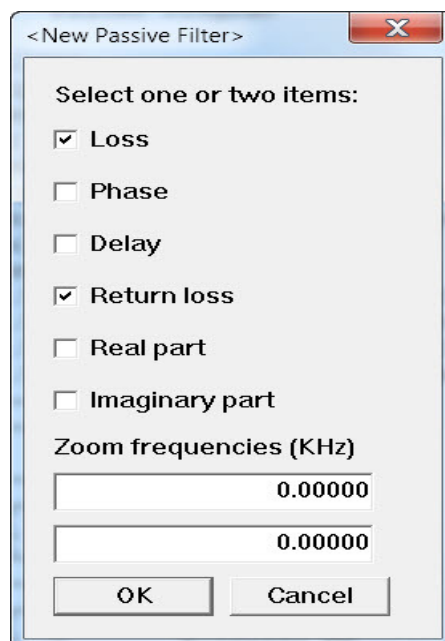
Please note that in the latest version (Version 1.70 or later) of the program you may specify up to five ranges of frequencies, either linear or logarithmic. The logarithmic

range is specified by a negative value in the **no. of frequencies** window and it is then interpreted as the number of frequencies per decade. The **Next** button (see below) is to be used if more than one range is required. Also note the global frequency unit (KHz) that is selected.



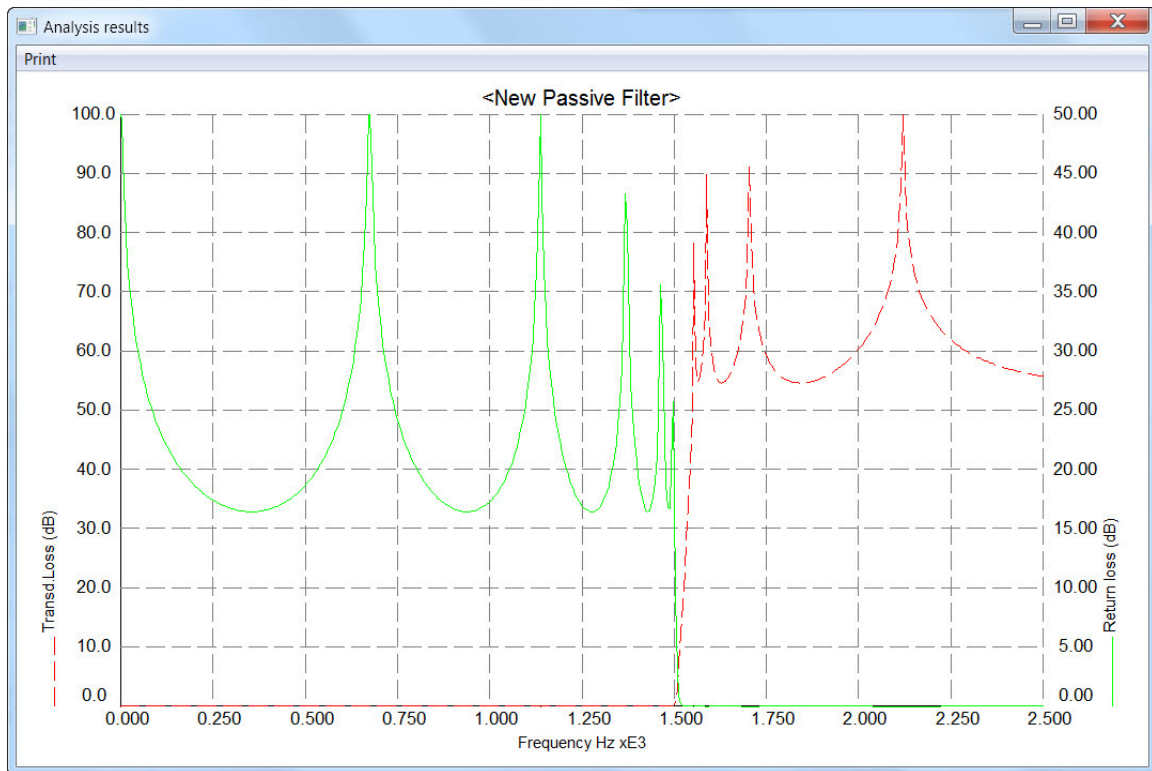
The screenshot shows a dialog box titled "< New Passive Filter >". It contains several input fields and buttons. The first section is "Start and end frequencies (KHz)" with two text boxes containing "0.00000" and "2.50000E+00". Below this is "Increment (KHz)" with a text box containing "0.00000". Then, "or no. of frequency points" with a text box containing "500" and a small up/down arrow icon. Next is "Inductor and capacitor Q's" with two text boxes, both containing "0.00000000". At the bottom are three buttons: "OK", "Cancel", and "Next".

The maximum number of frequencies is 501. Clicking on the **OK** button, we are asked if we need the results tabulated (displayed on the screen), which we usually do not. Instead we wish to see a plot, which we can get by selecting the **Analysis->Plot->Frequency** menu option:

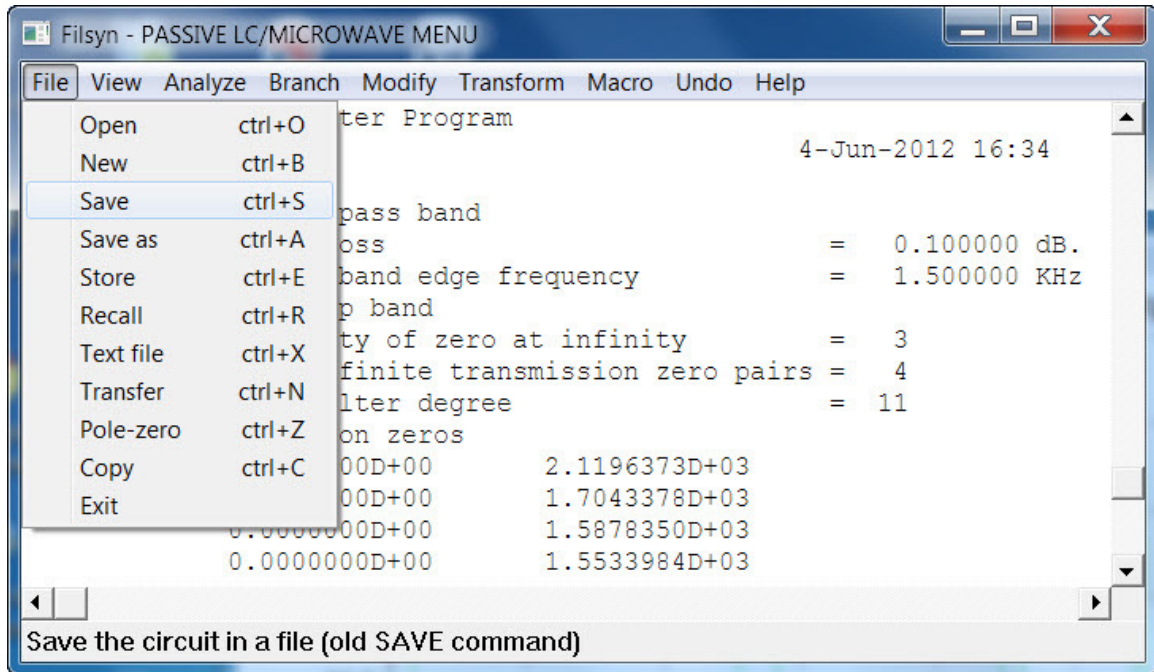


The screenshot shows the same dialog box, but now it has a section titled "Select one or two items:" with several checkboxes. The "Loss" and "Return loss" checkboxes are checked, while "Phase", "Delay", "Real part", and "Imaginary part" are unchecked. Below this is "Zoom frequencies (KHz)" with two text boxes, both containing "0.00000". At the bottom are two buttons: "OK" and "Cancel".

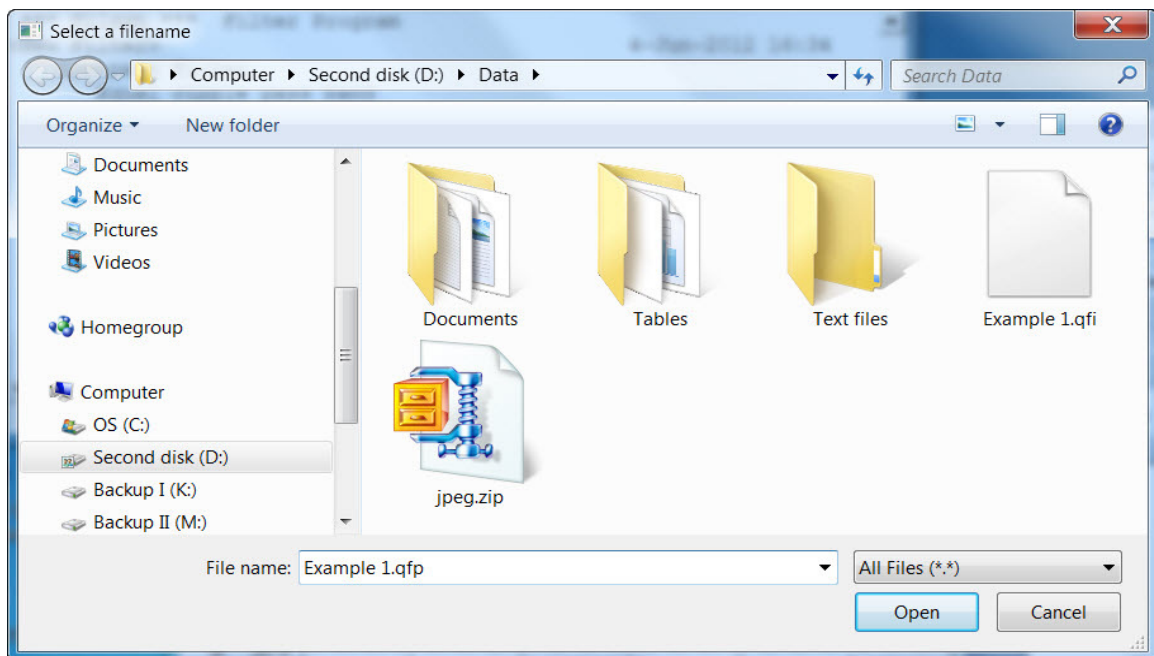
Selecting the loss and the return loss as shown above (up to two items can be selected at a time), we get the following display:



This can be printed and any other computed characteristic can be plotted, or the analysis results can be saved to a file. Selected areas may be plotted by specifying the zoom frequencies above. We just wish to show how to save the filter itself to a file, using the **File->Save** menu option:



This opens up a standard Windows file name menu, where we can select the name as well as the folder we wish to put the file in. We again selected the name *Example_1* to which the program appended the '.qfp' extension.



3. PASSIVE LC/MICROWAVE DESIGN

For those of our users, who are accustomed to older versions of **Filsyn**, the design of linear-phase filters are now available under the **Design->Delay line/equ.** menu option.

3.1 Normal Design

As an introduction, consider a few examples. We start with a standard elliptic bandpass, with all requirements visible in the data input screen:

<New Filter>

Filter kind

- ☒ LC
- ☐ Microwave
- ☐ Active RC
- ☐ IIR Digital

Quarter-wave/Sampling freq (Hz)

0.00000

Filter type

- ☐ Low pass
- ☐ High pass
- ☒ Band pass

Lower passband freq (Hz)

10.000000E+03

Upper passband freq (Hz)

20.000000E+03

Passband type

- ☐ Max. flat
- ☒ Equal ripple
- ☐ Functional
- ☐ Sloping

Band-edge loss/return loss (dB)

0.100000000

Loss Slope (dB/oct)

6.00000000

Flat loss (dB)

0.00000000

Function Type

- ☒ E
- ☐ F

Multiplier

0.00000000

Stopband

- ☐ Monotonic
- ☒ Equal min
- ☐ Placer
- ☐ Specified

Lower stopband freq (Hz)

9.000000E+03

Loss (dB)

50.0000000

Upper stopband freq (Hz)

22K

Loss (dB)

50.0000000

Detail Parameters

of zeros at zero

0

of zeros at infinity

0

of unit elements

0

Parametric

- ☒ Conventional
- ☐ Parametric
- ☐ Matching

R1

50.000000E+00

R2

50.000000E+00

ZS parameter

-1

Q for predistortion

0.00000000

Predistortion

- ☒ Passband only
- ☐ Pass/stop bands

Shifted bandpass trans.

Center frequency (Hz)

0.00000

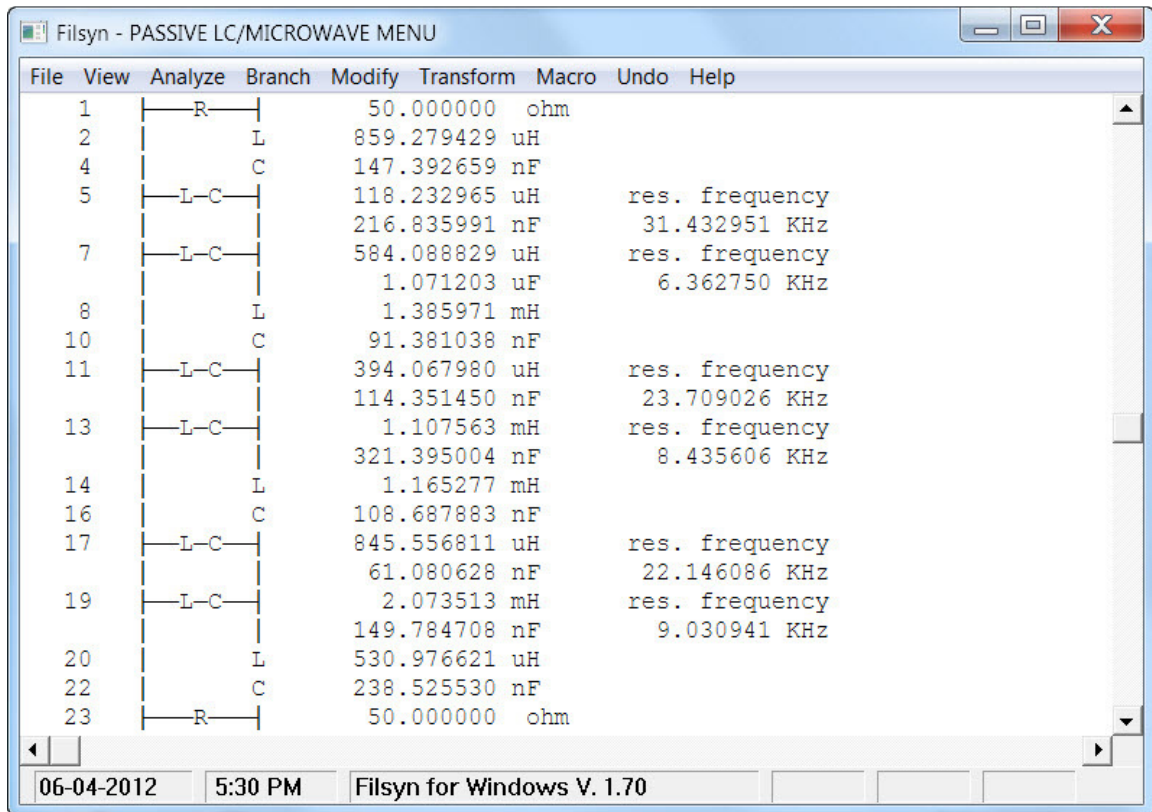
☐ Odd parametric

☐ Save design data

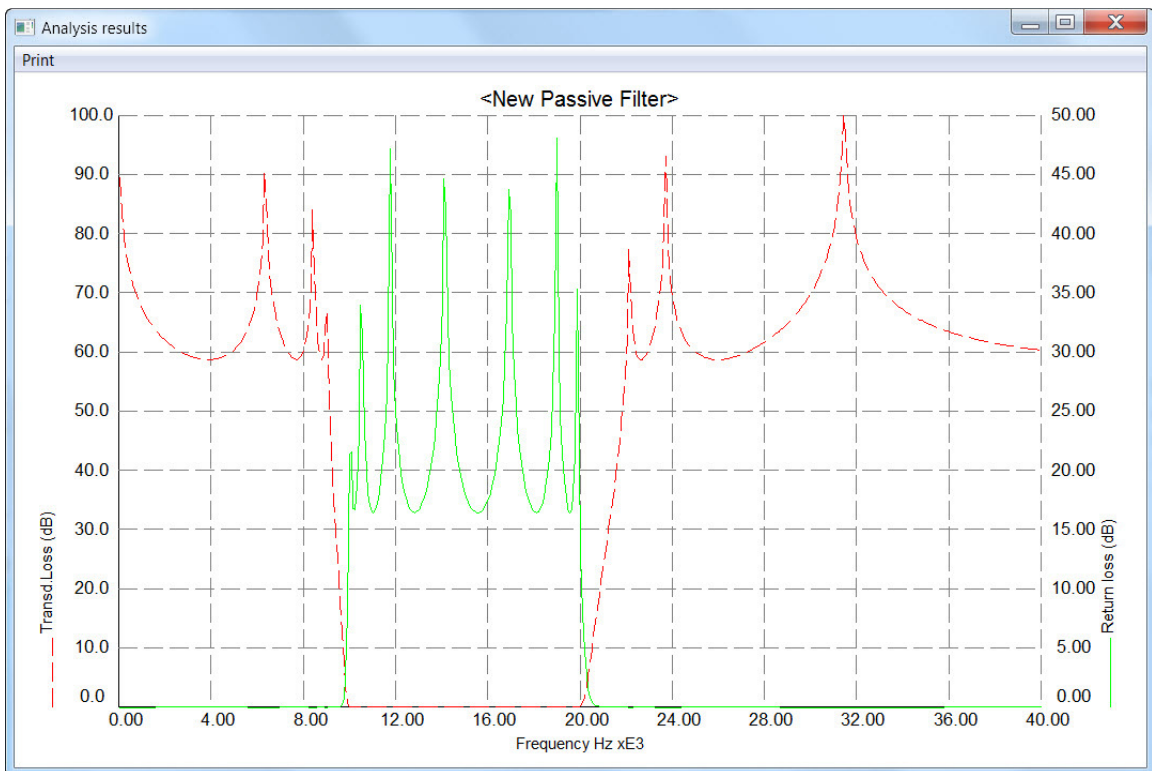
OK Cancel

This filter type, a bandpass with equal-minima type stopband, will be generated from a lowpass by the familiar lowpass-to-bandpass transformation, and therefore may not prove satisfactory. This kind of filter yields an efficient design only if the requirements are close to being symmetrical on a logarithmic frequency scale. The implementation will contain too many inductors, unless the overall degree is a multiple of four. Any other time, use the **Placer** option segment instead.

We needed at least one zero at both zero and infinite frequencies for the circuit to be ladder realizable. This yields a circuit that is not very desirable either in the number of components, or in its structure. The design procedure goes through two more data entry screens, one offers us the ladder form of implementation and computer-selected (Auto) branch sequence, which we accept, while the next offers to print some intermediate results that we don't need:



The performance of this circuit is impeccable though, meeting requirements with plenty to spare.



Note that we can do very little with this circuit, if we wish to change element values, we can only do it at the expense of even more elements, of which we already have too many. Instead, let us consider the same set of requirements, but specifying a parametric design and using the **Placer** option to optimize the locations of the transmission zeros. Note that the **Equal min.** option cannot be used if we select the **Parametric** option.

This time, the **Detail Parameters** window has to be used to enter the rest of the stopband specification:

Requirements		
	Frequency (Hz)	Loss (dB)
1	9.000000E+03	50.00000000
2	22.000000E+03	50.00000000
3	0.000000	0.00000000
4	0.000000	0.00000000
5	0.000000	0.00000000
6	0.000000	0.00000000
7	0.000000	0.00000000
8	0.000000	0.00000000
9	0.000000	0.00000000
10	0.000000	0.00000000
11	0.000000	0.00000000

Fixed zeros		
	Frequency (Hz)	Multiplicity
1	0.000000	0
2	0.000000	0
3	0.000000	0
4	0.000000	0
5	0.000000	0
6	0.000000	0
7	0.000000	0
8	0.000000	0
9	0.000000	0
10	0.000000	0

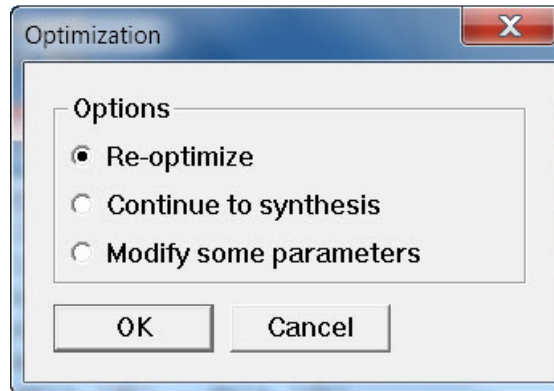
Movable zeros		
	Frequency (Hz)	Multiplicity
1	0.000000	0
2	0.000000	0
3	0.000000	0
4	0.000000	0
5	0.000000	0
6	0.000000	0
7	0.000000	0
8	0.000000	0
9	0.000000	0
10	0.000000	0
11	0.000000	0

or no. of zeros below passband: 3

and/or no. of zeros above passband: 5

The above data specifies that we need 50 dB attenuation from 9 KHz down to zero and from 22 KHz up to infinity. We specified the number of movable transmission zeros in the two stopbands, but this is not necessary. If we leave them at zero, the program will estimate the numbers necessary to meet the specifications. If the requirements are not met, or met with a lot of extra loss, we can modify the data and rerun the approximation easily enough.

The **Placer** procedure is a numerical optimization process . This can be restarted or the input modified if that seems to be necessary:

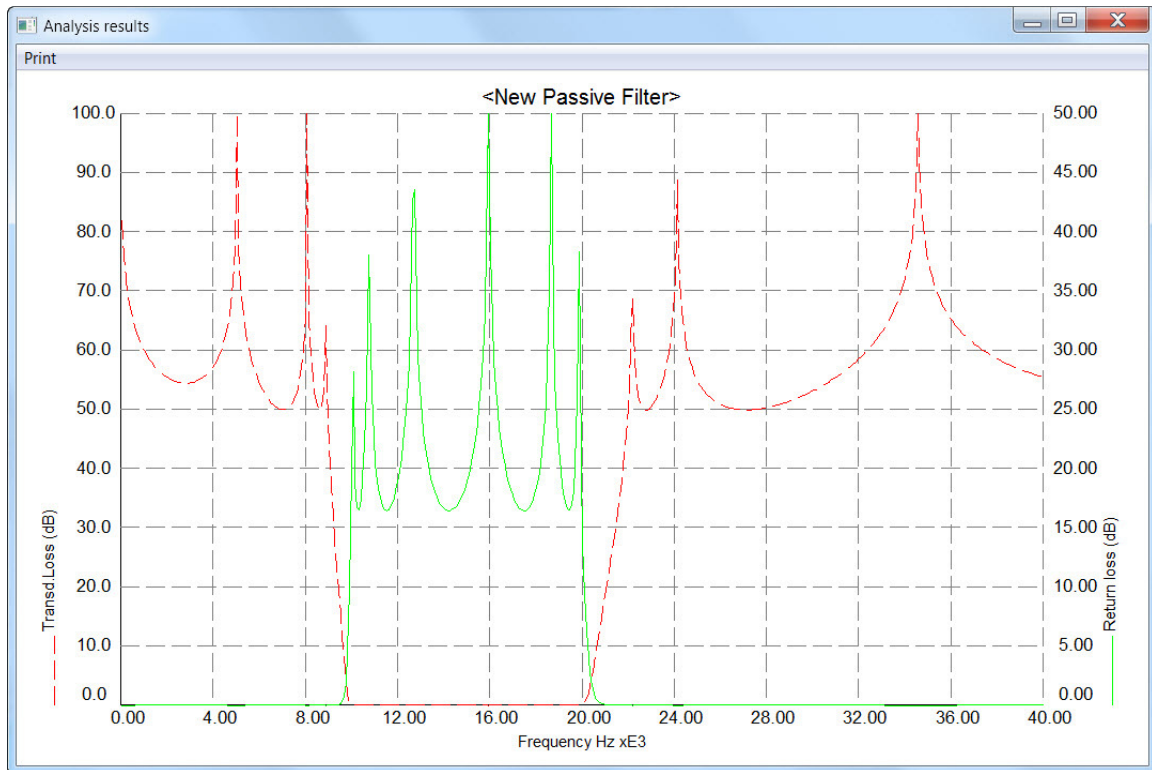


The tabulated results indicate that the 50 dB requirement is missed by about 0.2 dB which is usually considered acceptable. Selecting the **Continue...** step and all the default options, we get the following structure:

Filsyn - PASSIVE LC/MICROWAVE MENU					
File	View	Analyze	Branch	Modify	Transform Macro Undo Help
1		R		50.000000 ohm	
2			C	425.683191 nF	
3		L-C		498.085488 uH	res. frequency
				1.992687 uF	5.051829 KHz
5		C		193.189662 nF	
6			L C	923.576279 uH	res.frequency
				22.952102 nF	34.567862 KHz
8			C	114.542455 nF	
9		L-C		716.139977 uH	res. frequency
				539.661458 nF	8.095813 KHz
11		C		283.398213 nF	
12			L C	529.335261 uH	res.frequency
				82.101416 nF	24.142342 KHz
14			C	161.067352 nF	
15		L-C		1.580170 mH	res. frequency
				201.488581 nF	8.919552 KHz
17		C		200.252081 nF	
18			L C	174.391487 uH	res.frequency
				295.341475 nF	22.176616 KHz
20			C	819.230176 nF	
21		C		20.627593 nF	
23		R		51.901697 ohm	

06-04-2012 5:51 PM Filsyn for Windows V. 1.70

The resulting circuit is substantially more flexible, contains many fewer inductors and has a minimum stopband loss that is only about 8 dB lower and still meets specifications:



This structure is called a zig-zag form, since the branches, which may contain no more than three elements, are alternating series and shunt ones. Also, the circuit contains only 6 inductors as opposed to the 10 in the previous design.

A third way of designing bandpass filters is to start with a lowpass and use the **Shifted bandpass transform** option. This will generate a bandpass, the loss behavior of which is close to being symmetrical on a linear frequency scale (instead of the symmetry on a logarithmic scale, common to bandpass filters).

<New Filter>

Filter kind

- ☒ LC
- ☐ Microwave
- ☐ Active RC
- ☐ IIR Digital

Quarter-wave/Sampling freq (Hz)

Filter type

- ☒ Low pass
- ☐ High pass
- ☐ Band pass

Lower passband freq (Hz)

Upper passband freq (Hz)

Passband type

- ☐ Max. flat
- ☒ Equal ripple
- ☐ Functional
- ☐ Sloping

Band-edge loss/return loss (dB)

Loss Slope (dB/oct)

Flat loss (dB)

Function Type

- ☒ E
- ☐ F

Multiplier

Stopband

- ☐ Monotonic
- ☐ Equal min
- ☒ Placer
- ☐ Specified

Lower stopband freq (Hz)

Loss (dB)

Upper stopband freq (Hz)

Loss (dB)

of zeros at zero

of zeros at infinity

of unit elements

Parametric

- ☒ Conventional
- ☐ Parametric
- ☐ Matching

R1

R2

ZS parameter

Q for predistortion

Predistortion

- ☒ Passband only
- ☐ Pass/stop bands

Shifted bandpass trans.

Center frequency (Hz)

☐ Odd parametric

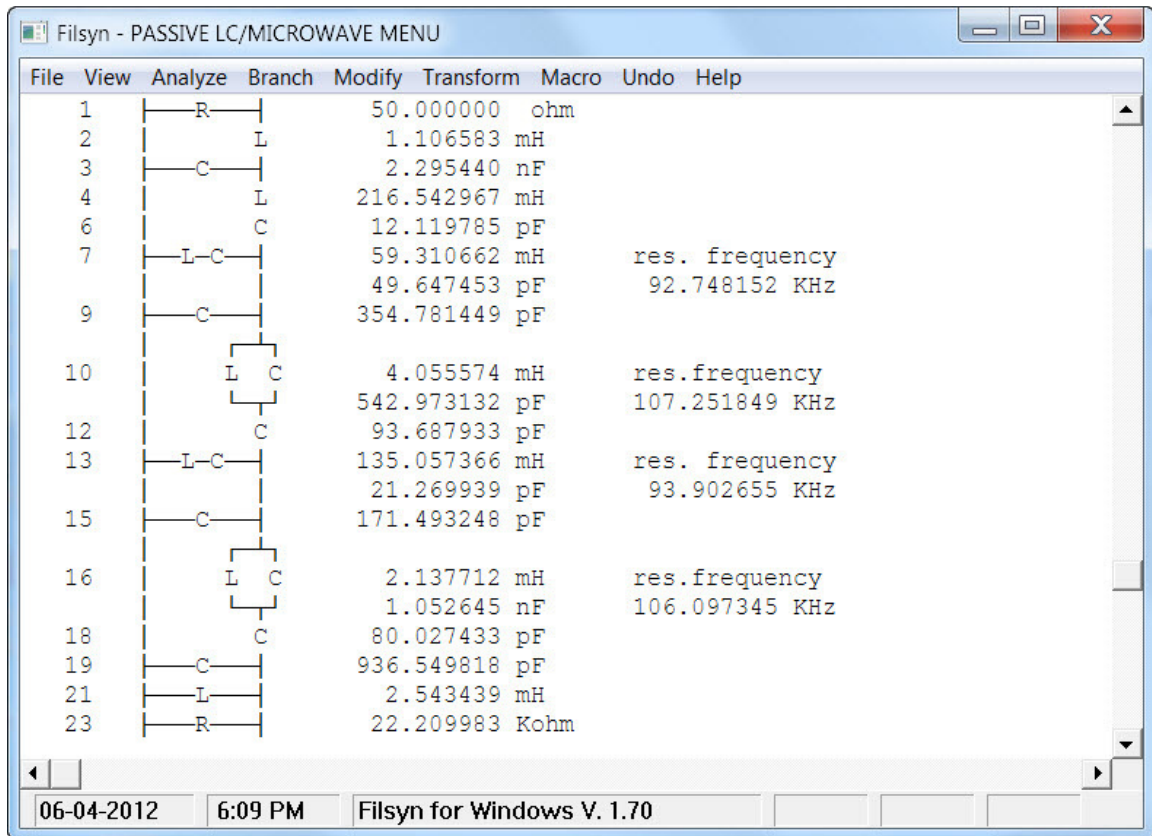
☐ Save design data

OK Cancel

The specification starts as a lowpass, of course, uses the **Placer** option, and could not have been an elliptic design, because we need a 3rd order zero at infinite frequency. The passband is up to 5 KHz, the stopband starts at 6 KHz and requires 50 dB loss and we shift this design to a center frequency of 100 KHz. This will make this a bandpass with passband from 95 KHz to 105 KHz.

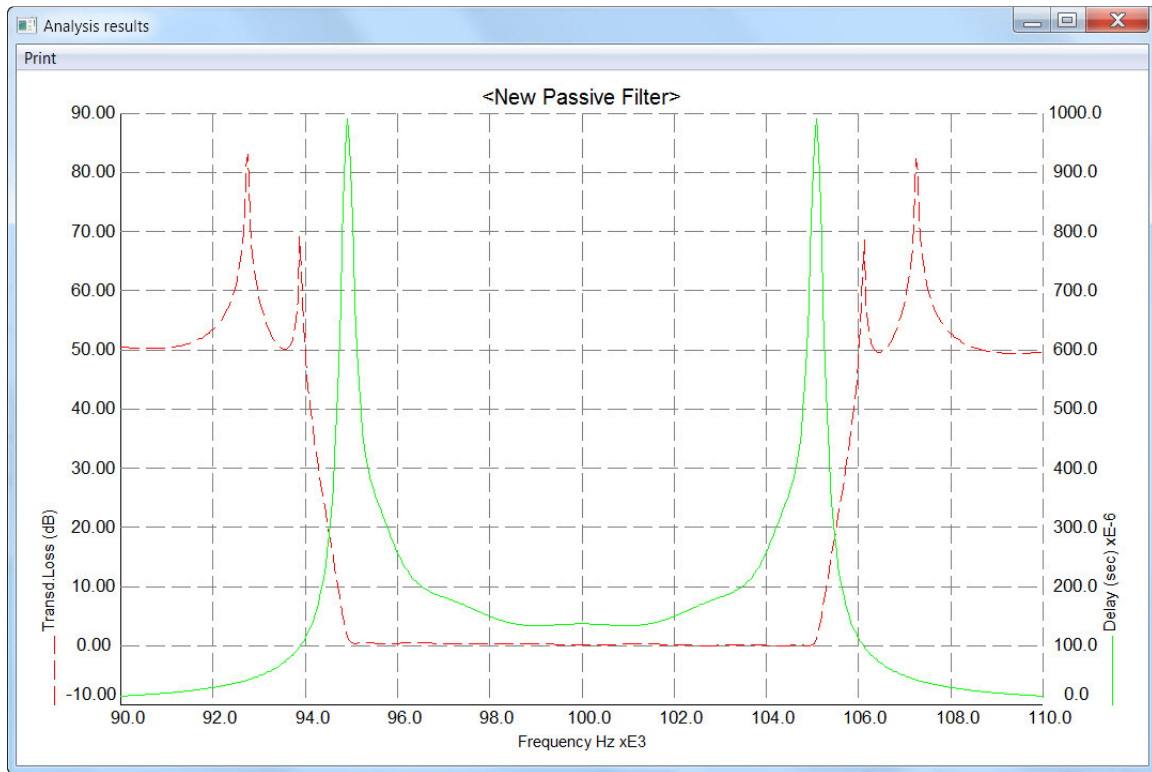
The **Placer** segment selected 2 transmission zeros for the lowpass and shows that the resulting stopband loss will be about 0.5 dB short, that we have accepted.

The resulting circuit is:

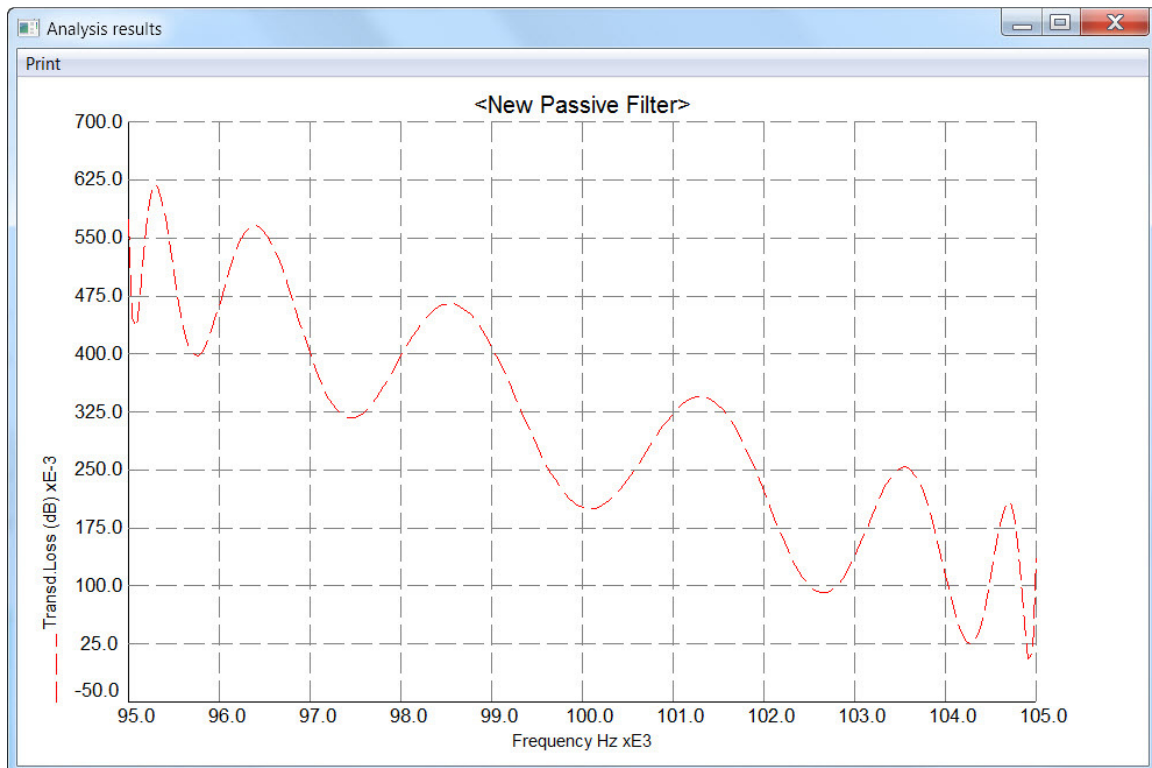


This is again in a convenient form that can be easily modified in many ways. The performance is as closely arithmetically symmetrical as expected. Plotting the loss and the delay this time results:

The screenshot shows the '<New Passive Filter>' dialog box. It has a title bar with a close button (X). The main area contains a section titled 'Select one or two items:' with several checkboxes. The 'Loss' and 'Delay' checkboxes are checked, while 'Phase', 'Return loss', 'Real part', and 'Imaginary part' are unchecked. Below this section is a label 'Zoom frequencies (Hz)' followed by two input fields, both containing the value '0.00000'. At the bottom are 'OK' and 'Cancel' buttons.

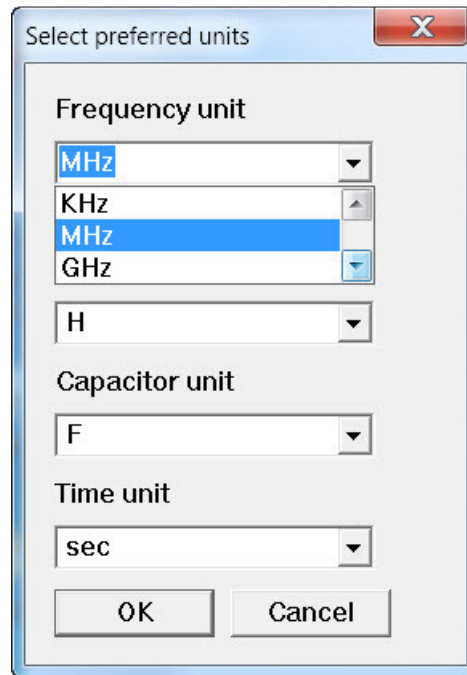


Since the **Shifted bandpass transformation** is an approximate procedure, the passband loss behavior has a small, approximately 0.5 dB slope (please note the vertical scale). The plot below was obtained using the plot selection menu window on the previous page zooming in on the passband details:



3.2 Placer design and lattice conversion

First, we invoke the global unit selector to reset the frequency unit to MHz, since this is the area we are now interested in:



Consider a simple, narrow band filter with passband from 1 MHz to 1.03 MHz and 0.5 dB passband loss ripple. The critical stopband requirements are 40 dB minimum from 1.033 MHz to 1.043 MHz, and 20 dB beyond. No requirements in the lower stopband and we specify 3 transmission zeros at zero (we need shunt inductors at both ends) and one at infinity and finally, we estimate the need for two finite transmission zeros in the upper stopband, although we could leave this setting at zero. The input data screen is as shown:

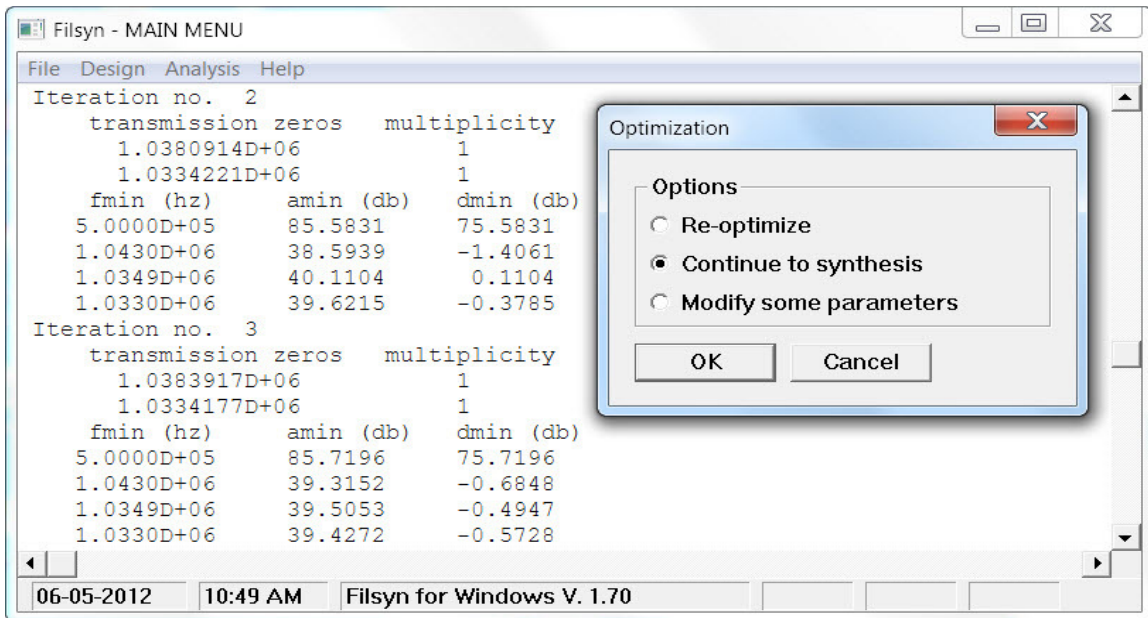
The **Detail Parameters** windows shows the stopband requirements:

Requirements		Fixed zeros		Movable zeros	
	Frequency (MHz)	Loss (dB)		Frequency (MHz)	Multiplicity
1	1.033000E+00	40.00000000	1	0.00000	0
2	1.043000E+00	20.00000000	2	0.00000	0
3	500.000000E-03	10.00000000	3	0.00000	0
4	0.00000	0.00000000	4	0.00000	0
5	0.00000	0.00000000	5	0.00000	0
6	0.00000	0.00000000	6	0.00000	0
7	0.00000	0.00000000	7	0.00000	0
8	0.00000	0.00000000	8	0.00000	0
9	0.00000	0.00000000	9	0.00000	0
10	0.00000	0.00000000	10	0.00000	0
11	0.00000	0.00000000			

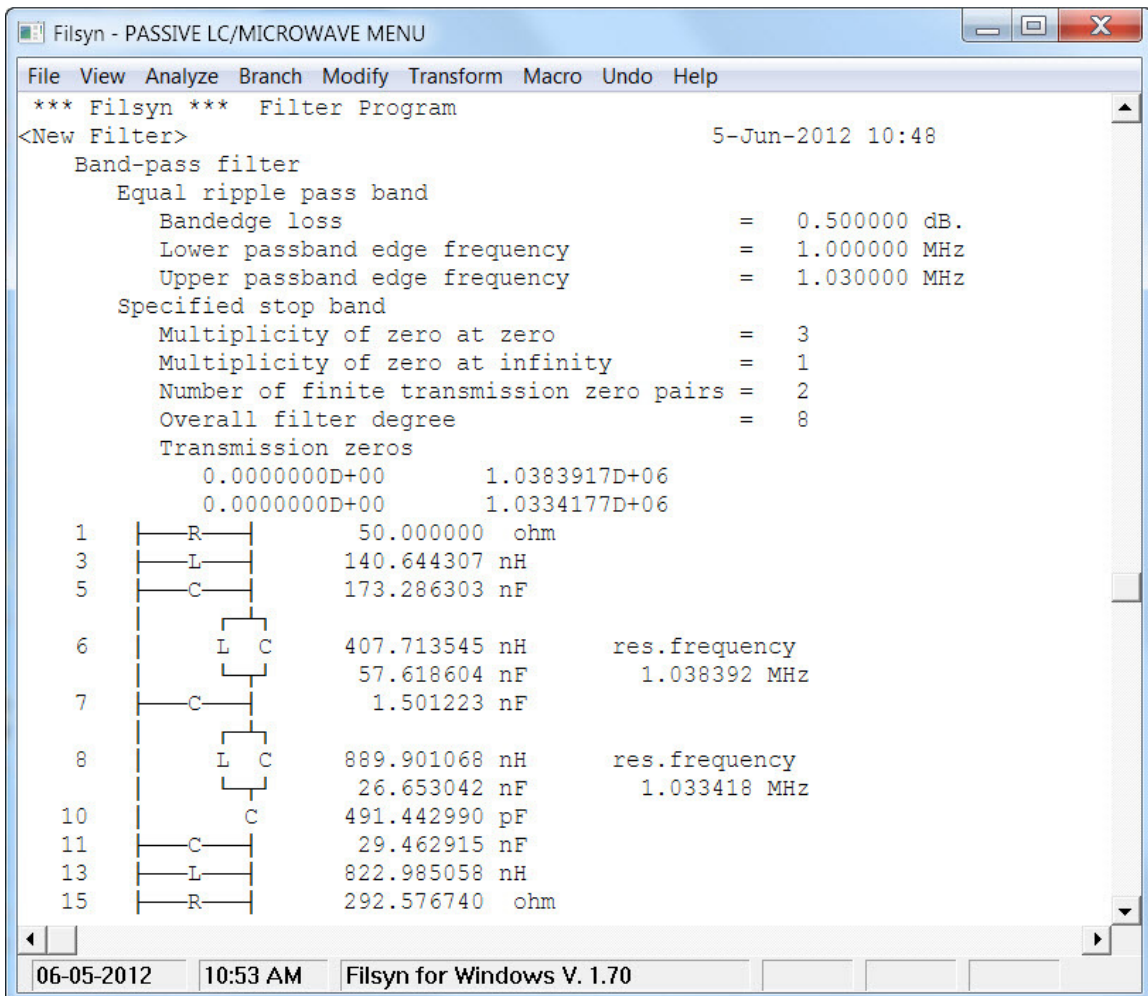
or no. of zeros below passband

and/or no. of zeros above passband

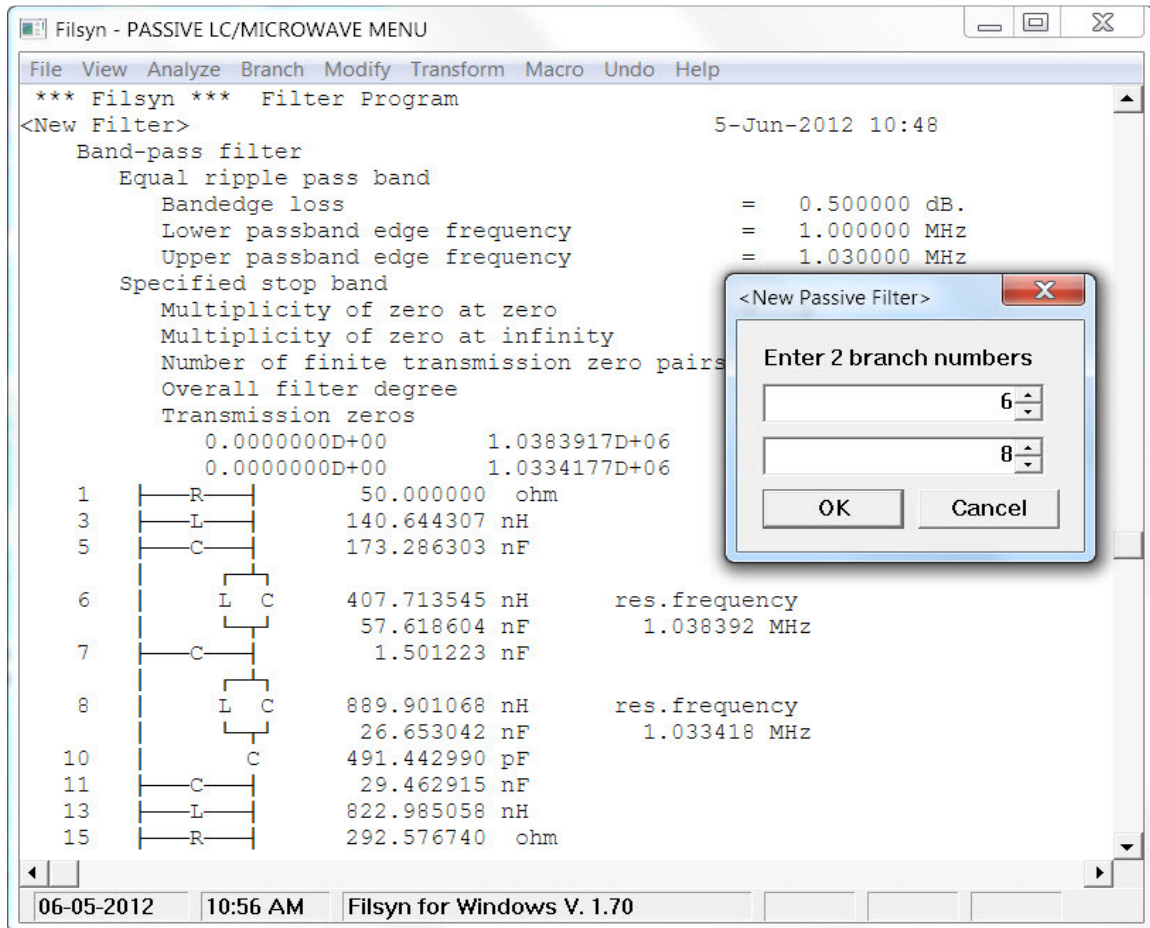
Note that we need at least one breakpoint in the lower stopband. Since that is immaterial, we select a 10 dB requirement at 0.5 MHz. Clicking on **OK** everywhere, the **Placer** routine starts and stops after only 3 iterations:



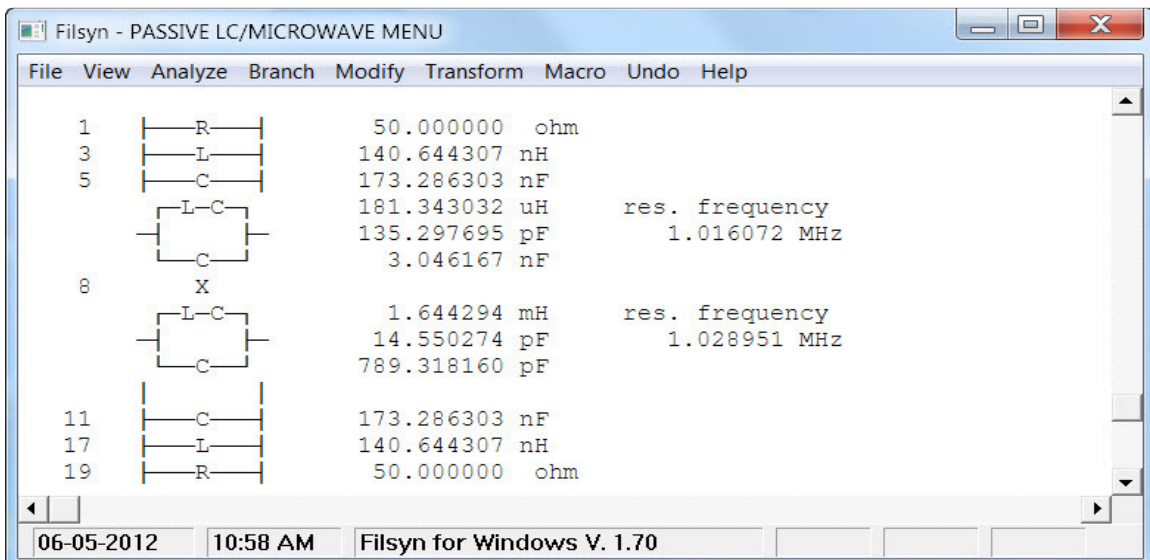
We need no more iteration, we see that the requirements are missed by about half a dB, which we consider acceptable and proceed to the synthesis. The computer selected (default) configuration is shown next:



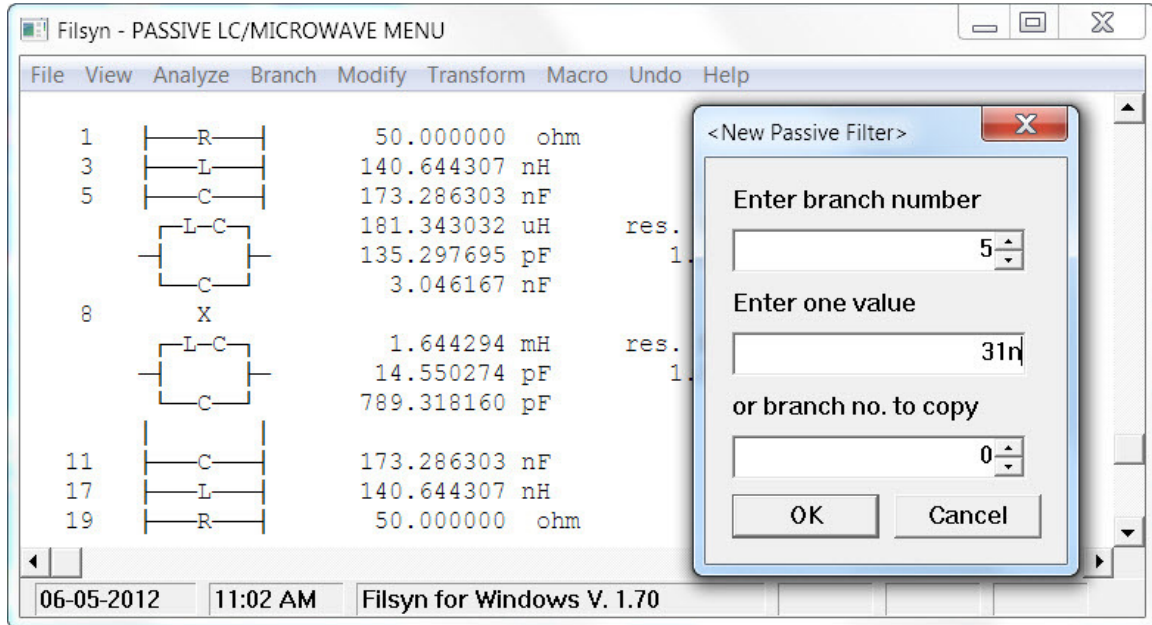
We shall use this occasion, to demonstrate the possibility of converting the inside of this structure to a lattice. This is done by the **Modify->Lattice** menu option, which brings up the entry screen:



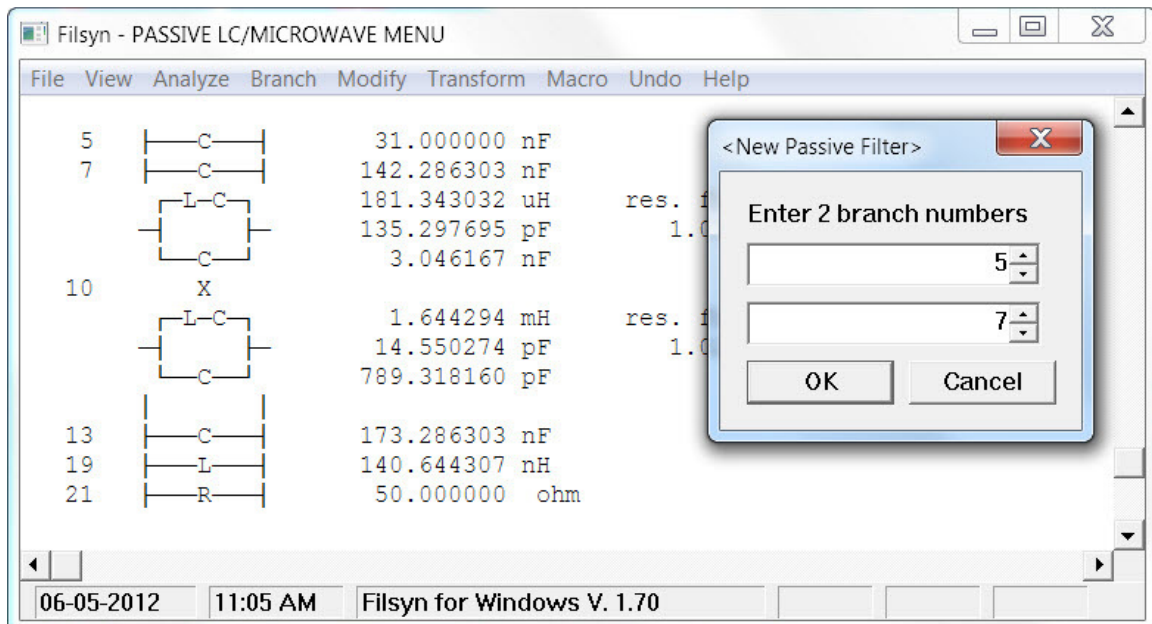
We must indicate the two resonant branches we wish to convert to a lattice. The results are (note that the conversion made the circuit exactly structurally symmetrical):



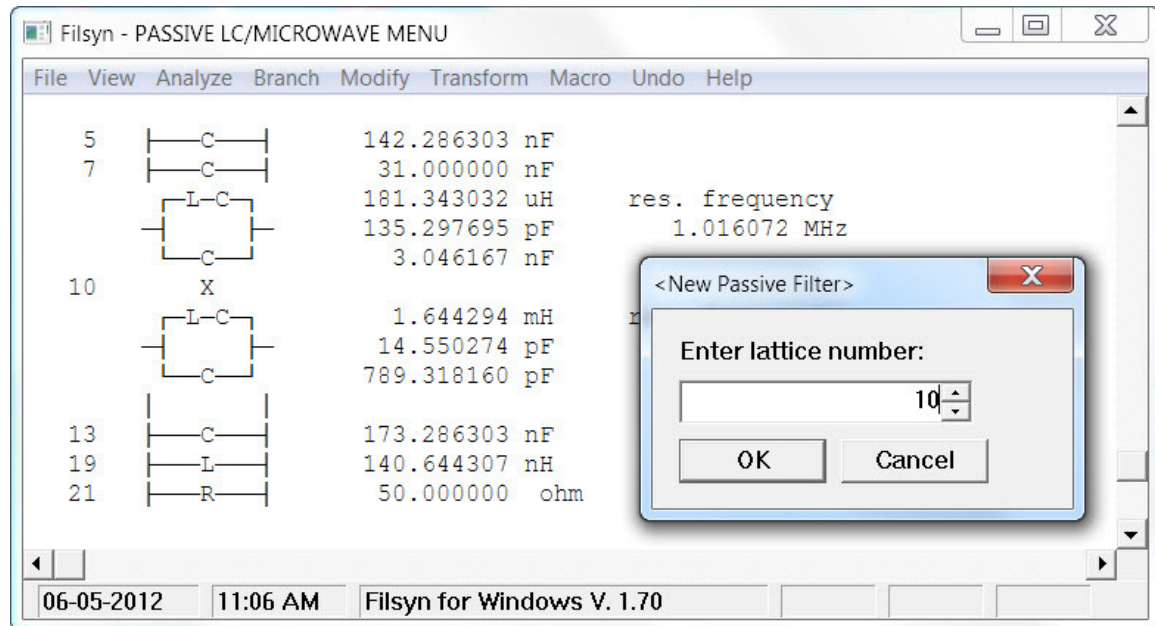
It would be nice to be able to implement these lattice branches as quartz resonators, but their capacitance ratios (the ratio of the shunt C to the series C) are way too small. However there are shunt capacitors on both sides of the lattice, which or some parts of which can be shifted inside. Let us assume, that we need a capacitance ratio of at least 250, i.e. lattice branch 1 (the one with the smaller ratio) needs at least a 34 nF parallel C. That means that we need to shift at least 31 nF capacitance inside the lattice. This can be achieved by first splitting either C_5 or C_{11} into two such that one of the pieces will be 31 nF. Using the **Branch->Split** menu option and declining the offer to split the element into two halves, we have the data screen:



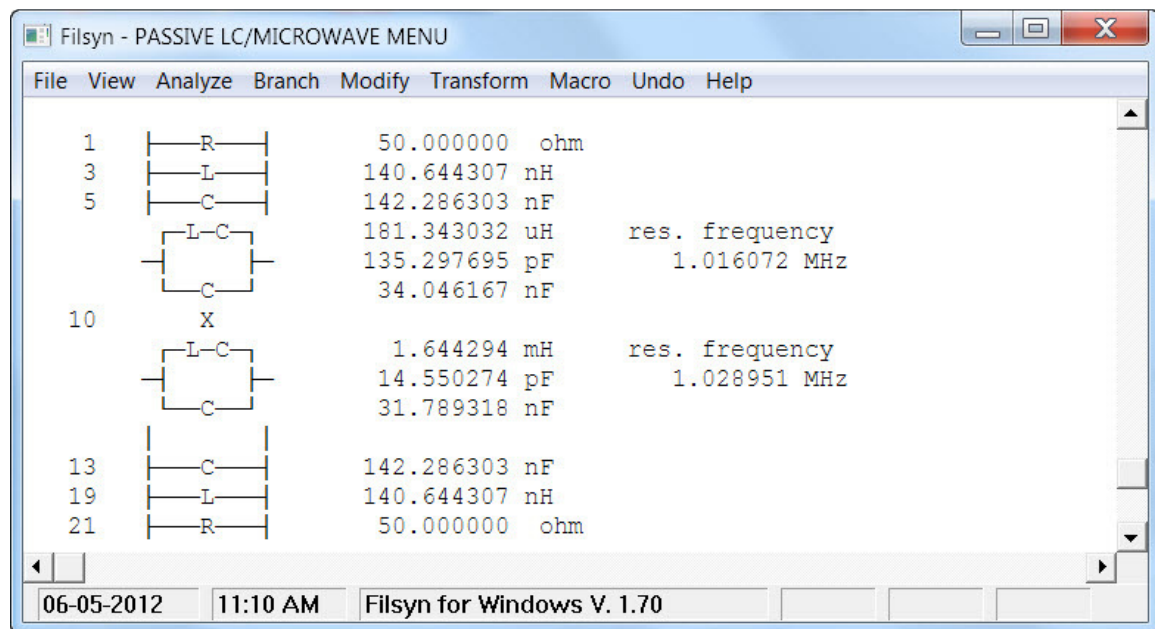
and the result:



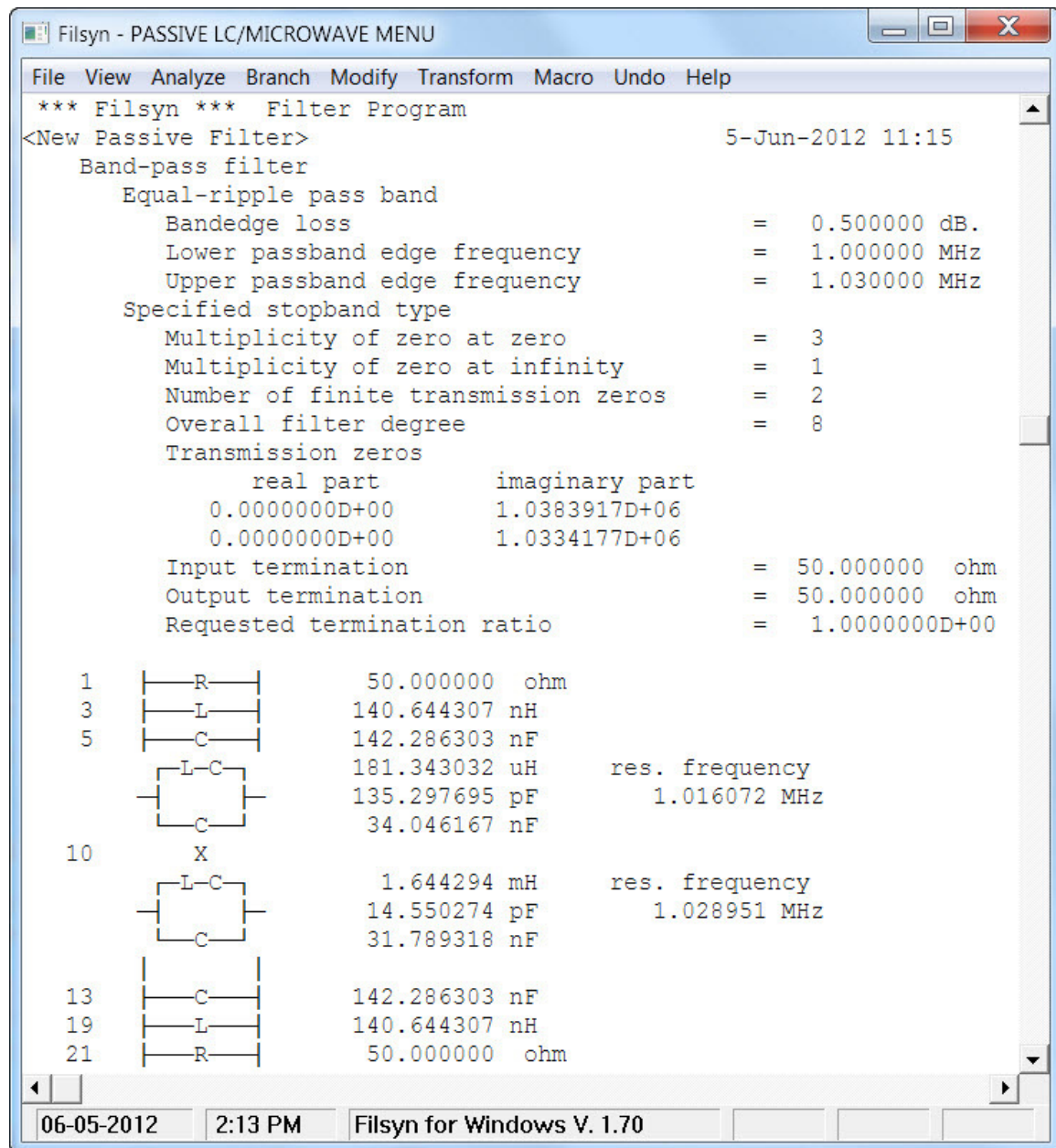
It turns out that the 31 nF capacitor is not next to the lattice, so we must interchange C_5 and C_7 , using **Branch->Interchange** menu (see above). After that step we use the **Modify ->Include** menu that takes the smaller of the capacitive branches on both sides of the lattice and an equal part of the other and shifts those branches inside. These branches can either both be shunt or series ones. We only need to specify the lattice branch number (there might be more than one lattice here):



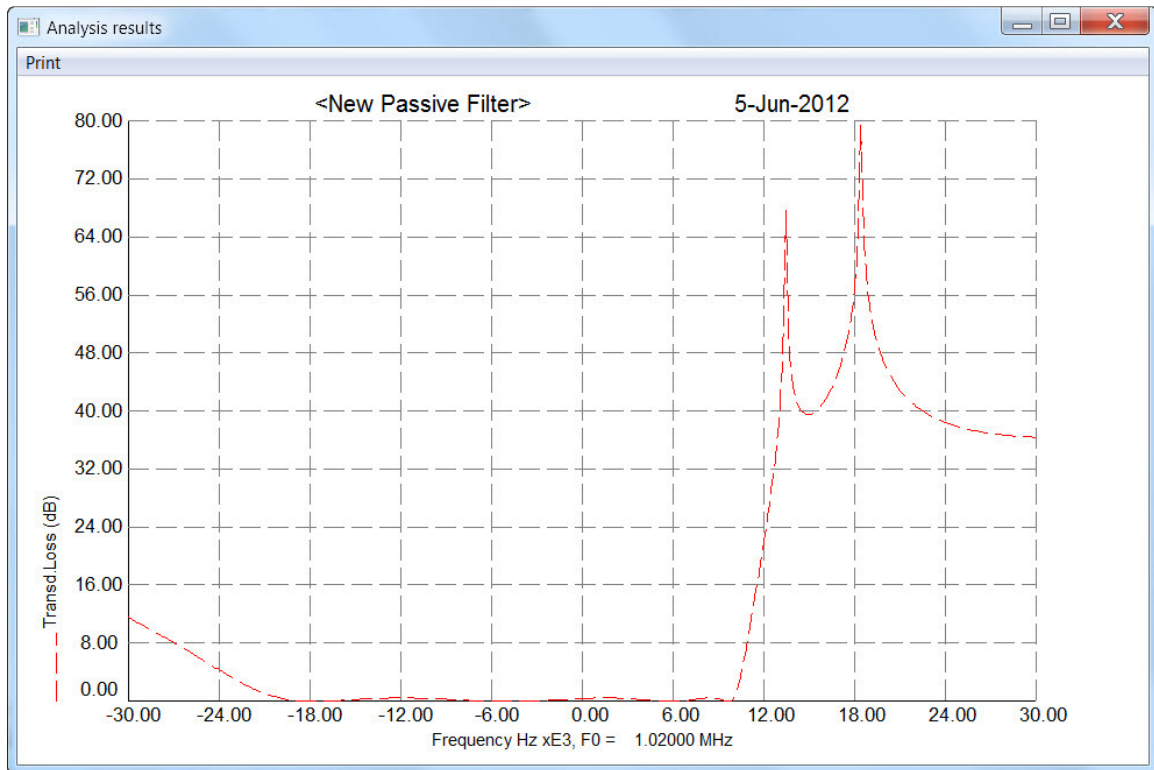
The resulting circuit is now quite acceptable and we could compute the resonant frequency of the L_3 - C_5 pair (the L_{19} - C_{13} will resonate at the same frequency) using the **Analyze->Resonance** menu option. The result is 1.125 MHz.



The final data is summarized here using the **View->Display** and **->Show** menu items:



Finally the frequency domain analysis confirms the design. Note here the frequency scaling, due to the fact that we are displaying a very narrow band.



Just as an aside, we could also get a single lattice implementation of this filter directly and after having done that, we selected one of the pair of implementations yielding:

Realization of lattice branch no. 1					
normalized		denormalized		dual denorm.	
1.8203550D-02	—l—	140.640043 nH	c	56.256017 pF	
5.6330416D+01	—c—	174.082798 nF	l	435.206995 uH	
2.1248816D+02	—l—c—	1.641677 mH	c	36.433798 nH	res. frequency
4.7157583D-03		14.573519 pF	c	656.670696 nF	1.028950 MHz
Realization of lattice branch no. 2					
normalized		denormalized		dual denorm.	
1.8203550D-02	—l—	140.640043 nH	c	56.256017 pF	
5.7059290D+01	—c—	176.335302 nF	l	440.838255 uH	
2.3470240D+01	—l—c—	181.330329 uH	c	338.274929 nH	res. frequency
4.3784148D-02		135.309972 pF	c	72.532132 nF	1.016062 MHz

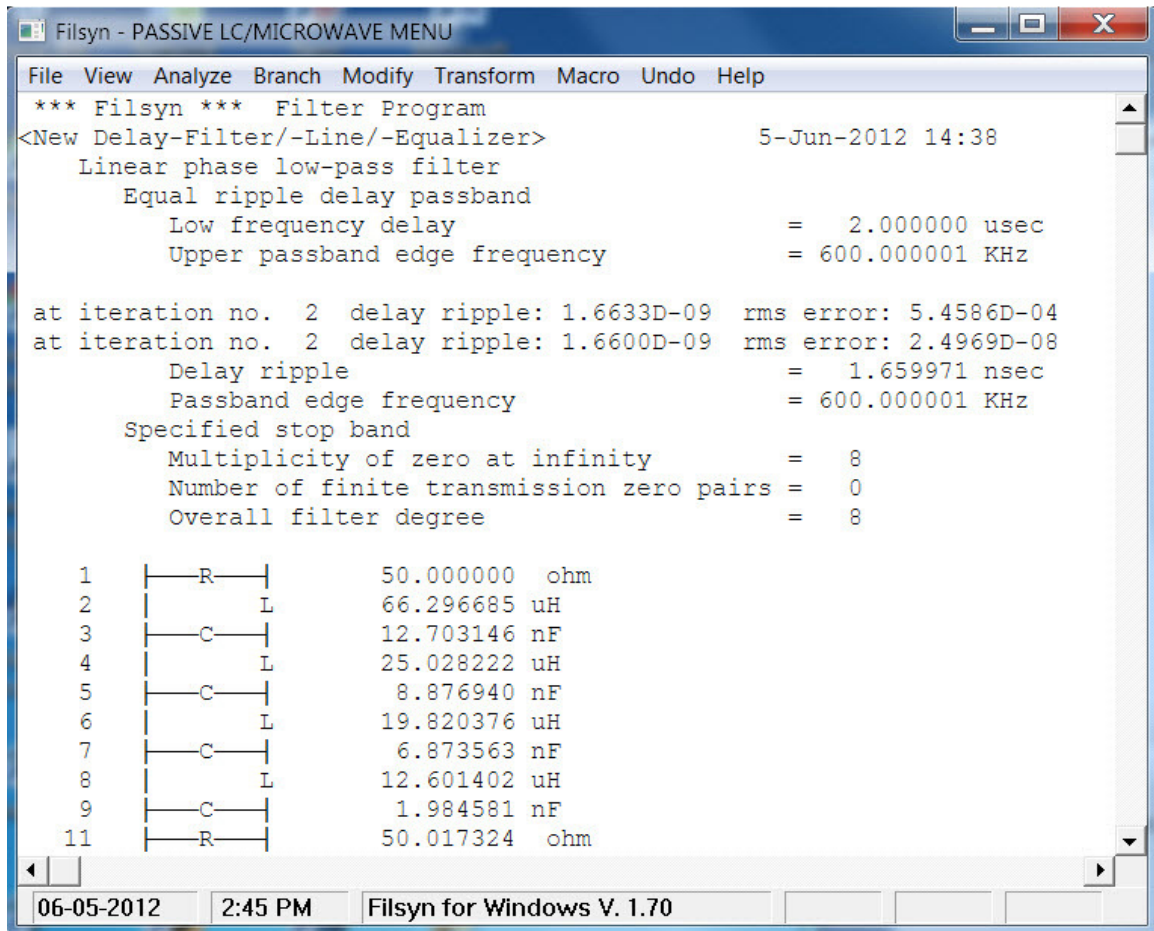
which, in this simple case, is nearly the exactly identical result.

3.3 Linear-phase lowpass

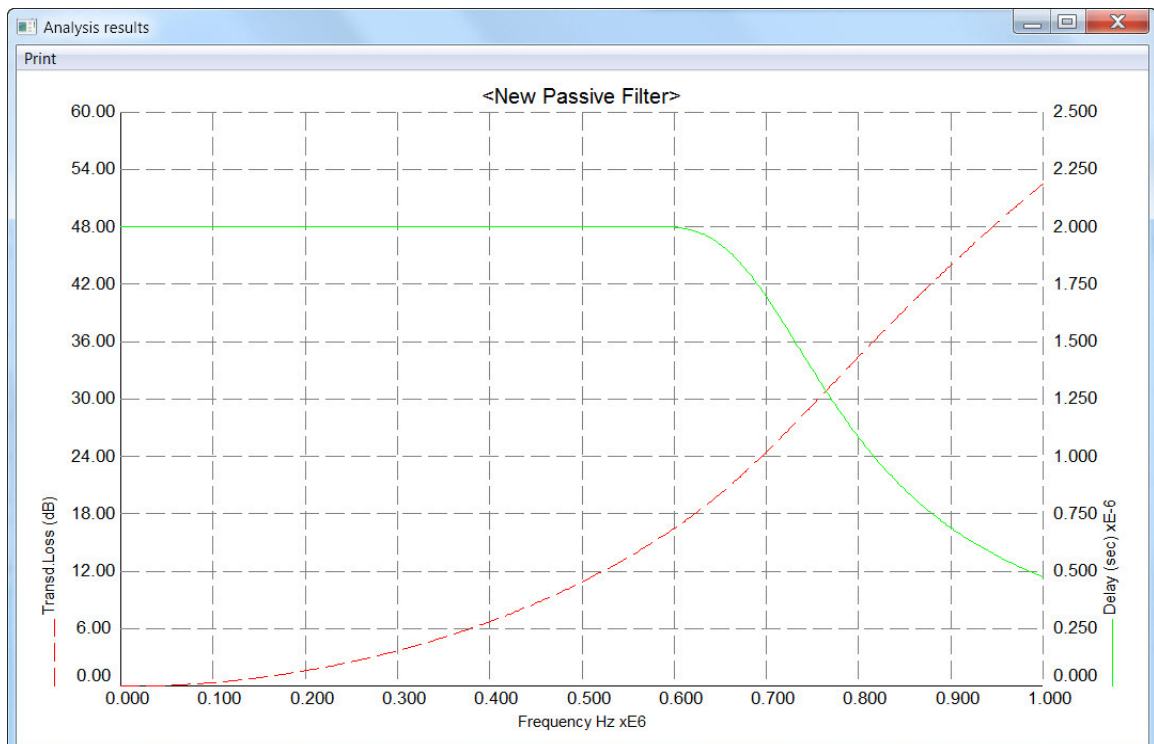
Consider the design of an 8th order linear-phase lowpass, with 2 μ sec delay from 0 to 600 KHz. We request an equal ripple delay as shown:

The procedure to obtain equal ripple delay uses an optimization method that may be repeated as indicated:

but an rms (root mean square) error of 2.5×10^{-8} is already much better than necessary. The final design is shown next after we accept the default computer-selected synthesis:



while the performance shows excellent results:



We can, of course, generate a delay line, instead of a filter, but note that the delay of an 8th order delay line will be twice that of the same degree filter, hence we specify 4 μ sec delay:

<New Delay-Filter/-Line/-Equalizer>

Delay type

- ☐ Max. flat
- ☒ Equal ripple
- ☐ Equalizer

Implementation

- ☒ LC elements
- ☐ Active RC
- ☐ Microwave
- ☐ IIR digital

Delay value (sec)

4.000000E-06

Stopband

- ☐ Equal min.
- ☒ Specified

End of passband (Hz)

600.000000E+03

Start of stopband (Hz)

0.000000

Degree

8

Zeros at infinity

0

Transm. zeros

Terminations

50.000000E+00

50.000000E+00

Q-value

0.00000000

Hurwitz type

- ☒ Hurwitz
- ☐ Non Hurwitz

Selection

- ☒ Auto
- ☐ Manual

Indicators

Shifted bandpass trans.

Center frequency (Hz)

0.000000

☐ Odd parametric

☐ Save design data

Network type

- ☐ Filter
- ☒ Delay line

Data

FQ or FS (Hz)

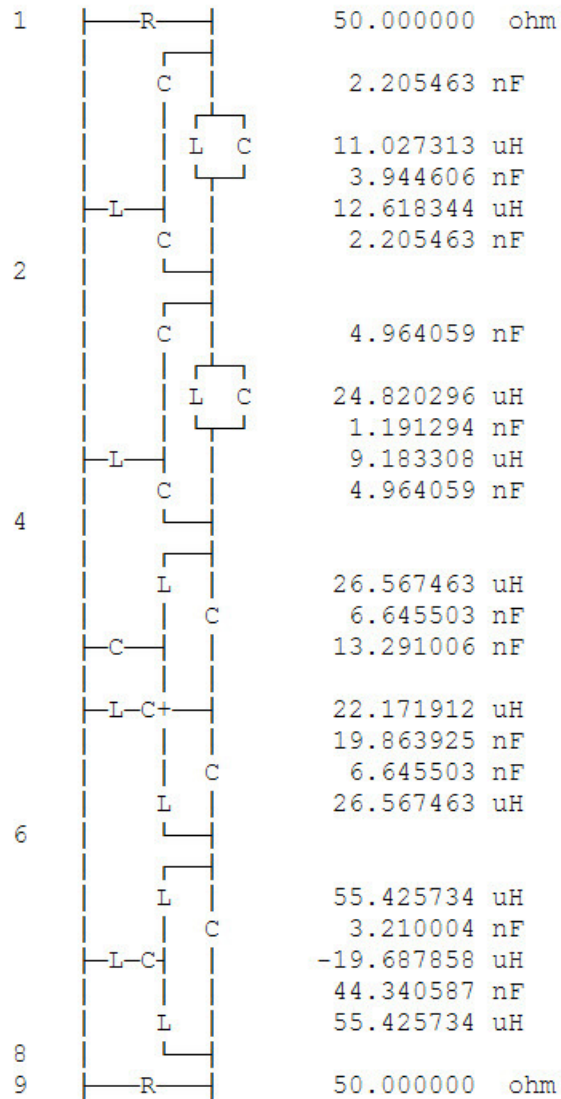
0.000000

Normalization freq (Hz)

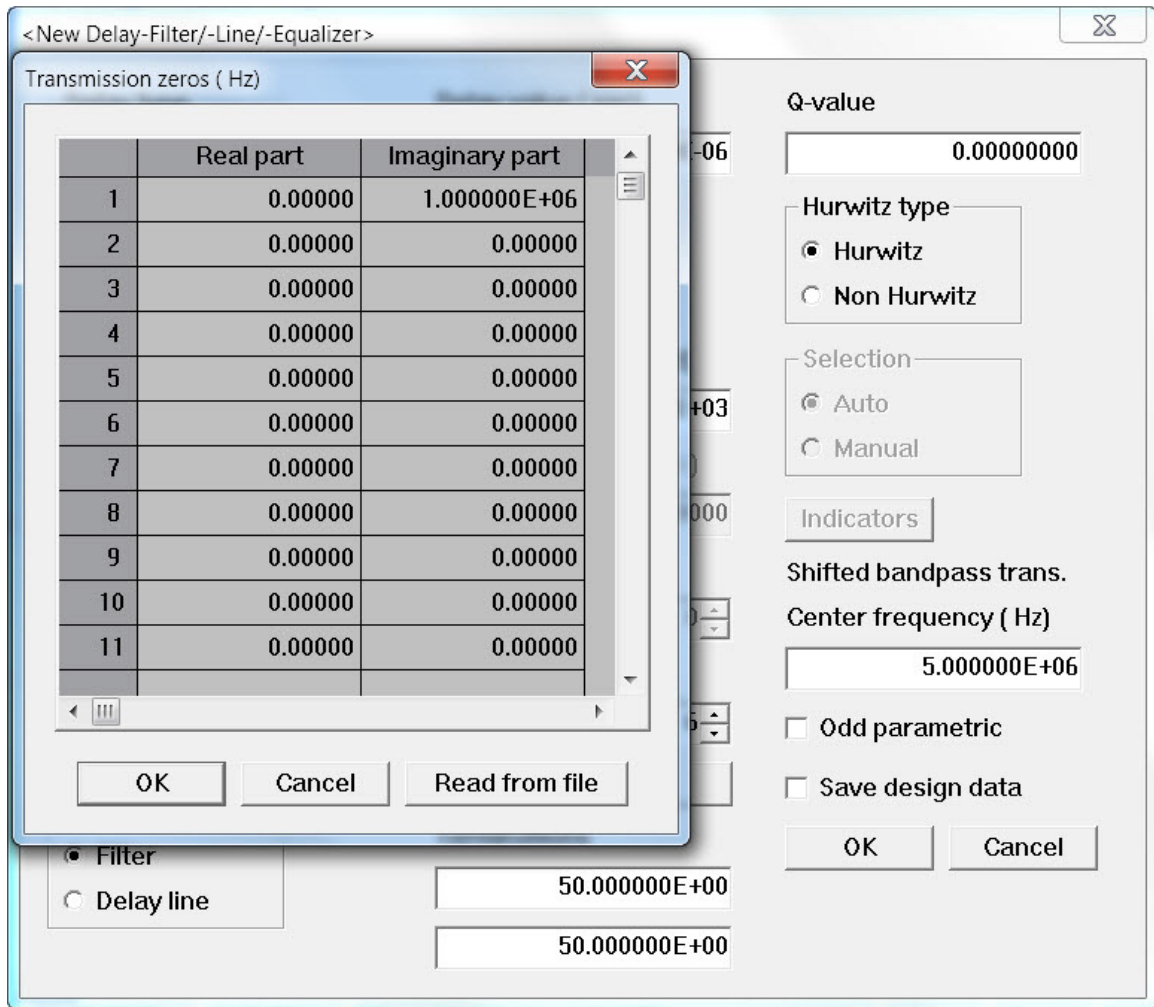
0.000000

OK **Cancel**

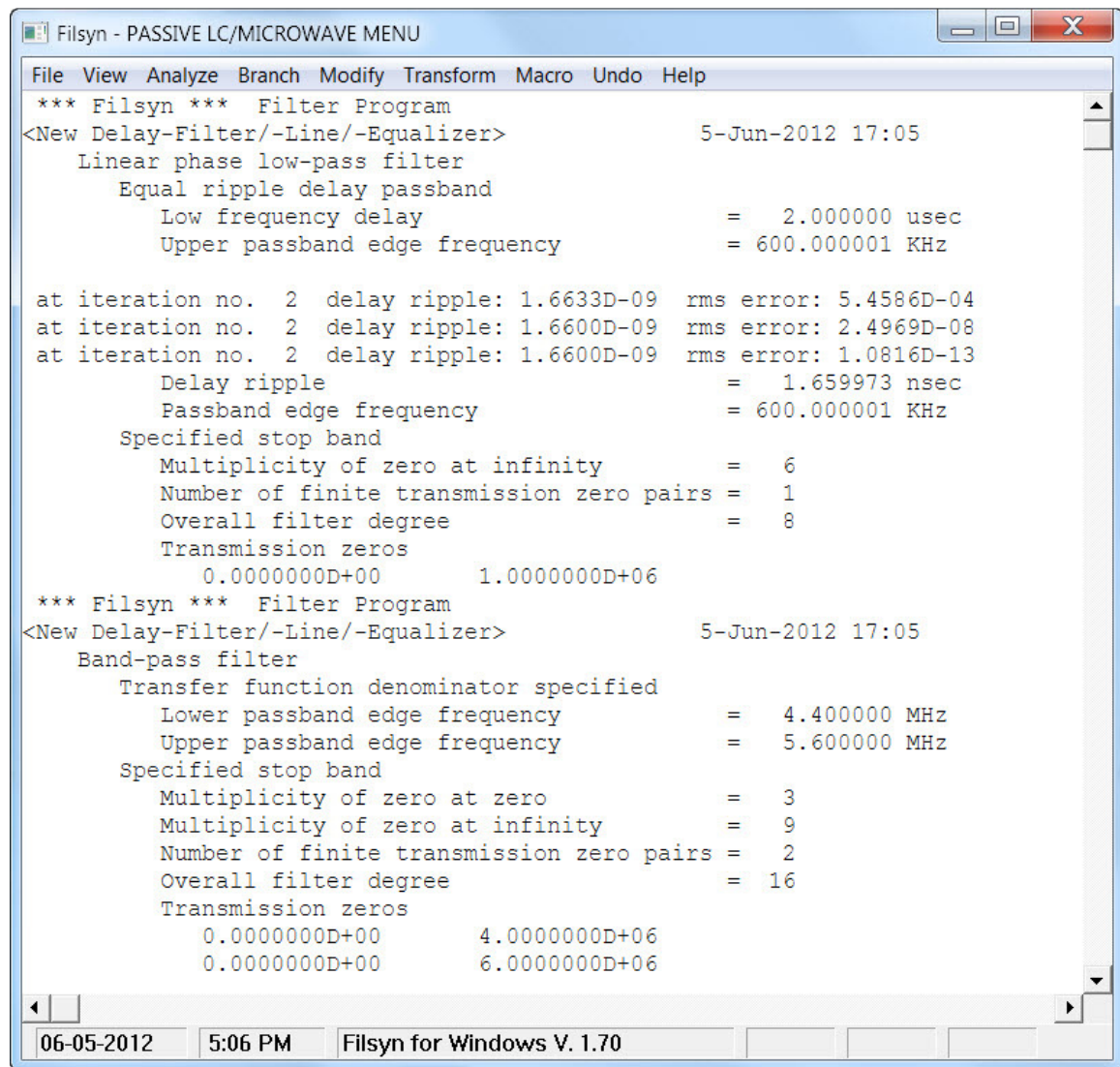
The resulting circuit contains 4 second order delay sections:

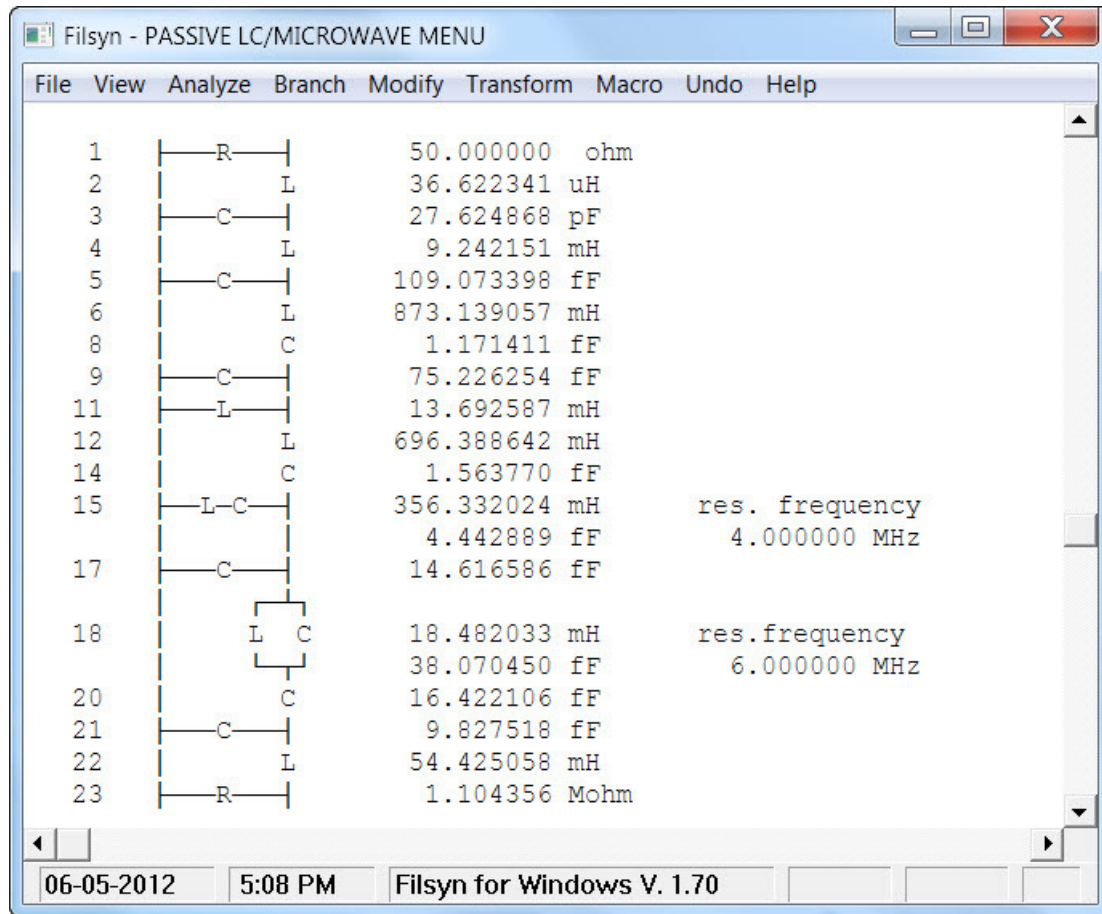


Finally, we may create a linear phase bandpass, by using the **Shifted bandpass trans.** (transformation) option by specifying a 5 MHz center frequency. We will also specify a finite transmission zero to show further flexibility in the design:

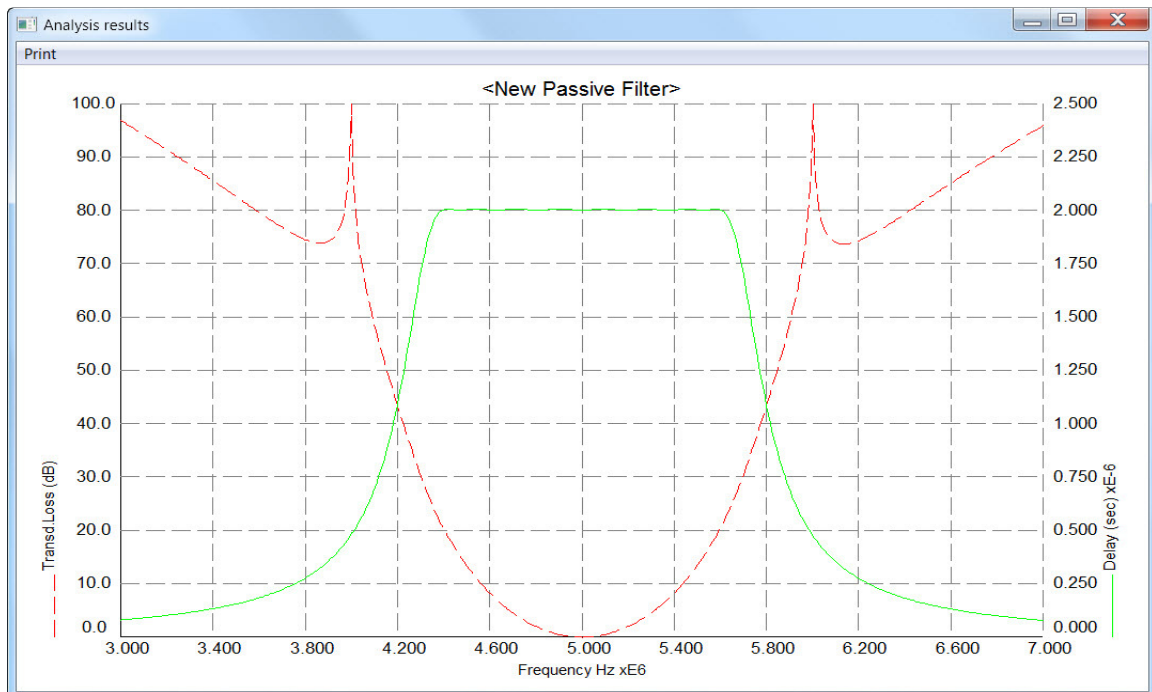


The resulting bandpass is shown next, but note that the element values are not all desirable. They can of course, be modified extensively, which is dealt with in the next chapter. First we show the design summary, then the circuit itself:

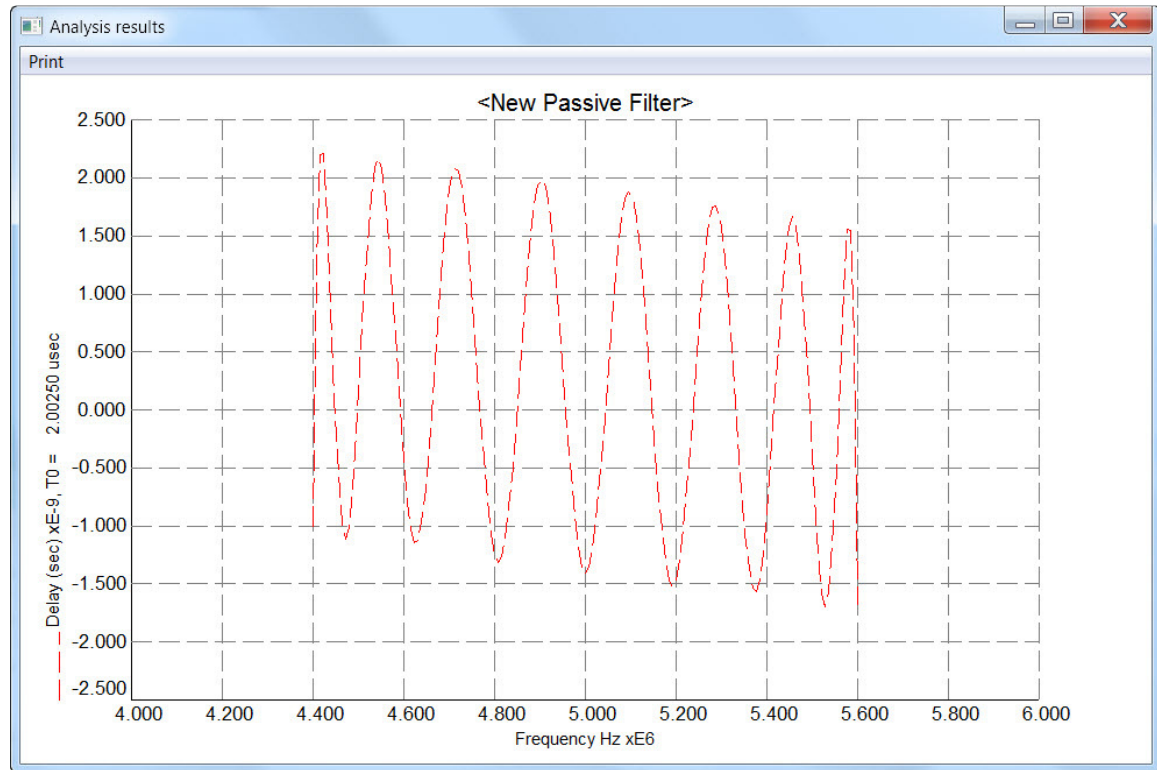




The performance of this bandpass, see below, shows a symmetrical behavior. The program moved 3 zeros to zero frequency and 9 zeros to infinite frequency with this object in view; the optimal is to have three times as many zeros at infinity as at zero.



Looking at the details of the delay in the passband, we see that the peak-to-peak ripple is indeed about 3.4 μsec and the slight slope, due to the shifted transformation, is a clearly negligible 0.7 nsec, much smaller than the 3 nsec ripple.



As we can see, we have no freedom in selecting the transmission zero locations or how many zeros there will be at zero and infinite frequencies. If there is a need to change that, the procedure to follow is to write the pole-zero data to a file at this point. That file may then be edited to change the locations of the finite transmission zeros. Next we restart the design using functional input (see below) and read the pole-zero data from the file. At this time we can also set the extreme zeros as long as their sum is unchanged.

3.4 Functional input

This particular option permits us to enter the transfer function or the characteristic function directly into the program. This can be done even in the case of digital (bilinear Z-transform only) or microwave filters, provided that the functions are expressed in terms of the equivalent analog 's' variable. For definition and relationships between the transfer and characteristic functions and their interpretations for digital and microwave filters, *Appendix G* should be consulted. At this point it is sufficient to say that entering the transfer function means entering the two polynomials $E(s)$ and $Q(s)$, and entering the characteristic function means entering the polynomials $F(s)$ and $Q(s)$. The $E(s)$ or $F(s)$ functions are specified by their zeros in either radian/sec units or normalized to the (upper) passband edge, while the $Q(s)$ polynomial is specified by entering the (finite) transmission zeros in Hz. In the case of IIR digital and microwave filters, $E(s)$ and $F(s)$ should be entered in transformed - normalized forms, but the zeros of $Q(s)$ are still entered in Hz.

These quantities may be entered from the keyboard into the proper data entry screen, but alternatively, we may prepare a text file containing the data and request that the program read them from that file. In either case, complex quantities need be entered only once, its complex conjugate is *not* needed.

The file, if we prepare one, must have a *.dat* extension and have the following structure:

1. It may start with any number of comment lines as long as they have an ! (exclamation mark) as their first nonblank character
2. The roots of $E(s)$ [or $F(s)$] must follow, one complex number (real part and imaginary part) on a line, separated by blank(s) and in scientific notation only
3. If there are finite transmission zeros, they must follow after a separating line that also begins with an !
4. There should be no other comment lines anywhere, although blank lines can be used

The signs of the real and/or imaginary parts of these quantities are immaterial with the exception of the real parts of the $F(s)$ polynomial zeros. A different sign pattern will yield an equivalent but non-identical, circuit. Furthermore, for the F polynomial we must enter a multiplier, which is not needed for the E polynomial. Note that if the function is of the bandpass type, the lower passband edge is immaterial but should be selected to be near the actual passband. Such a selection will help to maintain numerical accuracy during computation. Finally, if the F polynomial has zero(s) at zero frequency, they must be specified as a very small, but nonzero (< 0.001) real part and an, of course zero imaginary part.

In the special case of a lowpass $E(s)$ function, an equal minima type stopband becomes an available option, in place of specifying the $Q(s)$ polynomial (i.e. the transmission zeros). Parametric bandpasses are specified either by entering an odd degree E or F , or by entering an even F function that has a pair of real zeros – one positive, the other negative. One might try to request a parametric case when entering the E polynomial with at least two (negative) real roots, but there is no guarantee that it will work. If a parametric design is not possible under the circumstances, an advisory message to this effect will be printed. All other bandpass cases are necessarily conventional, i.e. non-parametric.

Specified $F(s)$ example

We shall now show the procedure for entering the characteristic function, $F(s)$. First we choose a bandpass characteristic function with a numerator having zeros at 800 Hz, 850 Hz, 1150 Hz and 1200 Hz, and a pair of complex zeros at $125 \pm j1000$ Hz. The poles (transmission zeros) should be at 600 Hz, 1400 Hz and there should be a pair at zero frequency. Comparing the degrees of the numerator and denominator, we observe that the transmission zero at infinity will be of order four. Next we select the passband edges to be 750 Hz and 1250 Hz. The upper band edge (1250 Hz) will be our normalization frequency, but apart from that, neither of the band edge frequencies has a direct influence on the final results. The C multiplier is computed to yield a loss of 0.25 dB at the center frequency of 1 KHz, which gives $C = 102.8288$.

Normalizing everything to 1250 Hz we obtain the polynomials:

$$F(s) = C(s \pm j0.64)(s \pm j0.68)(s - 0.1 \pm j0.8)(s \pm j0.92)(s \pm j0.96)$$

and

$$Q(s) = s^2(s \pm j0.48)(s \pm j1.12)$$

The data entry screen is therefore:

<New Filter>

Filter kind

- ☒ LC
- ☐ Microwave
- ☐ Active RC
- ☐ IIR Digital

Quarter-wave/Sampling freq (Hz)

0.00000

Filter type

- ☐ Low pass
- ☐ High pass
- ☒ Band pass

Lower passband freq (Hz)

750.000000E+00

Upper passband freq (Hz)

1.250000E+03

Passband type

- ☐ Max. flat
- ☐ Equal ripple
- ☒ Functional
- ☐ Sloping

Band-edge loss/return loss (dB)

0.500000000

Loss Slope (dB/oct)

6.00000000

Flat loss (dB)

0.00000000

Function Type

- ☐ E
- ☒ F

Multiplier

102.8288

Stopband

- ☐ Monotonic
- ☐ Equal min
- ☐ Placer
- ☒ Specified

Lower stopband freq (Hz)

0.00000

Loss (dB)

50.0000000

Upper stopband freq (Hz)

0.00000

Loss (dB)

50.0000000

of zeros at zero

2

of zeros at infinity

4

of unit elements

0

Parametric

- ☒ Conventional
- ☐ Parametric
- ☐ Matching

R1

50.000000E+00

R2

50.000000E+00

ZS parameter

-1

Q for predistortion

0.00000000

Predistortion

- ☒ Passband only
- ☐ Pass/stop bands

Shifted bandpass trans. Center frequency (Hz)

0.00000

☐ Odd parametric

☐ Save design data

OK Cancel

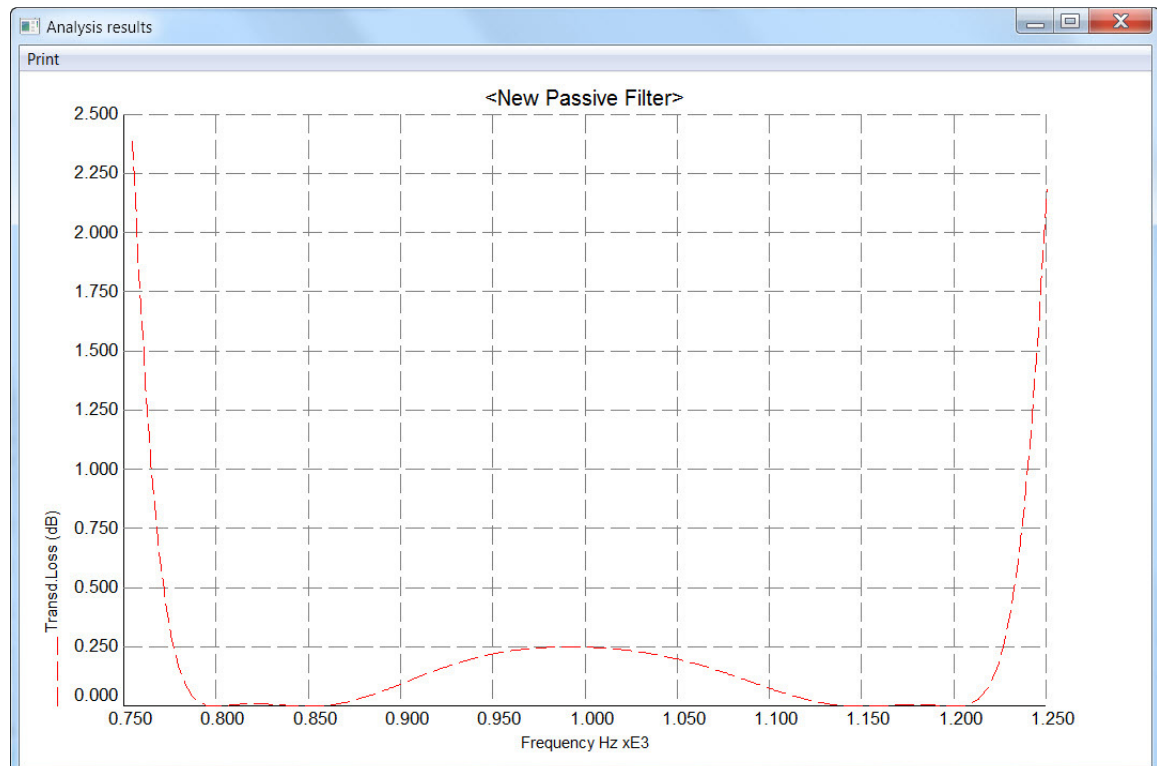
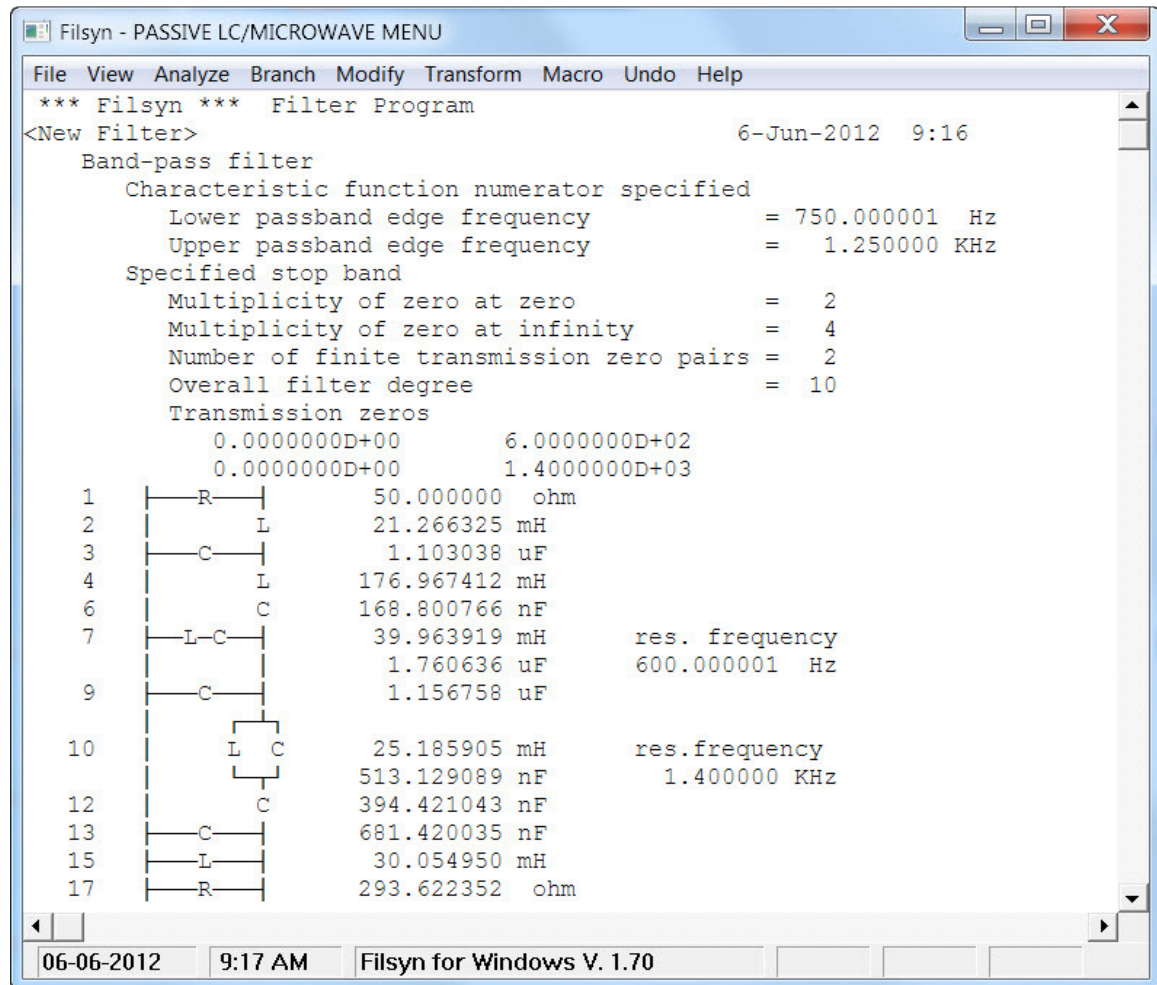
where the **Detail parameters** window shows the function zeros and transmission zeros we enter:

Filter Parameters

Function zeros (in rad/sec)			Transmission zeros (in Hz)		
	Real Part	Imaginary Part		Real Part	Imaginary Part
1	0.00000	5.026548E+03	1	0.00000	600.000000E+00
2	0.00000	5.340708E+03	2	0.00000	1400
3	785.398000E+00	6.283185E+03	3	0.00000	0.00000
4	0.00000	7.225663E+03	4	0.00000	0.00000
5	0.00000	7.539822E+03	5	0.00000	0.00000
6	0.00000	0.00000	6	0.00000	0.00000
7	0.00000	0.00000	7	0.00000	0.00000
8	0.00000	0.00000	8	0.00000	0.00000
9	0.00000	0.00000	9	0.00000	0.00000
10	0.00000	0.00000	10	0.00000	0.00000
11	0.00000	0.00000	11	0.00000	0.00000

OK Cancel Read from file

The resulting network of the default configuration again is shown next, while the passband details of the loss performance shows perfect agreement with our specifications:



Specified E(s) example

This example involves the specification of the transfer function, i.e. the polynomial $E(s)$. Ulbrich and Piloty (see ref. [57]) tabulate the natural modes (zeros of E) of a 10th degree bandpass with an equal ripple delay from 40 kHz to 62.5 kHz (compare this with the example about linear phase filters above):

- $0.114851468 \pm j 1.025224774$
 - $0.109598959 \pm j 1.146020455$
 - $0.109753066 \pm j 0.904326415$
 - $0.084987609 \pm j 1.259372919$
 - $0.085231480 \pm j 0.790671590$

These are normalized to 50 KHz, so the denormalized values are the ones we saved in a simple text file named *Chap3_1.dat* where we added a couple of transmission zeros:

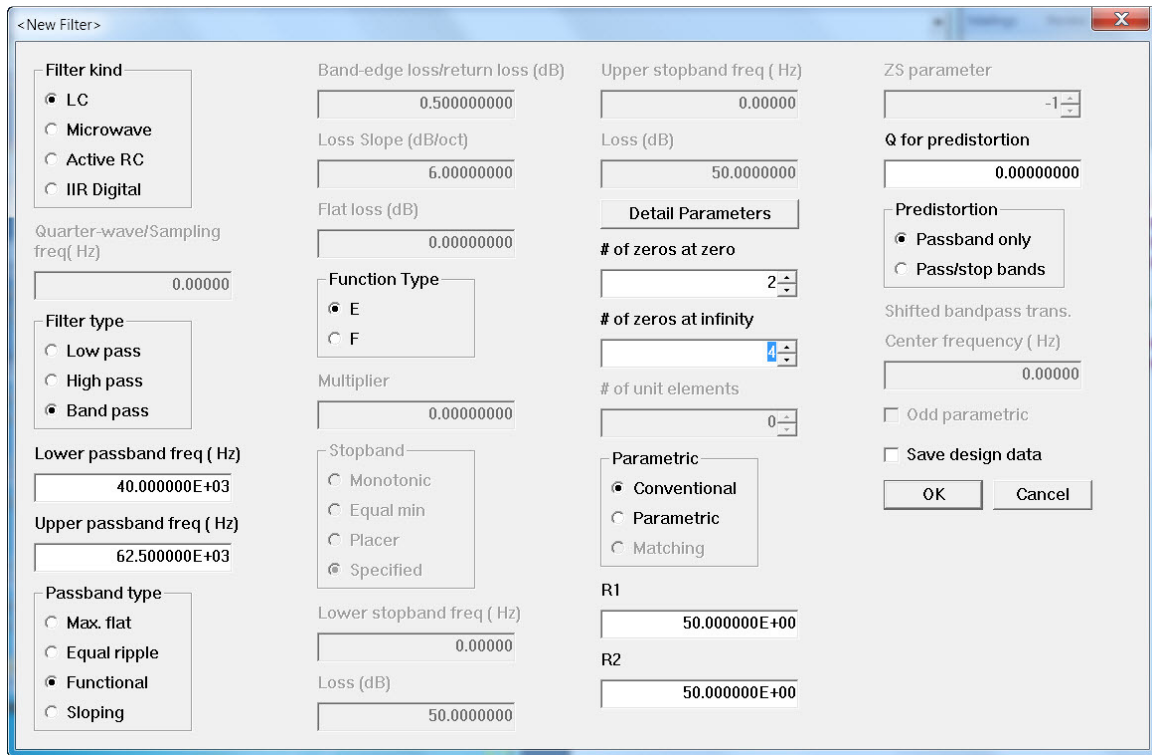
```
! Ulbrich & Piloty
! Transfer function
! Poles (natural modes)
! real part      imag. part

3.608165e4      3.220839e5
3.443154e4      3.600329e5
3.447994e4      2.841025e5
2.669964e4      3.956437e5
2.677626e4      2.483992e5

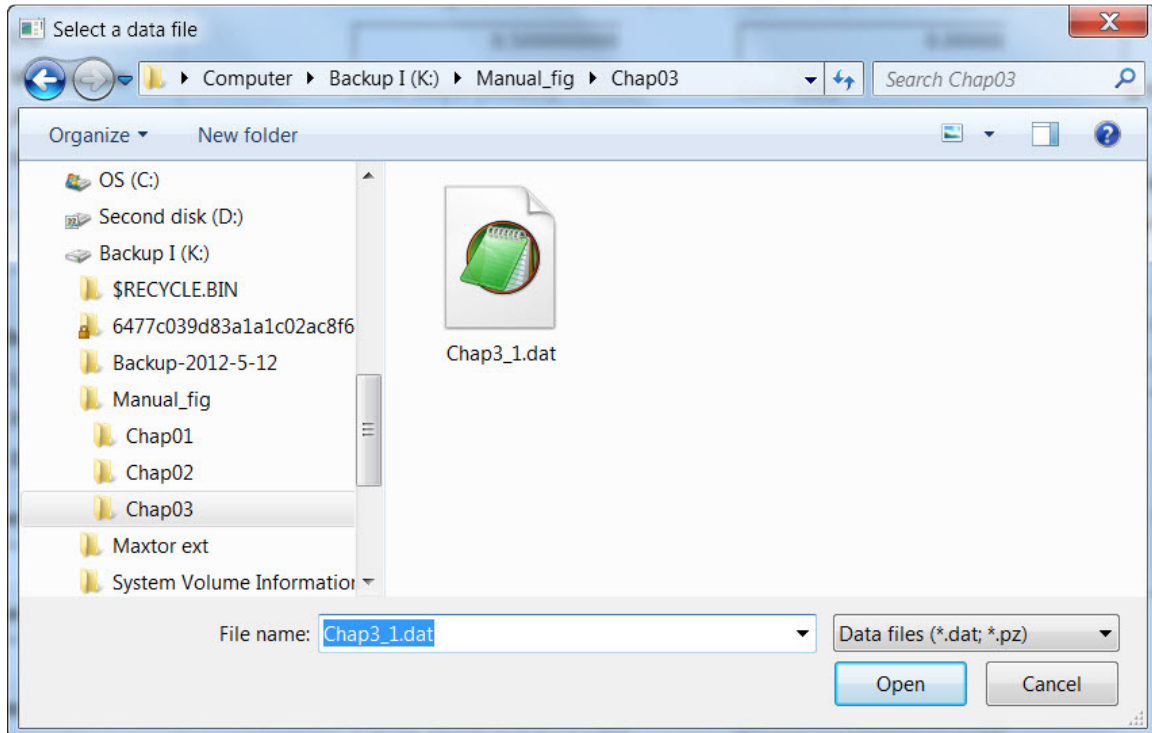
! Transmission zeros

0.0            25e3
0.0            75e3
```

Now we can start the data entry procedure.



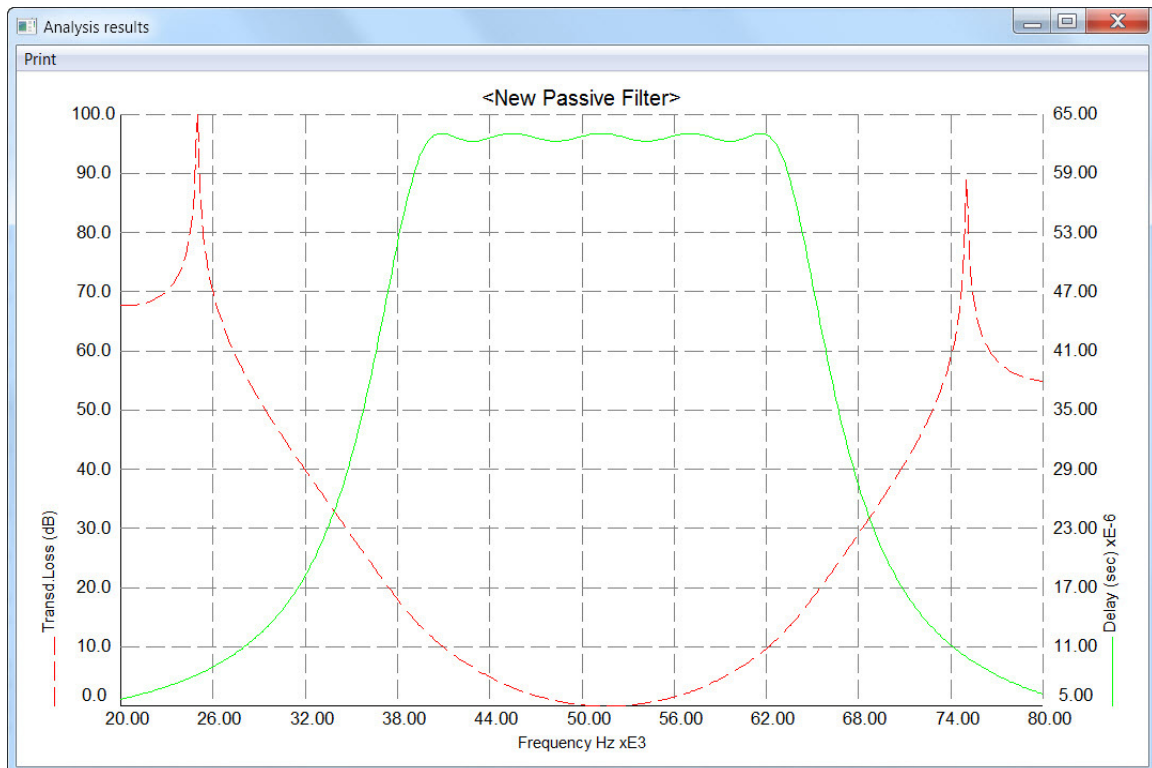
Under the **Detail Parameters** button we click on the **Read from file** button that brings up the file selection window:



Here we have selected our data file and clicking on **Open** we have the file read and the data stored in the proper places. This may be checked by clicking on the **Detail Parameters** button again and see the numerical data already there:

Function zeros (in rad/sec)			Transmission zeros (in Hz)		
	Real Part	Imaginary Part		Real Part	Imaginary Part
1	36.081650E+03	322.083900E+03	1	0.00000	25.000000E+03
2	34.431540E+03	360.032900E+03	2	0.00000	75.000000E+03
3	34.479940E+03	284.102500E+03	3	0.00000	0.00000
4	26.699640E+03	395.643700E+03	4	0.00000	0.00000
5	26.776260E+03	248.399200E+03	5	0.00000	0.00000
6	0.00000	0.00000	6	0.00000	0.00000
7	0.00000	0.00000	7	0.00000	0.00000
8	0.00000	0.00000	8	0.00000	0.00000
9	0.00000	0.00000	9	0.00000	0.00000
10	0.00000	0.00000	10	0.00000	0.00000
11	0.00000	0.00000	11	0.00000	0.00000

The rest of the synthesis is routine and we just show the analysis results here:



3.5 Signs of the reflection coefficient zeros

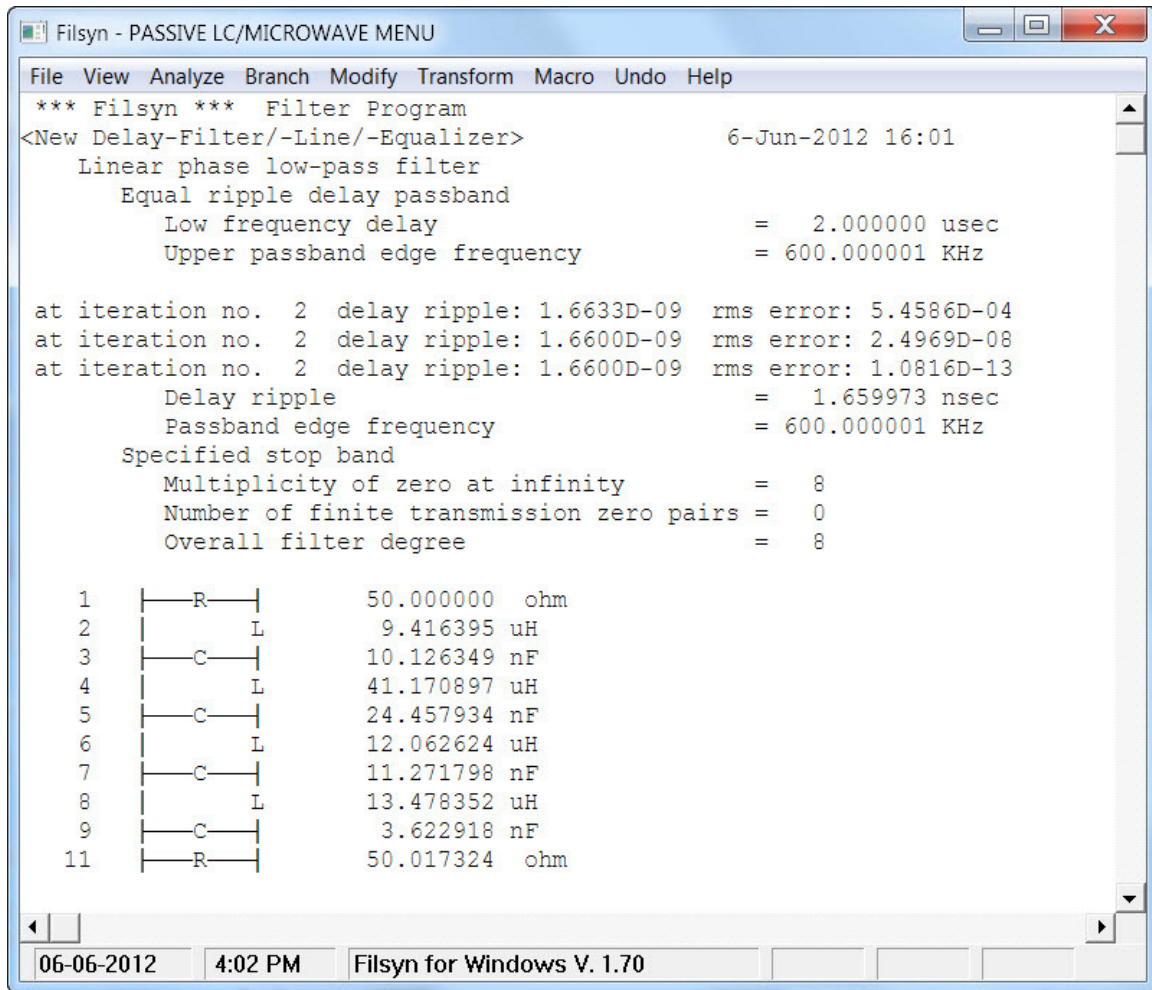
For most filters the zeros of the reflection coefficients (the roots of the $F(s)$ polynomial) will all be pure imaginary. Occasionally some or all of these roots become complex, and the signs of their real parts are arbitrary, but significant. As mentioned above, different

sign patterns lead to different – but equivalent – circuits. Consequently, these sign patterns might become useful for passive and microwave realizations and occasionally make the difference between a realizable and a non realizable network.

In such a case we may select a Hurwitz $F(s)$ polynomial (all real parts have the same sign) or a non-Hurwitz polynomial. In the non-Hurwitz case, the program offers a default sign pattern (yielding a minimum sum of real parts). If that is not satisfactory, we may specify the individual signs by entering the requisite number of + 1's and/or - 1's (see *Appendix E*).

We shall repeat the design of the equal ripple delay lowpass filter of degree 8, but with a non-Hurwitz characteristic function, using the computer-selected set of signs:

The resulting implementation, shown below, has the identical performance and same structure, but the element values are different (see page 77 above).



3.6 Predistortion

If a passive LC ladder realization is contemplated, one can predistort the filter characteristics to take into account the dissipation of the inductors and, possibly the capacitors. If the average inductor and capacitor Q 's are denoted by Q_L and Q_C respectively and evaluated at the upper passband edge frequency, we can pre correct for an average Q value given by:

$$Q = 2(Q_L^{-1} + Q_C^{-1})^{-1}$$

This value is entered in answer at the appropriate location. Leaving the zero default there bypasses the predistortion procedure.

There is a lower limit to the value of the Q for which we may be able to compensate and which will be printed along with the requested Q . The program then picks the larger of the two values, but if the lower limit is substantially higher than the requested value, the program stops and gives us the following choices:

1. Continuing with the computed minimum Q
2. Continue *without* predistortion
3. Terminate the design

It is recommended that predistortion be used with care, because of its drawbacks. The resulting structure becomes more sensitive to component variations, and in the lowpass and highpass cases, will yield a great disparity between the terminating resistances (regardless of the specifications). The return loss characteristics of the filters also become much worse in the passband.

This procedure is not available for microwave filters for mathematical reasons.

Predistorted Elliptic Filter

Here is what we get when we consider an elliptic highpass filter of degree 5, requesting predistortion with a very low average Q value of 5.

<New Filter>

Filter kind

- ☒ LC
- ☐ Microwave
- ☐ Active RC
- ☐ IIR Digital

Quarter-wave/Sampling freq (Hz) 0.00000

Filter type

- ☐ Low pass
- ☒ High pass
- ☐ Band pass

Lower passband freq (Hz) 500.000000E+00

Upper passband freq (Hz) 0.00000

Passband type

- ☐ Max. flat
- ☒ Equal ripple
- ☐ Functional
- ☐ Sloping

Band-edge loss/return loss (dB) 0.100000000

Loss Slope (dB/oct) 6.00000000

Flat loss (dB) 0.00000000

Function Type

- ☒ E
- ☐ F

Multiplier 0.00000000

Stopband

- ☐ Monotonic
- ☒ Equal min
- ☐ Placer
- ☐ Specified

Lower stopband freq (Hz) 450.000000E+00

Loss (dB) 20.0000000

Upper stopband freq (Hz) 0.00000

Loss (dB) 50.0000000

Detail Parameters

of zeros at zero 0

of zeros at infinity 0

of unit elements 0

Parametric

- ☒ Conventional
- ☐ Parametric
- ☐ Matching

R1 50.000000E+00

R2 50.000000E+00

ZS parameter -1

Q for predistortion 5

Predistortion

- ☒ Passband only
- ☐ Pass/stop bands

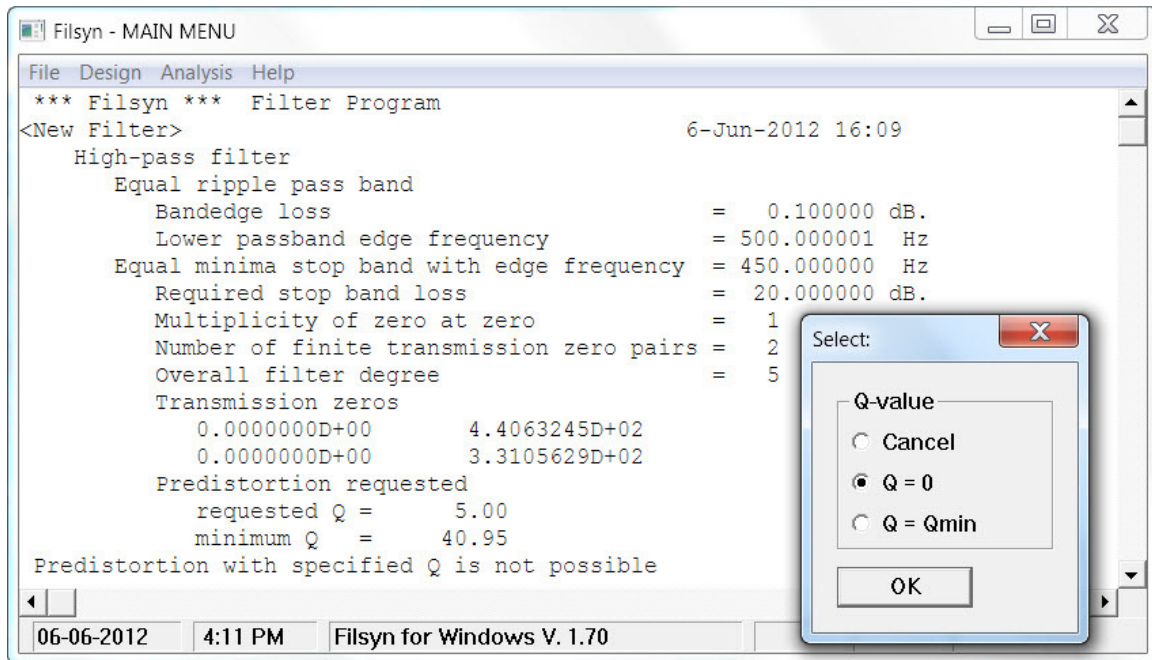
Shifted bandpass trans. Center frequency (Hz) 0.00000

☐ Odd parametric

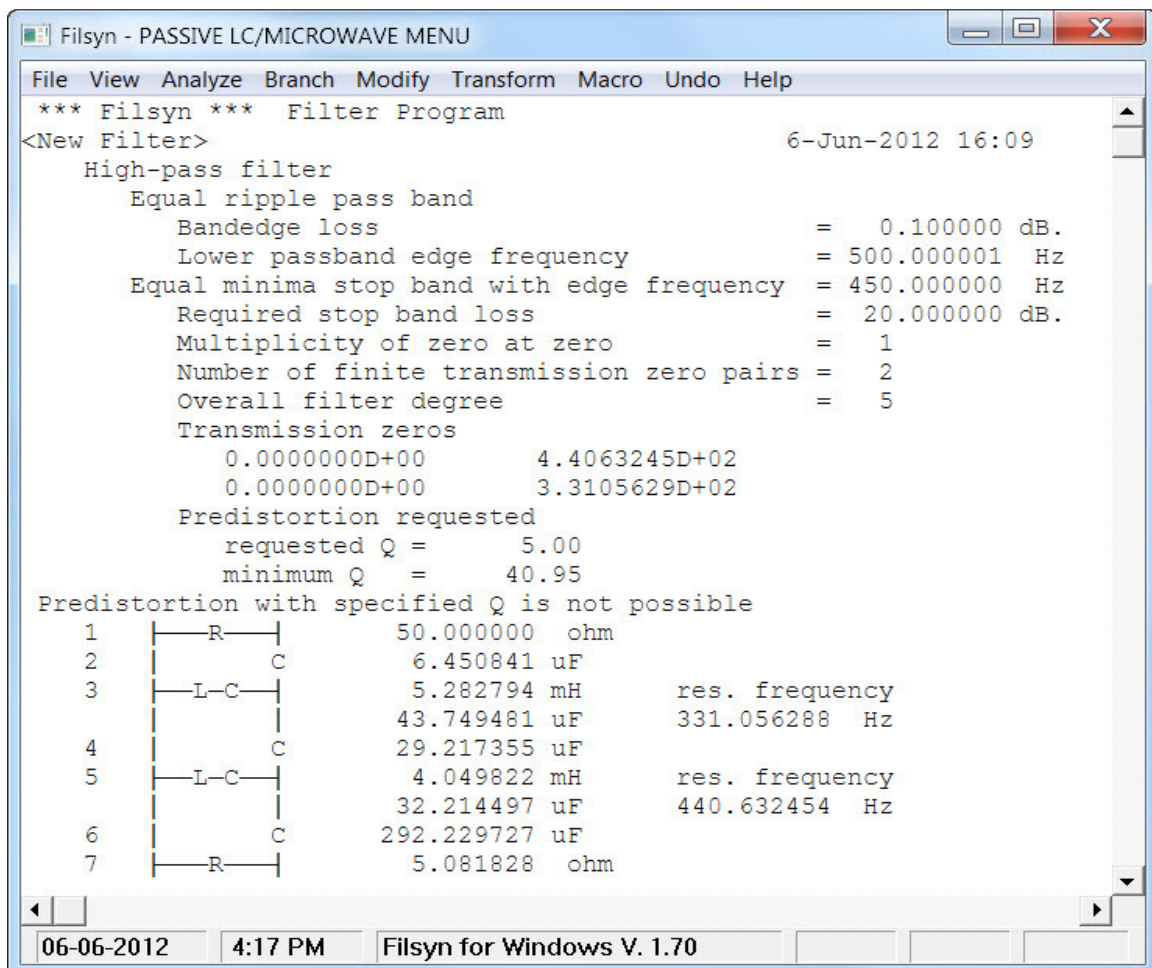
☐ Save design data

OK Cancel

Clicking on **OK**, we get to the point where we can see the computed minimum acceptable Q value of about 41 as well as the options available to us:



Accepting the computed minimum Q of 41, we obtain the implementation:



Note the unequal terminations.

Next we analyze this circuit with lossless capacitors but inductors with a Q of 20, that averages to a Q of about 40.

<New Passive Filter> 12-N... X

Start and end frequencies (Hz)

0.00000

3.000000E+03

Increment (Hz)

0.00000

or no. of frequency points

500

Loss in dB/length

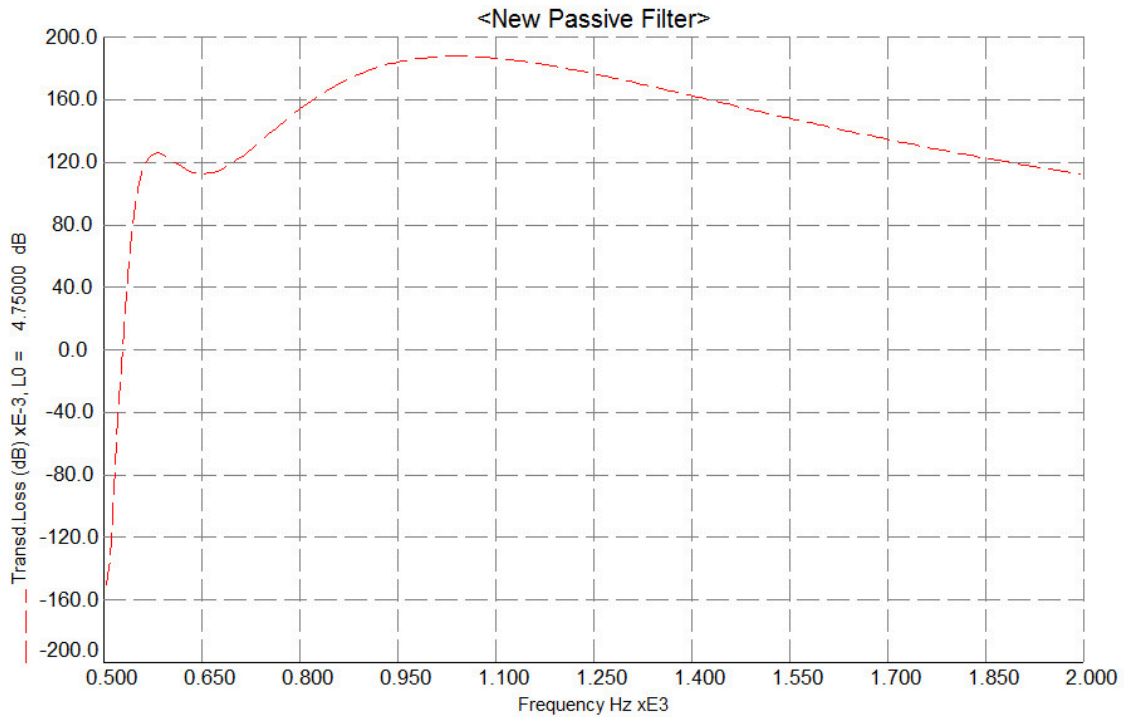
20

0.00000000

OK Cancel Next

We are also offered here the option of indicating whether the Q values are constant or they vary with the frequency. We selected the default, constant.

The computed response shows a very nice, flat passband loss. Plotting the significant part of the passband loss we can see the details near the edge of the passband:



The results show a flat loss in the passband of about 5 dB, with a ripple of about 0.2 dB. Without predistortion, this lossy filter would have a passband loss varying from 3.1 dB at 500 Hz down to zero. More details about predistortion are presented in *Appendix F*.

Upon requesting predistortion, one may encounter the message:

CHARACTERISTIC FUNCTION NUMERATOR HAS WRONG ZEROS

followed by an abort if the requested Q is close to its lower limit. In this case the design should be repeated with a somewhat higher requested Q .

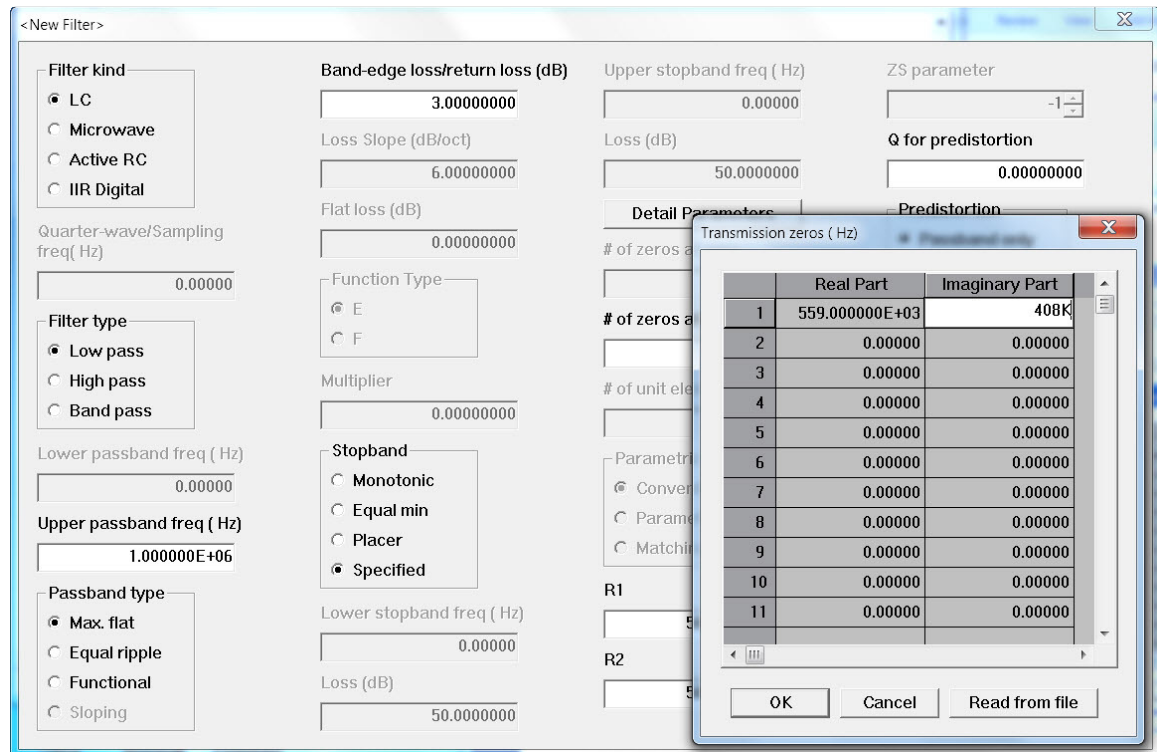
Also note that using finite Q values for analysis brings up another entry window, where we may specify a constant Q or one that varies linearly with the frequency. In this latter case the Q value specified is the value at the (upper) passband edge frequency.

3.7 Complex transmission zeros

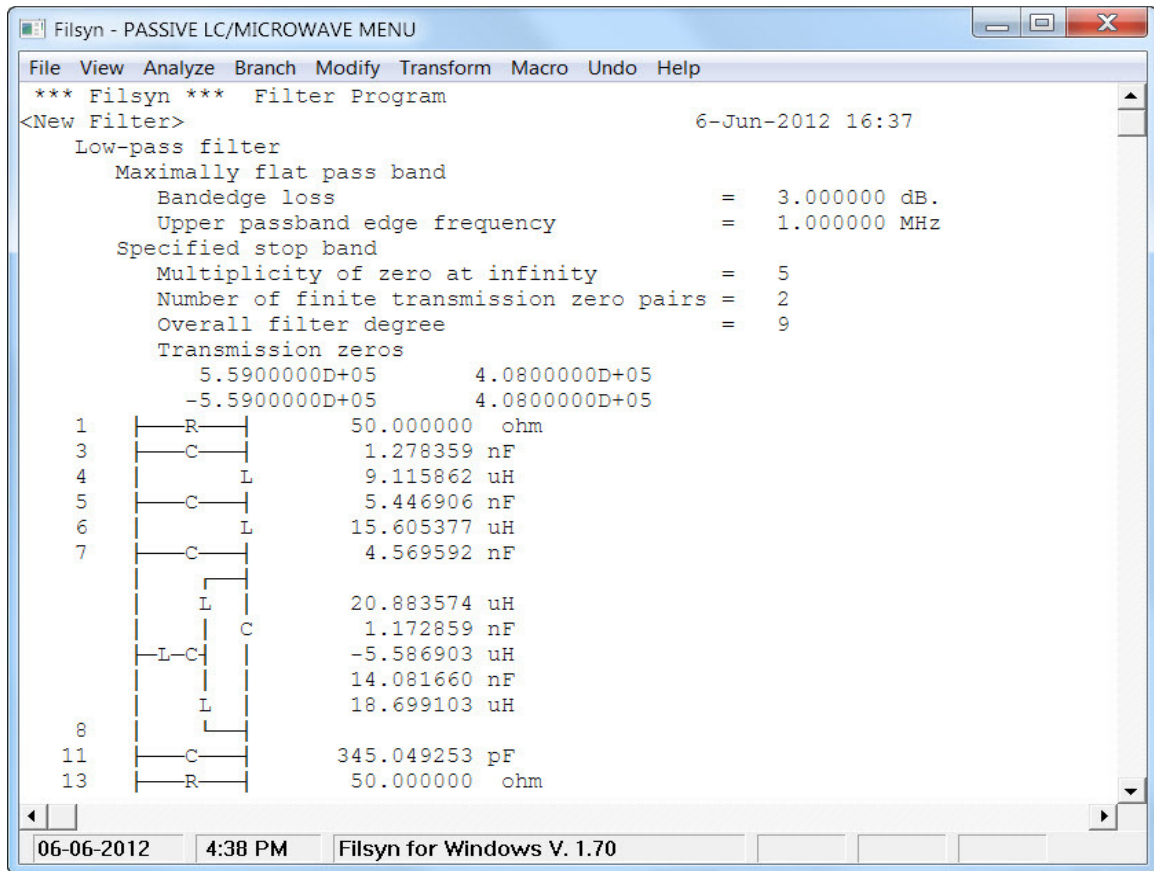
Complex transmission zeros can be used if the **Specified** stopband option is selected and a non-zero real part is entered for one or more transmission zeros. Each complex number entered this way adds *four* to the overall degree instead of the *two* a pure imaginary zero (real frequency zero) would add. This happens because the complex zero represents a quadruplet of four zeros in order to yield a pure even or odd $Q(s)$ polynomial necessary for LC implementation. Such a quadruplet will lead to a bridged-T or twin-T section embedded in the ladder and will contribute to both the loss and the delay characteristics. Its contribution to the delay is similar to that of a 2nd order delay equalizer section and as such, it can be used to yield a flatter delay characteristic, while contributing to the loss as well. These filter types can only be designed either by a trial-and-error method or by the use of functional input where the optimization has been done by a separate optimizer program before. (See also *Application Note 2* for additional information).

Filter with Complex Transmission Zero

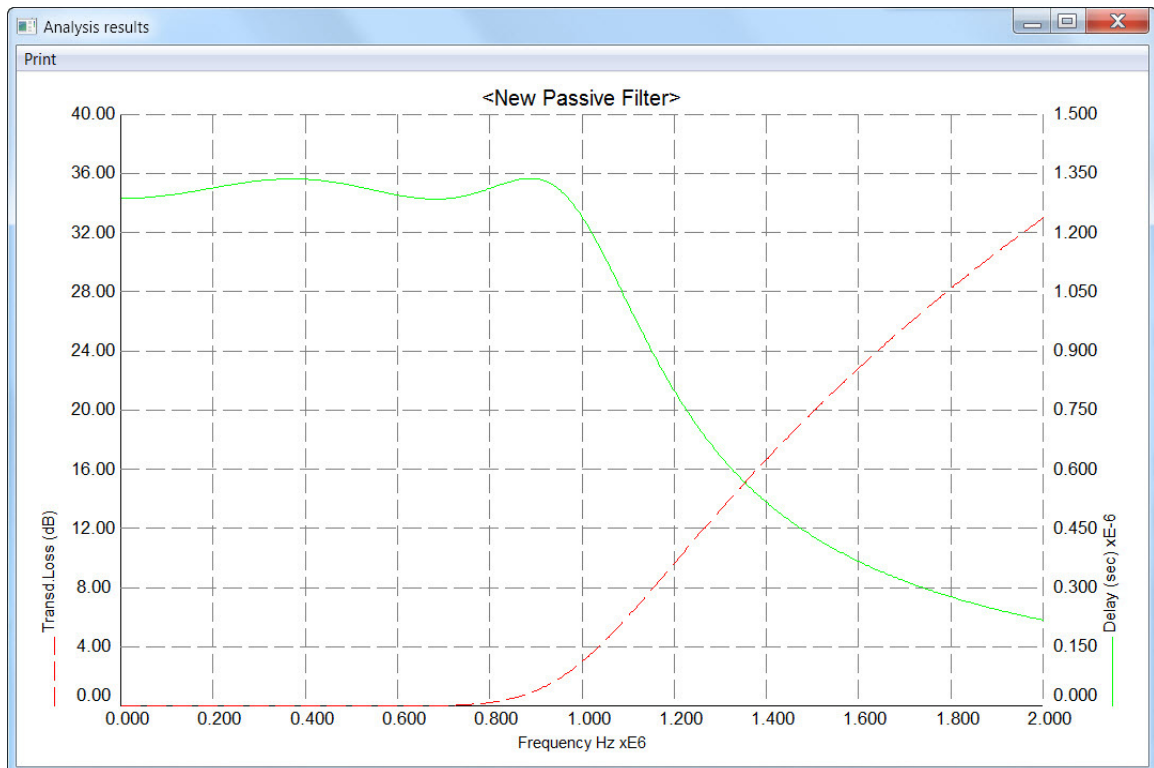
We will design a lowpass which, besides having a complex quadruplet of transmission zeros, is a 5th degree Butterworth filter. That is to say, it has maximally flat passband and a 5th order zero at infinity. The location of the complex zero was determined through a few trial runs of this example:



The resulting circuit contains a negative inductor, but that can be combined with two other inductors and implemented as a pair of coupled inductors with a specific coupling coefficient:



The performance of this design is as shown:



The complex zero contributes to the delay, making it much more constant than it would otherwise be and also contributes to the stopband loss, which is substantially higher than it would be for a straight 5th order Butterworth filter.

Using the maximally-flat passband type, one should *not* use very low band-edge loss (0.1 dB or less) to locate the edge of the passband. Such values could lead to severe numerical problems and loss of precision in the synthesis. Use instead the 3 dB point(s) or even higher band-edge loss values. One can easily compensate for this by widening the passband width accordingly.

3.8 Monotonic stopband

This option calculates the degree of lowpass, highpass and bandpass filters of the Butterworth and Chebyshev types in cases where the overall degree of the filter is *unknown*. Each stopband is specified by a single frequency-loss pair (point) defined by the filter requirements. If the requirements contain several points, they must be tried one at a time, and the one yielding the highest degree indicated by the program must be selected. Note that if the filter degree is known (even if monotonic), the **Specified** option should be used and monotonic stopband(s) will be obtained if no finite transmission zeros are specified.

Microwave monotonic bandpass filter

This example focuses on a microwave bandpass filter with passband from 2 to 3 GHz. The filter should have at least 25 dB of loss below 1.5 GHz and at least 35 dB above 4 GHz, with 5 GHz as the quarter-wave frequency. This option works only for conventional filters, as specified below. Also, for microwave filters we must specify the number of unit elements we want to use (zero is acceptable). The program will determine the number of zeros at zero frequency and at the quarter-wave frequency, which corresponds to infinite frequency in the lumped case. The data input screen is as follows:

< New Filter >

Filter kind

- ☐ LC
- ☒ Microwave
- ☐ Active RC
- ☐ IIR Digital

Quarter-wave/Sampling freq (Hz)

5.000000E+09

Filter type

- ☐ Low pass
- ☐ High pass
- ☒ Band pass

Lower passband freq (Hz)

2.000000E+09

Upper passband freq (Hz)

3.000000E+09

Passband type

- ☐ Max. flat
- ☒ Equal ripple
- ☐ Functional
- ☐ Sloping

Band-edge loss/return loss (dB)

0.100000000

Loss Slope (dB/oct)

6.00000000

Flat loss (dB)

0.00000000

Function Type

- ☒ E
- ☐ F

Multiplier

0.00000000

Stopband

- ☒ Monotonic
- ☐ Equal min
- ☐ Placer
- ☐ Specified

Lower stopband freq (Hz)

1.500000E+09

Loss (dB)

25.0000000

Upper stopband freq (Hz)

4.000000E+09

Loss (dB)

35.0000000

Detail Parameters

of zeros at zero

0

of zeros at FQ or FS/2

0

of unit elements

4

Parametric

- ☒ Conventional
- ☐ Parametric
- ☐ Matching

R1

50.000000E+00

R2

50.000000E+00

ZS parameter

-1

Q for predistortion

0.00000000

Predistortion

- ☒ Passband only
- ☐ Pass/stop bands

Shifted bandpass trans.

Center frequency (Hz)

0.000000

☐ Odd parametric

☐ Save design data

OK Cancel

After rearranging the branches of the computer-generated configuration, we accidentally get a nearly symmetrical structure:

Filsyn - PASSIVE LC/MICROWAVE MENU

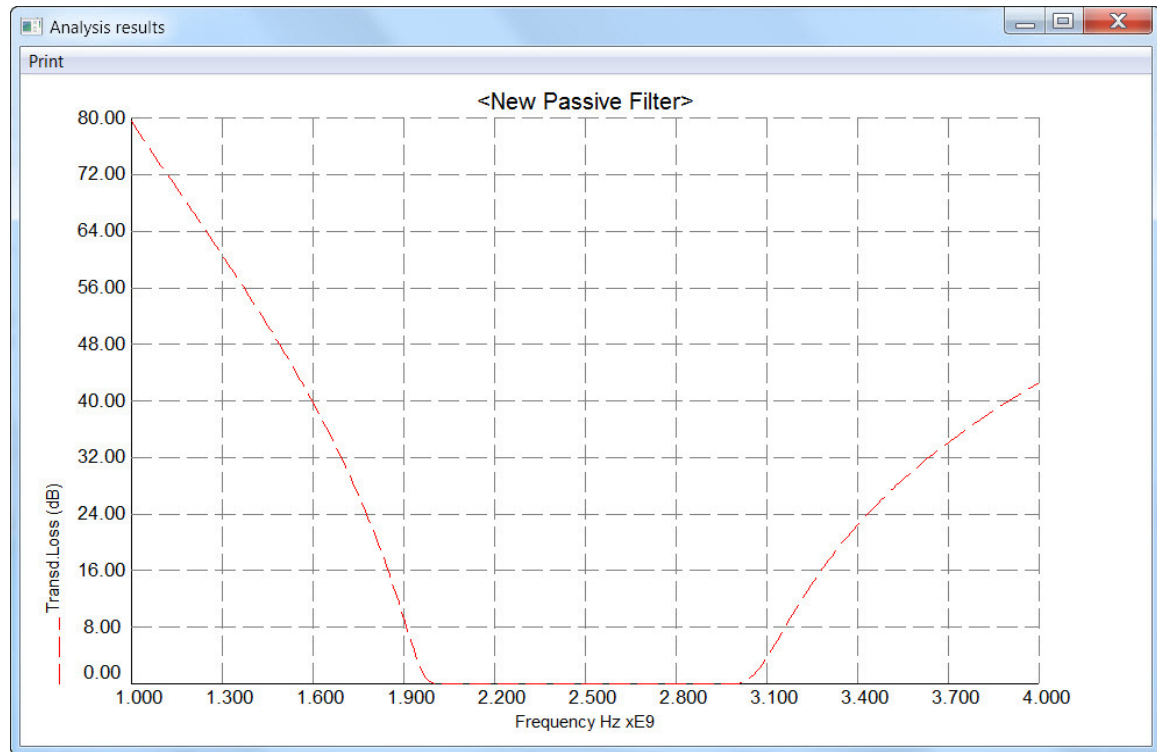
File View Analyze Branch Modify Transform Macro Undo Help

**** All values are impedances ****

1	—R—	50.000000	ohm
3	—L—	28.348176	ohm
	* * *		
5	* u e *	25.354386	ohm
	* * *		
8	C	204.824406	ohm
	* * *		
11	* u e *	207.040349	ohm
	* * *		
17	—L—	16.270879	ohm
	* * *		
19	* u e *	15.712662	ohm
	* * *		
26	C	204.824406	ohm
	* * *		
29	* u e *	216.682073	ohm
	* * *		
35	—C—	28.714302	ohm
37	—L—	28.348176	ohm
39	—R—	50.000000	ohm

06-06-2012 5:27 PM Filsyn for Windows V. 1.70

The summary below tells us that the overall filter degree must be 10 (conventional bandpass filters must have an even overall degree). We also see that there are four zeros at zero and two at quarter-wave frequency. The loss plot shows that all requirements have been met:



In monotonic bandpass design the multiplicity of the zero at zero or at infinity may turn out to be zero. This leads to a bandpass that either looks like a highpass or a lowpass respectively, with a poor termination ratio in a passive implementation. If this is not acceptable, one should repeat the design as a **specified** one, and enter unity for the previously zero multiplicity, while reducing that of the other by one. This will work in all but the most borderline cases, where we find it necessary to increase the other multiplicity by one rather than decreasing it. As mentioned above the sum of these multiplicities must be even for conventional bandpass filters. Alternatively, we could try a **parametric** case with arbitrary, nonzero, multiplicities, at least one of which must be odd, if that is acceptable.

3.9 Parametric bandpass

These filter types are explained in detail in *Appendix C*. The primary use of this type is in passive LC realization, where it may reduce the number of inductors used at the expense of capacitors and where awkward inductor values can be avoided in narrow band cases. Parametric filters can be of even or odd degree, as opposed to conventional band passes. In the case of even degree parametric filters, extreme termination is not allowed. Therefore, in multiplexers, either a conventional or an odd degree parametric design must be used. For paralleled multiplexers in particular, bandpass filters must have a transmission zero at infinity with even multiplicity.

The second version of our first example on page 66 above, is a good example of a parametric design.

Parametric multiplexer

Most multiplexers are filters connected in parallel at one of their ends. As shown in *Application Note 3*, a filter to be used as a multiplexer, must have a short-circuited end, which can be obtained by specifying $R_2 = 0.0$. Note that the input termination may *not* be zero. This will, in fact, be one of the outputs of the multiplexer. Specify therefore a bandpass as follows:

The screenshot shows the 'New Filter' dialog box with the following settings:

- Filter kind:** LC
- Quarter-wave/Sampling freq (Hz):** 0.00000
- Filter type:** Band pass
- Lower passband freq (Hz):** 2.50000E+03
- Upper passband freq (Hz):** 5.00000E+03
- Passband type:** Equal ripple
- Band-edge loss/return loss (dB):** 0.500000000
- Loss Slope (dB/oct):** 6.000000000
- Flat loss (dB):** 0.000000000
- Function Type:** E
- Multiplier:** 0.000000000
- Stopband:** Placer
- Lower stopband freq (Hz):** 0.00000
- Loss (dB):** 50.000000000
- Upper stopband freq (Hz):** 0.00000
- Loss (dB):** 50.000000000
- # of zeros at zero:** 1
- # of zeros at infinity:** 2
- # of unit elements:** 0
- Parametric:** Parametric
- R1:** 50.000000E+00
- R2:** 0
- ZS parameter:** -1
- Q for predistortion:** 0.000000000
- Predistortion:** Passband only
- Shifted bandpass trans. Center frequency (Hz):** 0.00000
- Odd parametric:** unchecked
- Save design data:** unchecked

The **Detail Parameters** window only contains stopband specifications:

Filter Parameters

Requirements			Fixed zeros			Movable zeros		
	Frequency (Hz)	Loss (dB)		Frequency (Hz)	Multiplicity		Frequency (Hz)	Multiplicity
1	2.000000E+03	25.00000000	1	0.000000	0	1	0.000000	0
2	6.000000E+03	40.00000000	2	0.000000	0	2	0.000000	0
3	0.000000	0.00000000	3	0.000000	0	3	0.000000	0
4	0.000000	0.00000000	4	0.000000	0	4	0.000000	0
5	0.000000	0.00000000	5	0.000000	0	5	0.000000	0
6	0.000000	0.00000000	6	0.000000	0	6	0.000000	0
7	0.000000	0.00000000	7	0.000000	0	7	0.000000	0
8	0.000000	0.00000000	8	0.000000	0	8	0.000000	0
9	0.000000	0.00000000	9	0.000000	0	9	0.000000	0
10	0.000000	0.00000000	10	0.000000	0	10	0.000000	0
11	0.000000	0.00000000				11	0.000000	0

or no. of zeros below passband

and/or no. of zeros above passband

OK Cancel

The design is uneventful and yields the following summary and circuit:

Filsyn - PASSIVE LC/MICROWAVE MENU

File View Analyze Branch Modify Transform Macro Undo Help

*** Filsyn *** Filter Program

<New Filter> 6-Jun-2012 17:45

Band-pass filter

Equal ripple pass band

Bandedge loss = 0.500000 dB.

Lower passband edge frequency = 2.500000 KHz

Upper passband edge frequency = 5.000000 KHz

Parametric bandpass type

Specified stop band

Multiplicity of zero at zero = 1

Multiplicity of zero at infinity = 2

Number of finite transmission zero pairs = 3

Overall filter degree = 9

Transmission zeros

0.000000D+00 1.8875913D+03

0.000000D+00 1.2572398D+04

0.000000D+00 6.1532477D+03

Value of freal = 3.535534 KHz

*** termination is extreme ***

1 |---R---| 50.000000 ohm

3 |---C---| 110.745407 nF

4 |---L---| 961.771911 uH res.frequency

6 |---C---| 695.599687 nF 6.153248 KHz

7 |---L---| 2.618310 uF

9 |---C---| 3.342230 mH res. frequency

10 |---L---| 2.127101 uF 1.887591 KHz

12 |---C---| 1.204164 uF

13 |---L---| 4.599069 mH res.frequency

14 |---C---| 34.844488 nF 12.572398 KHz

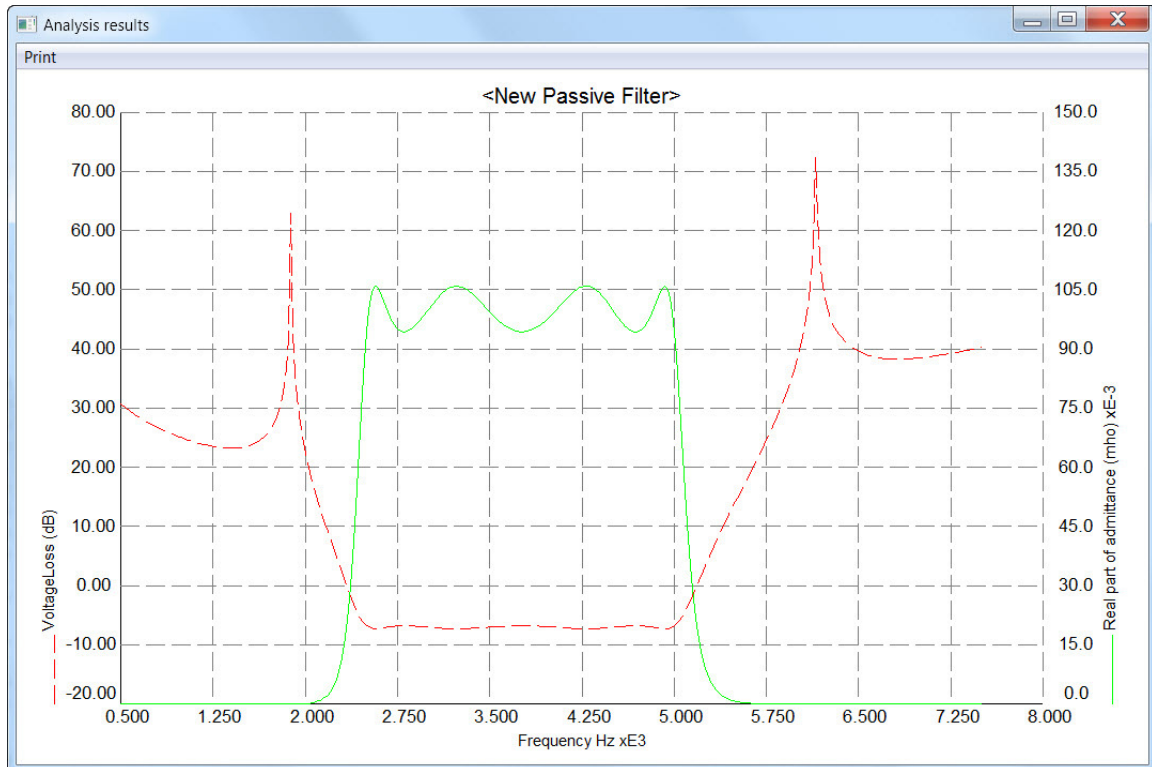
15 |---L---| 457.233853 nF

16 |---C---| 1.839339 uF

17 |---L---| 918.136083 uH

06-06-2012 5:46 PM Filsyn for Windows V. 1.70

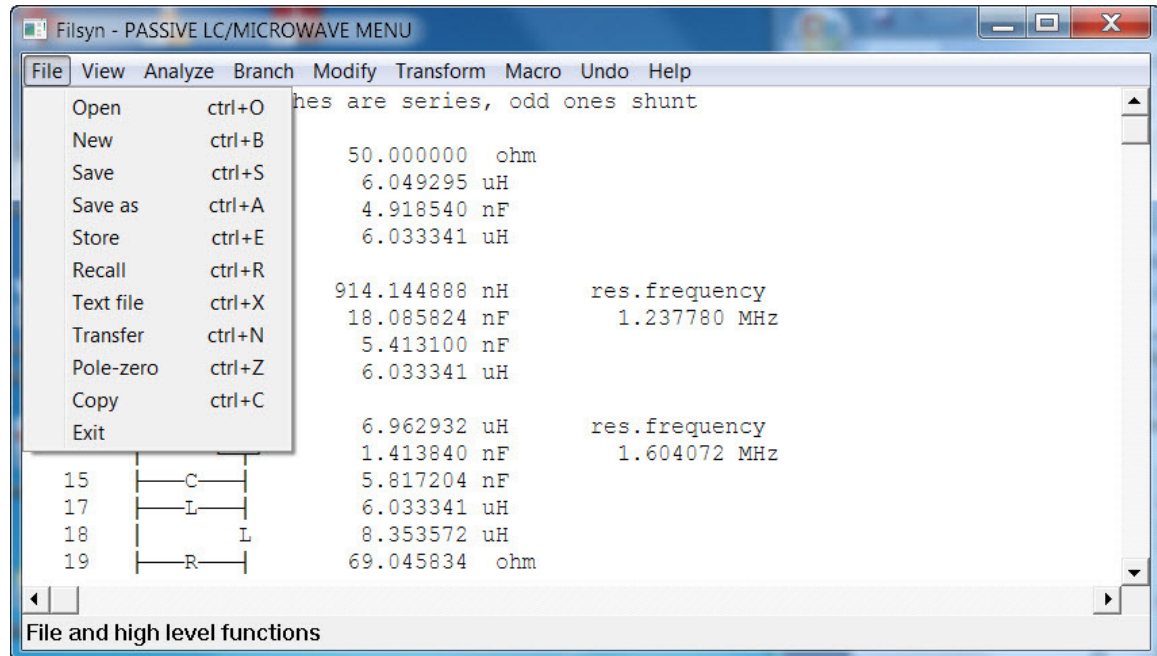
Note the warning at the top of the schematic which calls our attention to the termination condition, and the fact that the output termination is missing, being zero. Since the last branch is a series one, the output termination should be a short-circuit; otherwise the last series branch would have no influence on the filter behavior. This means that the filter should be driven by an ideal voltage source connected to the *output* (lower end) terminals. We can request a quick analysis to check the circuit, and we obtain the following results:



In accordance with our request, the calculated loss is voltage loss, which is due to the short-circuited output which is, in fact now the input side (a voltage source). This time we obtain a flat gain in the passband. Also, instead of the impedance at the output end, we calculate and tabulate the admittance as the more useful quantity and show the plot of the real part of the output admittance. This quantity is the significant one, when we connect this filter in parallel with others and should be approximately constant in the passband and go to zero in the stopbands.

4. PASSIVE LC/MICROWAVE ANALYSIS

This is the most complex and powerful of all the analysis segments with a large number of – sometimes confusing – set of menu options. Let us first go through these options quickly. We have a bandpass filter in the background, but forget about that for the time being:



4.1 The 'File' menu option has a number of submenu items:

1. **Open** would open another set of filter data stored in a file, but as a warning displays it, the current filter data would be overwritten.
2. **New** permits us to restart the synthesis or start a completely new one without the need for stopping and restarting the program.
3. **Save** saves the current filter data into the file we have read the current filter from. If the filter was not read from a file, the option is the same as the 'Save as' option.
4. **Save as** saves the filter data into a new file.
5. **Store** saves the current filter data into a temporary storage, which is deleted when we leave the program.
6. **Recall** recalls the data from the temporary storage.
7. **Text file** saves the filter data in a text file for documentation purposes. This file can be displayed and edited in any text editor, but cannot be read back into the program.
8. **Transfer** writes the filter data into a file, which can be read into one of several programs, including Spice, Touchstone, Pcfilt, Matlab and others.

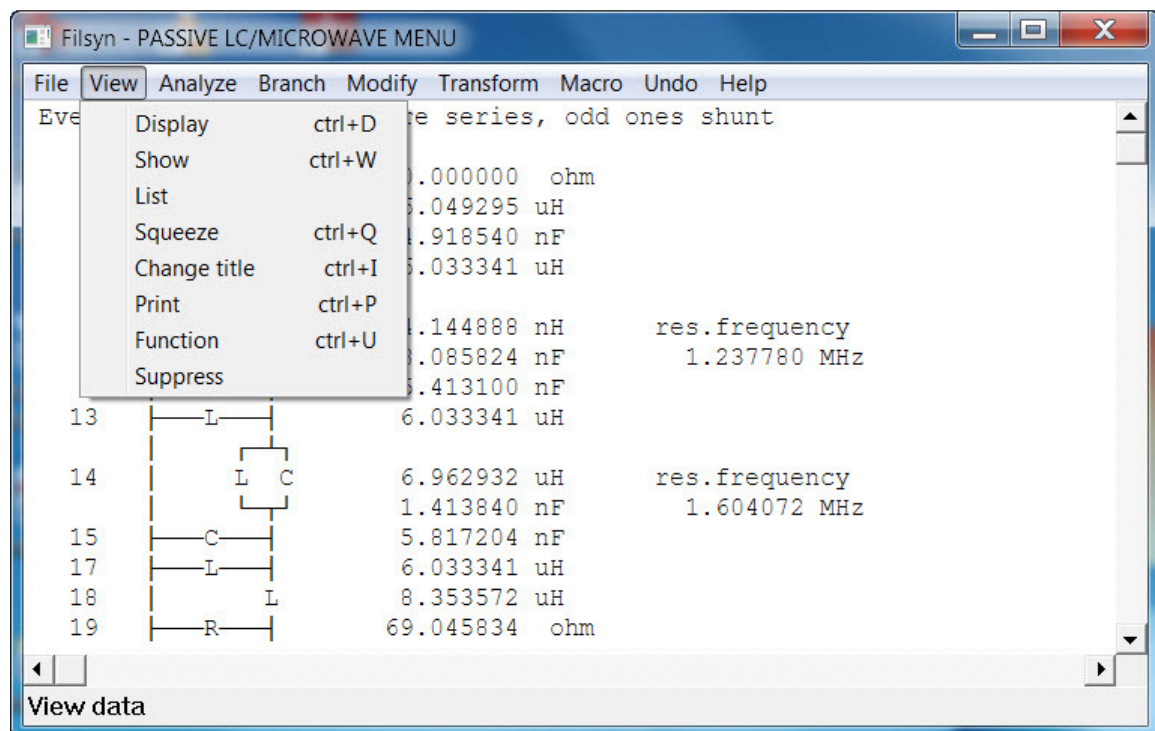
9. **Pole-zero** writes the pole-zero data of the filter (if available) into a file. If the filter does **not** contain a delay equalizer, the file will have a “.pz” extension and can be read back into **Filsyn** using the **Functional input** option. If however, the filter has been delay-equalized, the data is written to a file with a “.dat” extension and it may only be used for documentation purposes.

10. **Copy** can be used to transfer a part of the display, selected by the mouse in the usual manner, to the clipboard. We can select areas that have already scrolled off the screen by scrolling them back down.

11. **Exit** of course, terminates the program.

4.2 The ‘View’ menu options

The next set of menu items are related to viewing the information we have:

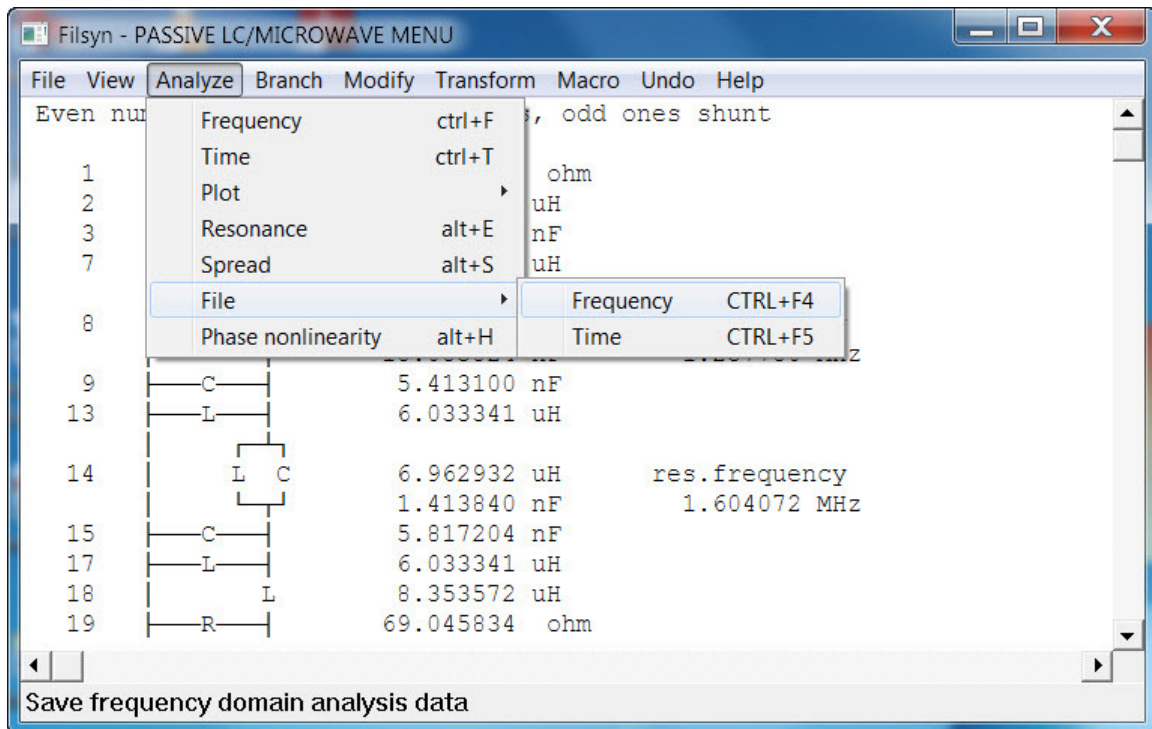


1. **Display** displays the design data, i.e. a summary of the specifications for the current filter.
2. **Show** displays the current circuit diagram and element values.
3. **List** lists the internal data used to describe the circuit. This menu item is useful mainly for debugging purposes. A few script files use this menu occasionally.
4. **Squeeze** is used to renumber the branches and sections in the circuit, eliminating large gaps in the numbers.

5. **Change title** does precisely that, since the program does not let us specify the title associated with any design before.
6. **Print** turns the print function on or off. When it is on, whatever goes to the screen also goes to either the printer or to a text file. Note that the printer will only print when a pageful of material has accumulated. If you need to print what is already on the screen, then use the **File->Copy** menu (see above) and paste the clipboard's content into Notepad, or any other text editor and print from there.
7. **Function** displays the transfer function data of the filter, if available.
8. **Suppress** turns the automatic display of the modified filter on or off. It is on by default and is inadvisable to turn it off. This menu item should be used exclusively in script files, where it prevents the display of a very large amount of unnecessary information.

4.3 The 'Analyze' menu options

Next we come to the Analyze menu that has all the expected submenus:



1. **Frequency** is the frequency domain analysis, where in the subsequent data entry window we can specify the starting and ending frequencies and either the increment or the number of frequencies. Up to five ranges of frequencies may be specified and the maximum total number of frequency points is 501. You may also specify a finite inductor and capacitor Q value for lossy analysis. The results

can be displayed in tabular form or not (default) but they are saved internally until overwritten, or we leave the program.

2. **Time** is the time-domain analysis option – when available – with similar limits. It computes the impulse and the step responses of the current filter. Lossy analysis is not available in the time domain, since we use the transfer function for this purpose.
3. **Plot** does just that, anything that is computed in the frequency or time domain can be plotted. The plot can show up to two responses, but can be called any number of times. Plots can also be sent to the printer.
4. **Resonance** computes the resonant frequency of an L – C pair, when that resonance is not displayed.
5. **Spread** computes and displays the element value spread of the current filter. This is a list of the minimum, maximum and average values of all inductors, capacitors and unit elements – if appropriate – as well as the ratios of the maximum and the minimum values.
6. **File** saves the frequency or time domain analysis results to a text file for documentation purposes. Note that in the plots the loss values are always limited to not more than 100 dB, the return loss to not more than 50 dB even if the computed values are higher. The data saved in a file will retain the correct computed values. You may also select which columns are to be placed in the file.
7. **Phase nonlinearity** When the filter has been analyzed, one can display the phase nonlinearity in a frequency range specified. Nonlinearity can be computed in one of three ways: a) osculating at the center frequency, b) least-mean-square approximation or c) straight line through the end points.

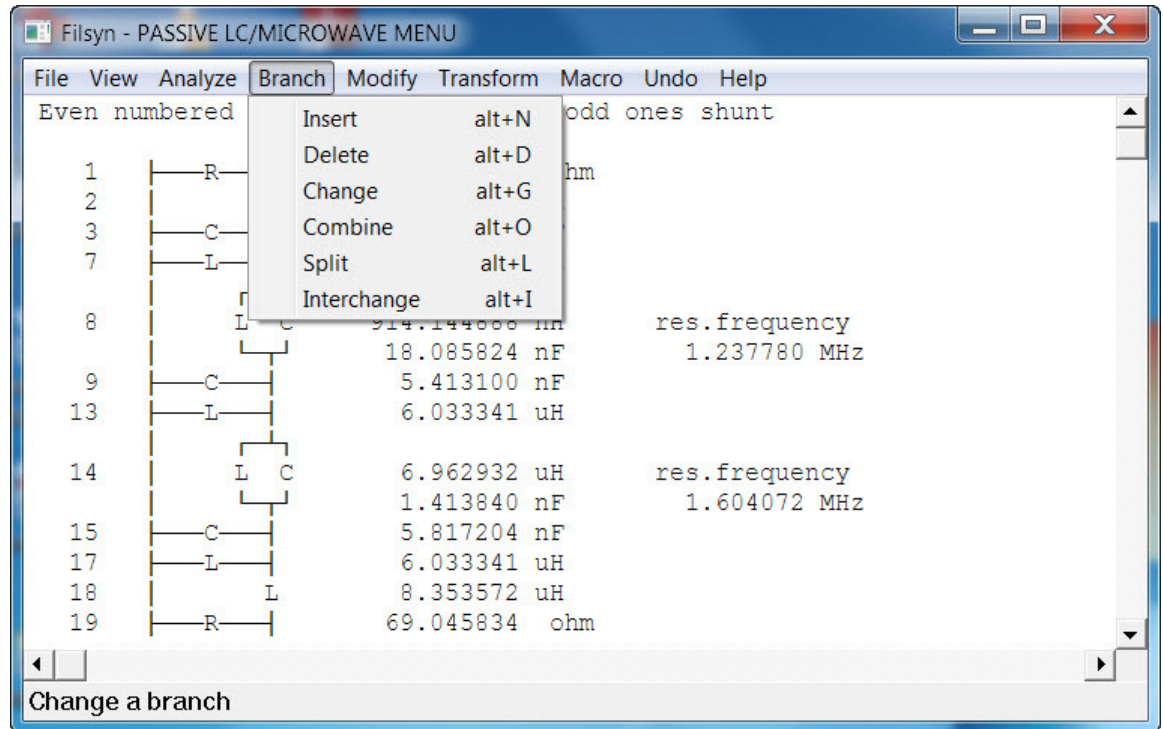
The next set of menu items deal with individual branches in the circuit:

4.4 The '*Branch*' menu options

Most of these menu items are self-explanatory:

1. **Insert** inserts a one, two or three-element branch, a unit element (where appropriate) or a loss-equalizer section. Note that unit elements may be inserted only next to a terminating resistance and must have the same impedance as the termination.
2. **Delete** deletes any numbered branch or section.
3. **Change** lets us change the element values of any one- or two-element branch.
4. **Combine** combines two consecutive series or shunt L's or C's.

5. **Split** splits a single L or C into two, where one of the values can be specified or it can be the same as another element, or the branch can be split into two halves.



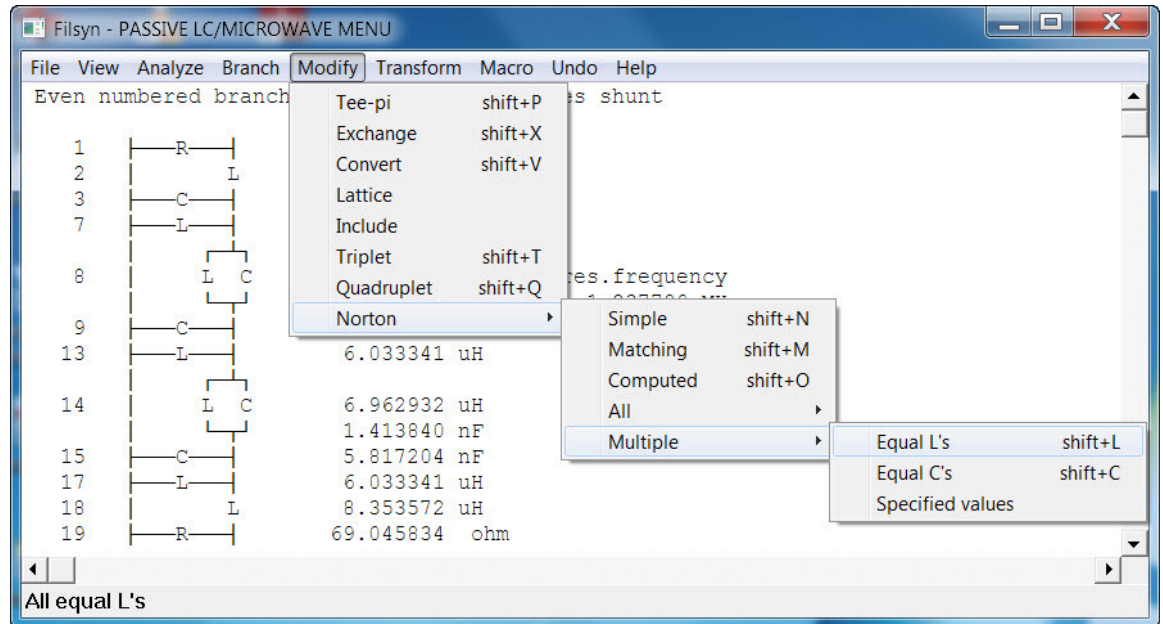
6. **Interchange** interchanges the location of two consecutive series or shunt branches. One of these may be a unit element (for microwave filters), in which case the operation is, in fact the use of one of the Kuroda's identities (see later). If, in addition, the other branch is a resonant one, we apply Levy's extension of those identities.

Now we come to the most complex of the operations. The **Modify** menu option includes operations that change part of the circuit. The first few items are relatively simple:

4.5 The 'Modify' menu options

1. **Tee-pi** converts a Tee- or Pi-section of inductors or capacitors into the other form. If the center branch of the Tee or Pi is a resonant branch, the conversion uses Brune's equivalence. We also have the option of selecting inductive or capacitive side branches in the result but only in bandpass filters. Note however, that one of these side branches will be negative, hence a similar positive branch will be needed to absorb that.
2. **Exchange** exchanges the locations of two resonant branches separated by at most a single capacitor. If they are separated by an inductor, take the dual first, then perform the operation, then take the dual again.
3. **Convert** converts a three-element branch to the other equivalent form.

4. **Lattice** takes a few branches containing one or two finite transmission zeros and converts them to a lattice.



5. **Include** is closely related to the item above. If we already have a lattice and it has a capacitor in series or shunt on both sides, this menu item will shift them inside the lattice such that at least one of them will disappear.
6. **Triplet** converts three ladder branches containing a transmission zero into a triplet form, where the transmission zero is now implemented by a single-element branch coupling nonadjacent shunt branches (see ref. [53]).
7. **Quadruplet** is similar, except two transmission zeros are now implemented by up to three bridging branches coupling nonadjacent shunt branches. See the same reference. This option can also be used to replace bridged-T circuits, which implement complex quadruplets of zeros.
8. **Norton** is where the fun begins. Basically this is the standard Norton (or impedance) transformation familiar to all filter designers. First note that these transformations can only be used for bandpass filters or microwave highpass filters that also contain unit elements. There are a number of special cases here, indicated in the screen above, with options as follows:

a) **Simple** is the usual simple impedance transformation, where we specify two one- or two-element branches and perform the operation. Here we may either specify the value of the new output termination we need, or the value of the transformation ratio, or just leave them at zero, when the maximum available transformation will be performed.

Fig. 1 shows several forms of the basic Norton equivalence used for the transformations. Z_r represents the rest of the circuit on the right side. We can see that the top half of the

figure raises the impedance level on the right, while the bottom half lowers it. The branches Z and aZ (or Z/a) can be any pair of inductors, capacitors or resonant branches. If resonant branches are used, they must both be either parallel or series resonant branches and they must resonate at the same frequency. This is easily achieved by splitting one of the elements, a step the program performs automatically.

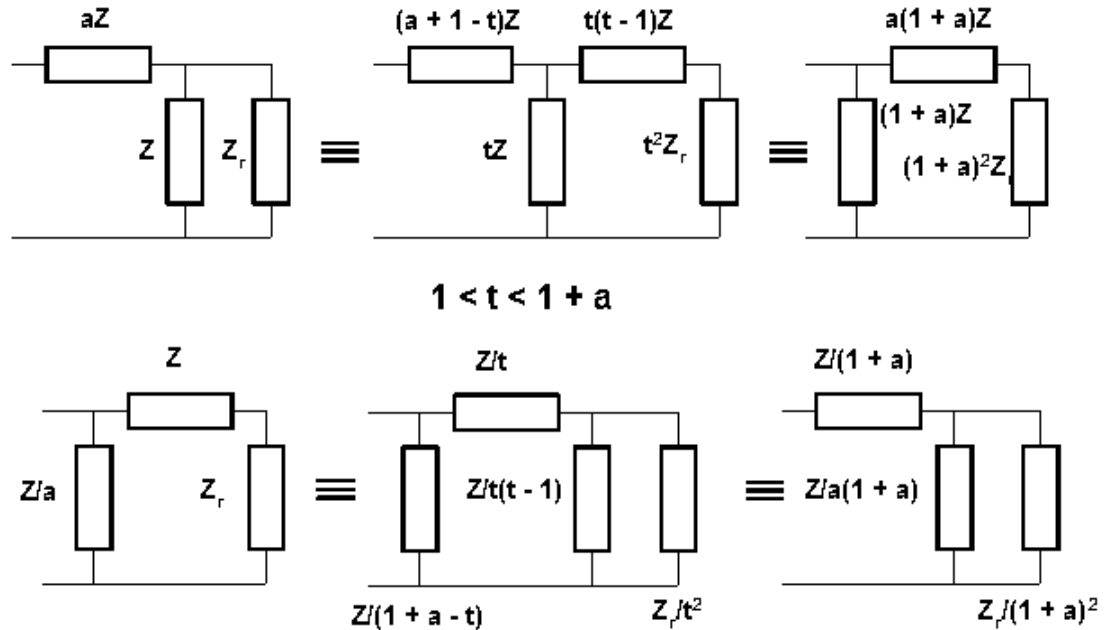


Fig. 1 Norton's equivalences

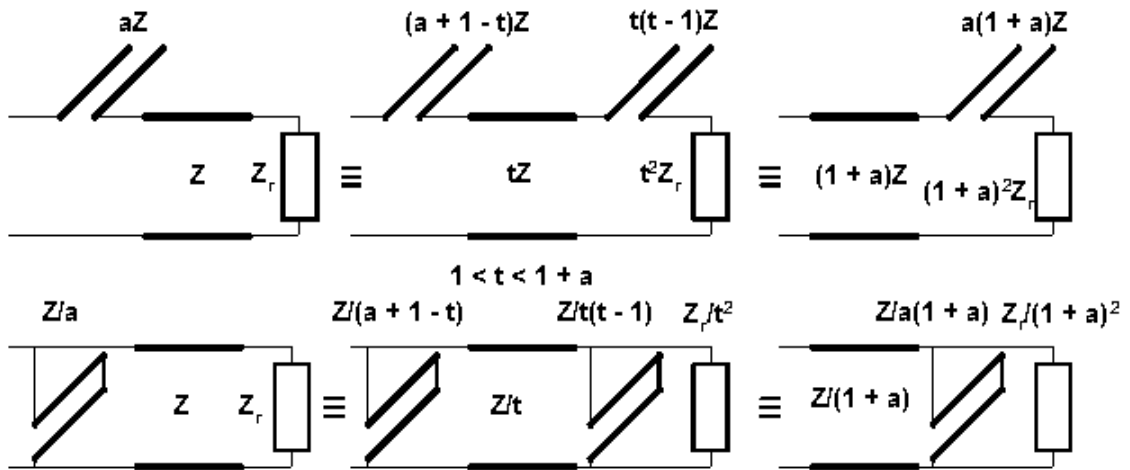
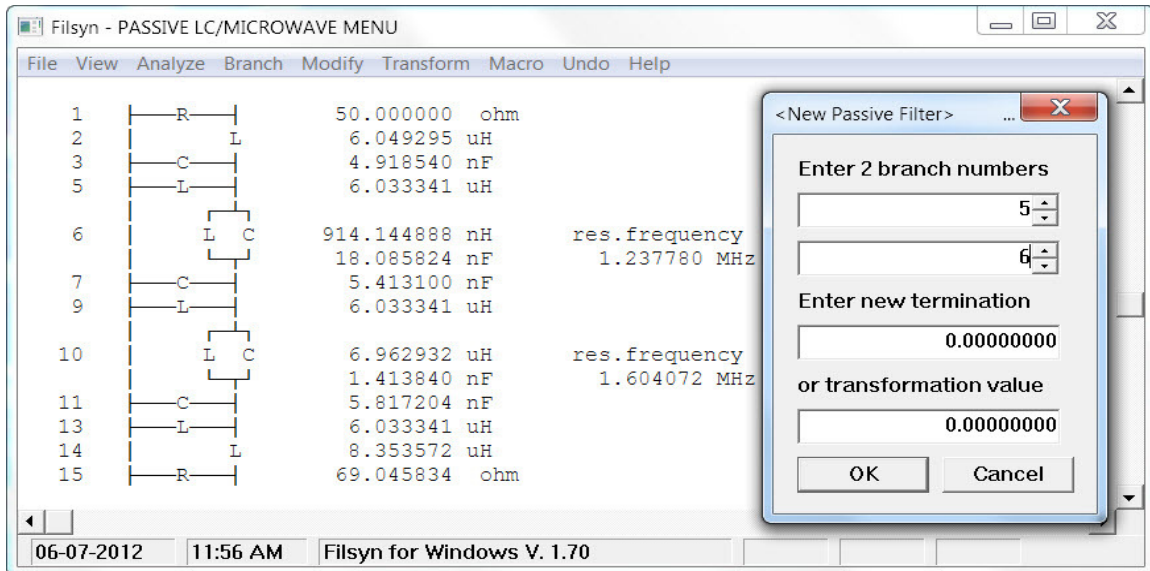


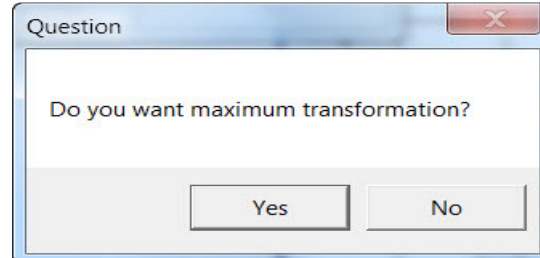
Fig. 2 Kuroda's highpass identities

If we look, for comparison, at Fig. 2 which shows (partial and total) highpass Kuroda's identities, we see immediately, that the equations are identical to those in Fig. 1. Therefore they perform exactly the same operations and can be used in the same manner.

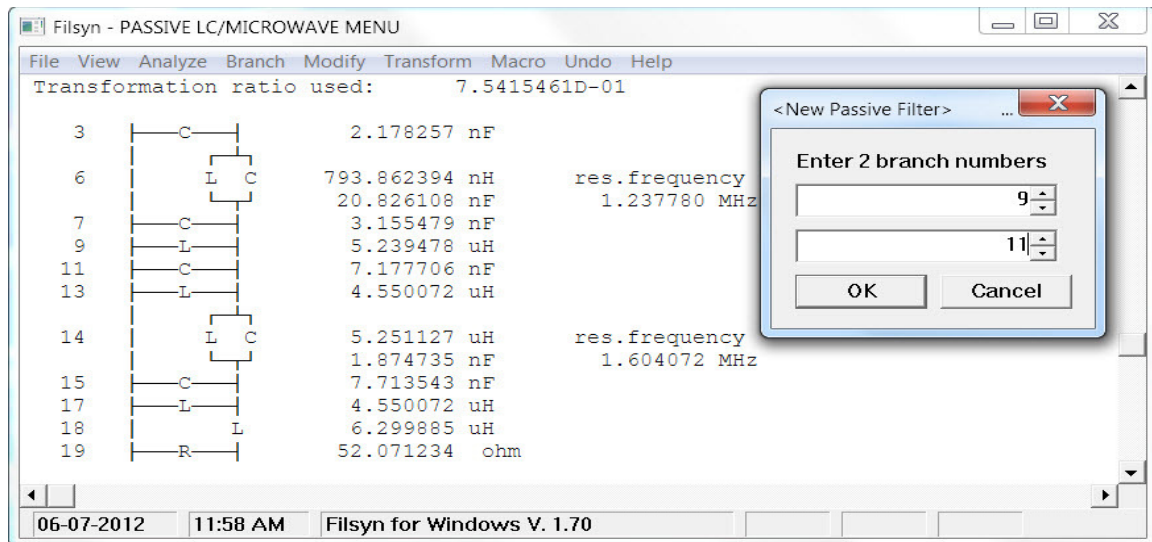
Let us illustrate this with a simple example. In the circuit below, we wish to get rid of inductor L_5 by performing a Norton transformation on branches L_5 and resonant branch 6. After calling menu item **Modify->Norton->Simple**, we get the data input screen below where we entered branch numbers 5 and 6 and nothing else:



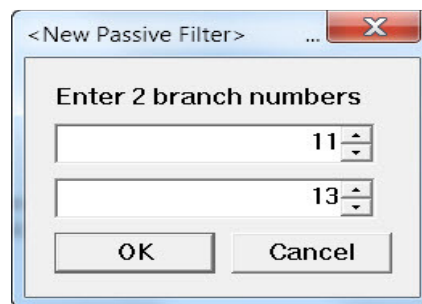
Clicking on the **OK** button, we get the prompt:



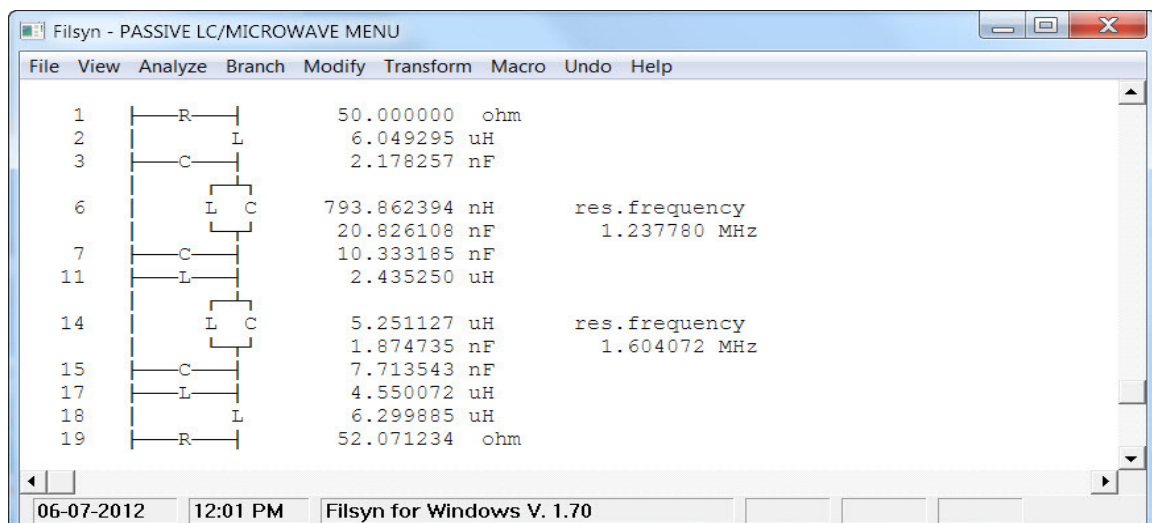
Selecting **Yes** we get the results:



We now have some branches, which should be combined. To facilitate that, we first call the menu **Branch->Interchange** and when we get the above data entry screen, we enter 9 and 11, then click **OK**. Next we select the menu item **Branch->Combine** and enter 11 and 13 into the resulting entry screen:

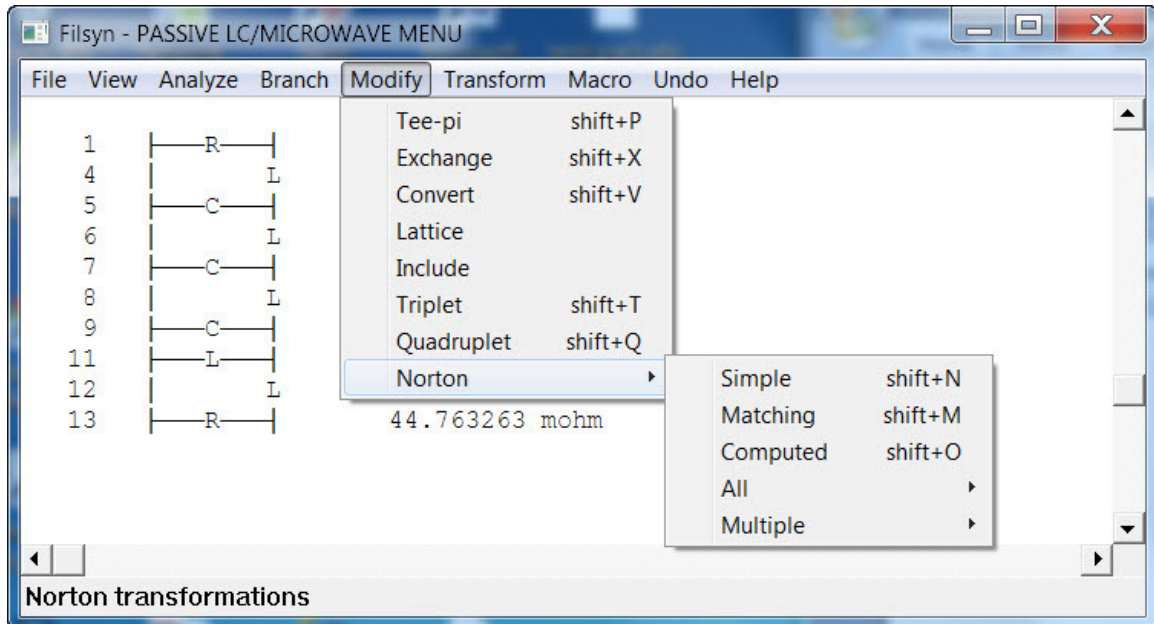


Clicking on **OK** will combine the two inductors L_{11} and L_{13} into a single one. After combining inductors C_{11} and C_{13} in the same way, we get the result we wanted:

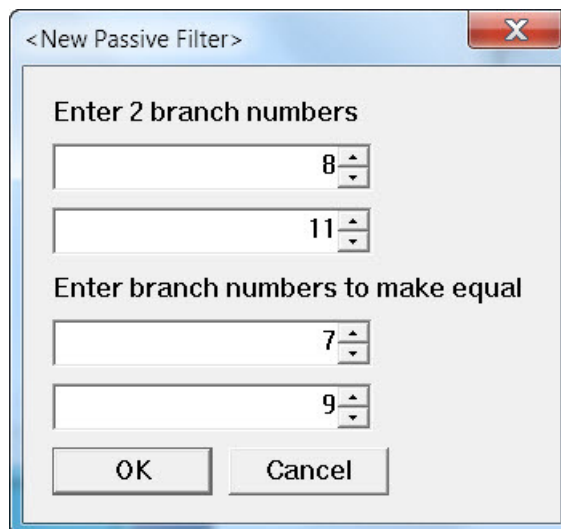


b) **Matching** is a case where we specify two more branches and we tell the program to make those two equal.

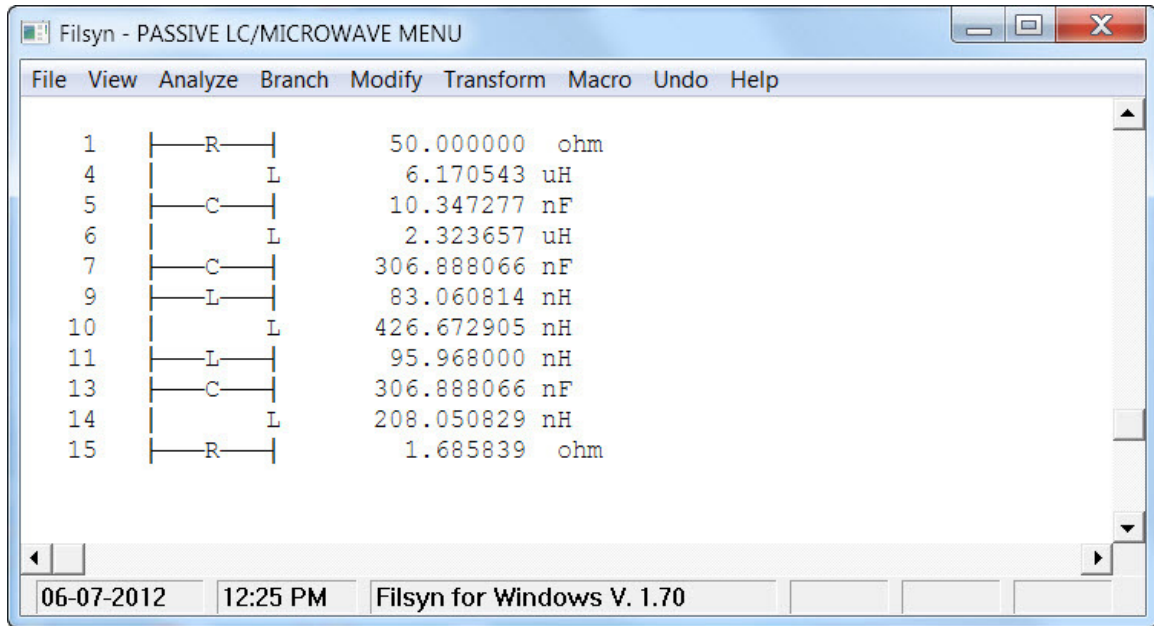
Consider the following starting circuit:



The first step is to make C_7 and C_9 equal, using L_8 and L_{11} as transforming branches. Using the menu selection indicated above, we get the data input screen where we enter the proper numbers:



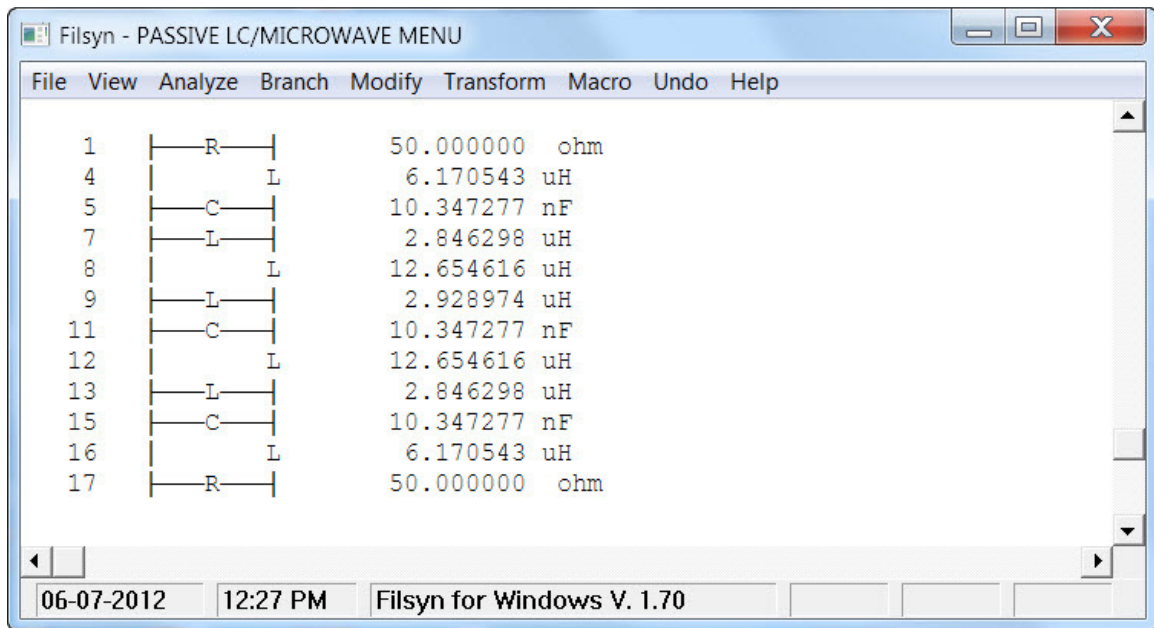
Clicking on the **OK** tab, we get the circuit, where the two capacitors are indeed identical:



The screenshot shows the Filsyn - PASSIVE LC/MICROWAVE MENU window. The menu bar includes File, View, Analyze, Branch, Modify, Transform, Macro, Undo, and Help. The main window displays a list of 15 components, each with a schematic symbol, a value, and a unit. The components are arranged in a vertical list, with their schematic symbols shown to the left of their values. The values and units are: 1: 50.000000 ohm, 4: 6.170543 uH, 5: 10.347277 nF, 6: 2.323657 uH, 7: 306.888066 nF, 9: 83.060814 nH, 10: 426.672905 nH, 11: 95.968000 nH, 13: 306.888066 nF, 14: 208.050829 nH, and 15: 1.685839 ohm. The status bar at the bottom shows the date 06-07-2012, time 12:25 PM, and version Filsyn for Windows V. 1.70.

Component	Value	Unit
1	50.000000	ohm
4	6.170543	uH
5	10.347277	nF
6	2.323657	uH
7	306.888066	nF
9	83.060814	nH
10	426.672905	nH
11	95.968000	nH
13	306.888066	nF
14	208.050829	nH
15	1.685839	ohm

After repeating the process to make C_5 and C_7 equal too, we have all shunt capacitors as well as the terminations equal and have a structurally symmetrical circuit:



The screenshot shows the Filsyn - PASSIVE LC/MICROWAVE MENU window. The menu bar includes File, View, Analyze, Branch, Modify, Transform, Macro, Undo, and Help. The main window displays a list of 17 components, each with a schematic symbol, a value, and a unit. The components are arranged in a vertical list, with their schematic symbols shown to the left of their values. The values and units are: 1: 50.000000 ohm, 4: 6.170543 uH, 5: 10.347277 nF, 7: 2.846298 uH, 8: 12.654616 uH, 9: 2.928974 uH, 11: 10.347277 nF, 12: 12.654616 uH, 13: 2.846298 uH, 15: 10.347277 nF, 16: 6.170543 uH, and 17: 50.000000 ohm. The status bar at the bottom shows the date 06-07-2012, time 12:27 PM, and version Filsyn for Windows V. 1.70.

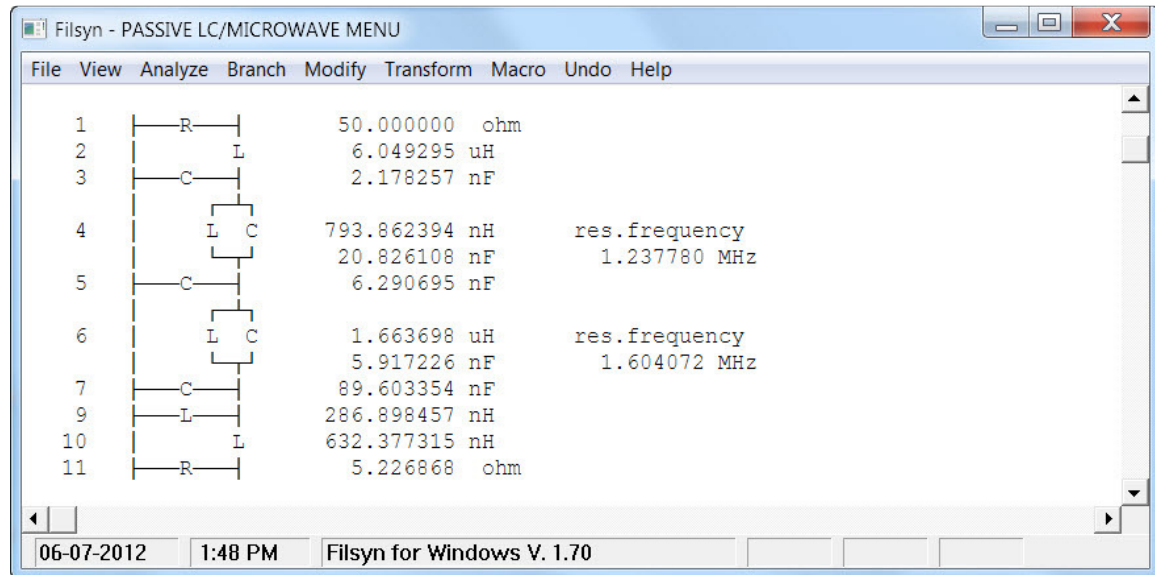
Component	Value	Unit
1	50.000000	ohm
4	6.170543	uH
5	10.347277	nF
7	2.846298	uH
8	12.654616	uH
9	2.928974	uH
11	10.347277	nF
12	12.654616	uH
13	2.846298	uH
15	10.347277	nF
16	6.170543	uH
17	50.000000	ohm

c) **Computed** is a version related to the next item on the list. When we compute a number of transformation values using that submenu item, then we can come back to this one and tell it which two branches and which transformation value to use. This will be demonstrated presently.

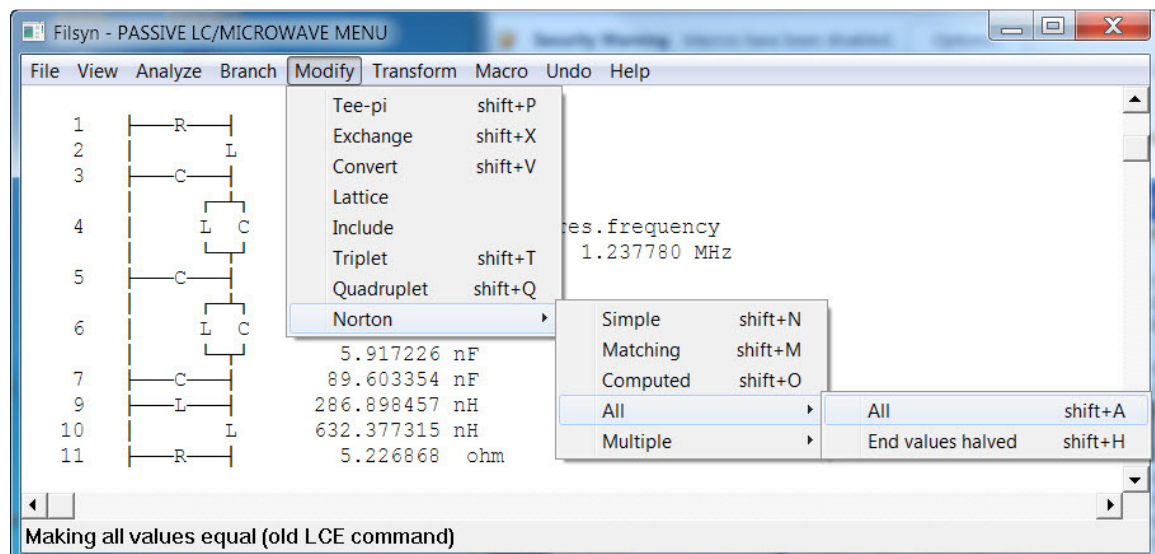
d) **All** does not actually do any transformation. That is done by the **Computed** item above. What it does is to calculate a number of transformation values, which, if used in sequence by **Computed**, will make a number of shunt inductors or capacitors equal. There are two

versions of this, the first makes all values equal, the other makes the end values equal to half the values of the others. This last combination is useful for the design of edge-coupled microwave filters.

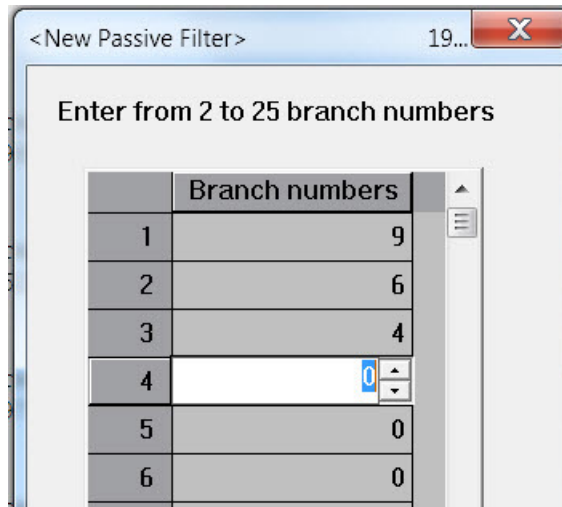
To illustrate this as well as the **Computed** item above, consider the starting circuit:



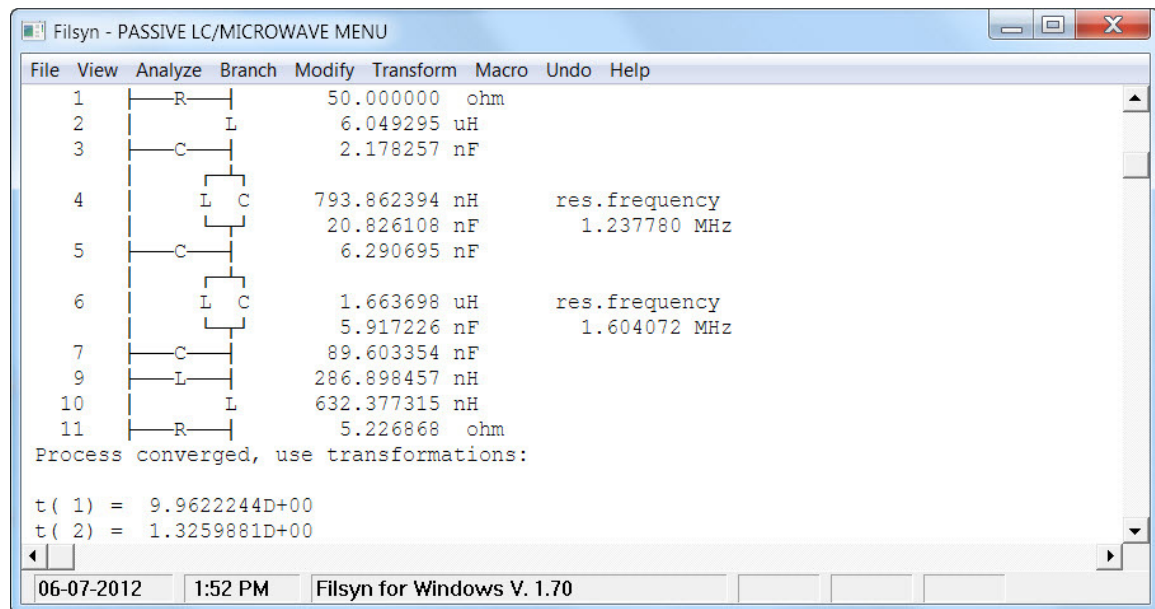
In this circuit, we would like to perform a couple of Norton transformation steps, first using branches 6 and 9 and then using the resulting inductor parallel with C_5 and branch 4. We want the resulting shunt inductors all have the same value. If we write down the equations for the necessary transformation ratios, we get a set of nonlinear equations. The menu item selections shown below will actually solve these equations and print the transformation ratios:



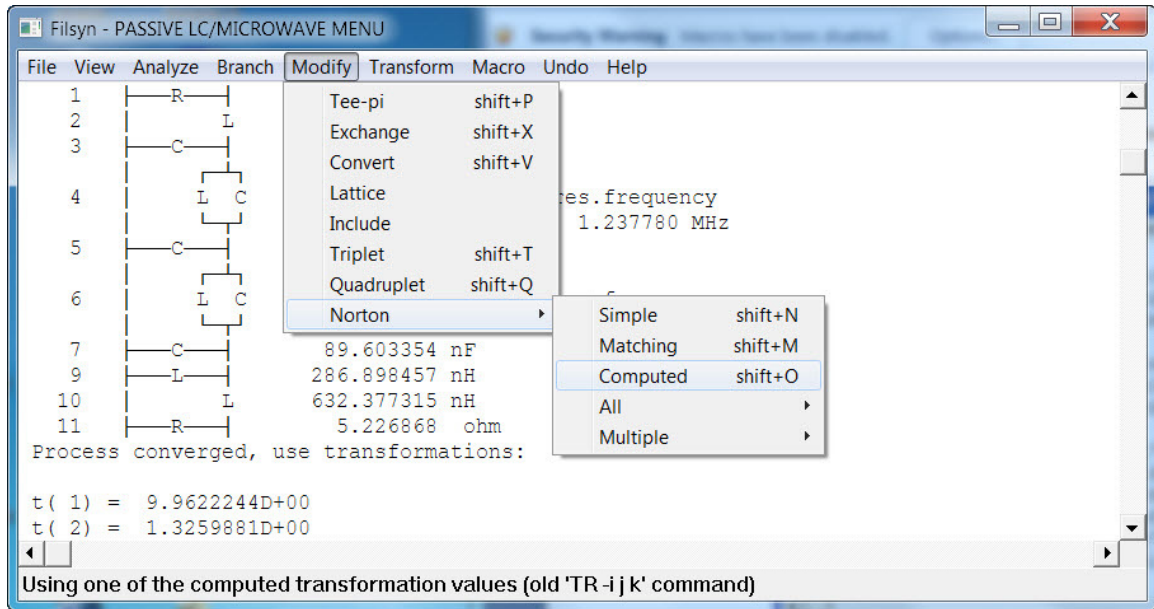
But first, we must however, enter the branches involved. The first branch must be a shunt branch at the end of the structure; here it is either C_7 or L_9 . We need to select L_9 since we wish to make all inductors the same. That is followed by all subsequent series branches in sequence:



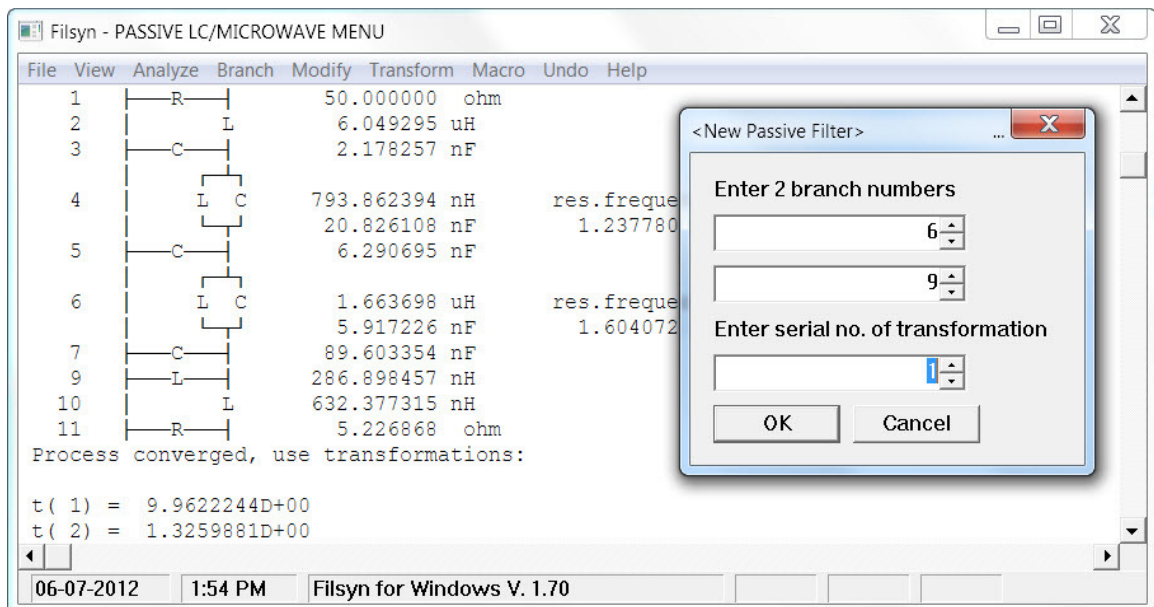
The sequence of numbers is important here, but the direction is not, the sequence 4, 6, 9 would work equally well. Clicking on **OK** will yield (notice the two numbers at the bottom):



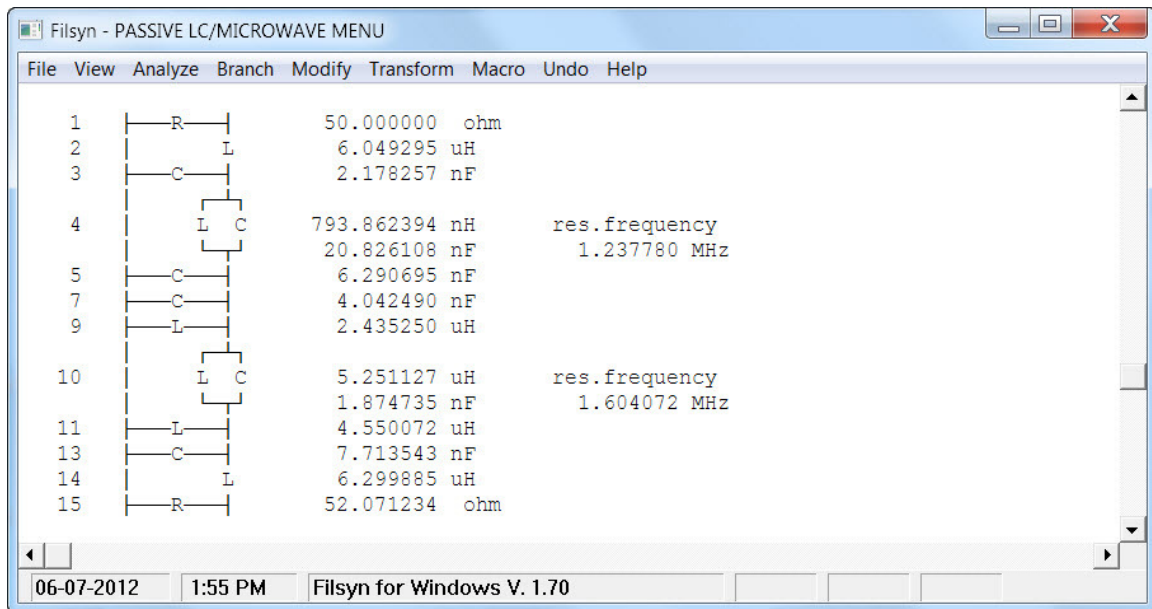
Now we could enter these values manually using the **Modify->Norton->Simple** menu item, but it is much simpler to use instead the **Modify->Norton ->Computed** option and enter the data as follows. First we select the menu item:



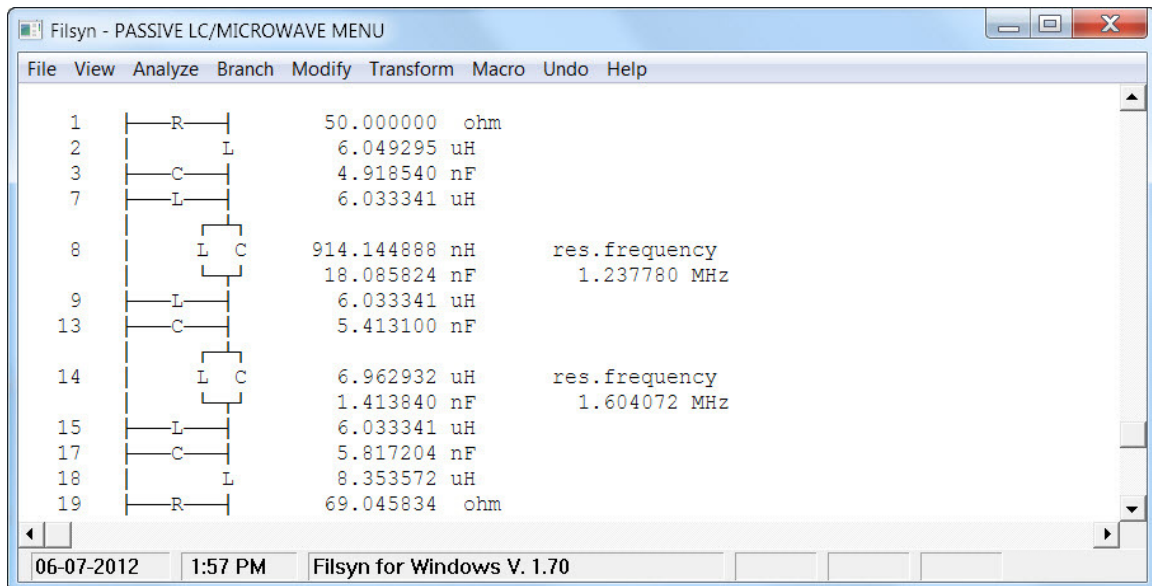
Next we enter the first two branch numbers needed for the transformation and the serial number of the transformation:



The resulting circuit now is as shown:



Combining the two capacitors C_5 and C_7 and repeating the process with branches 9 (the new inductor) and 4, as shown above and again combining some parallel capacitors, we get the final result:



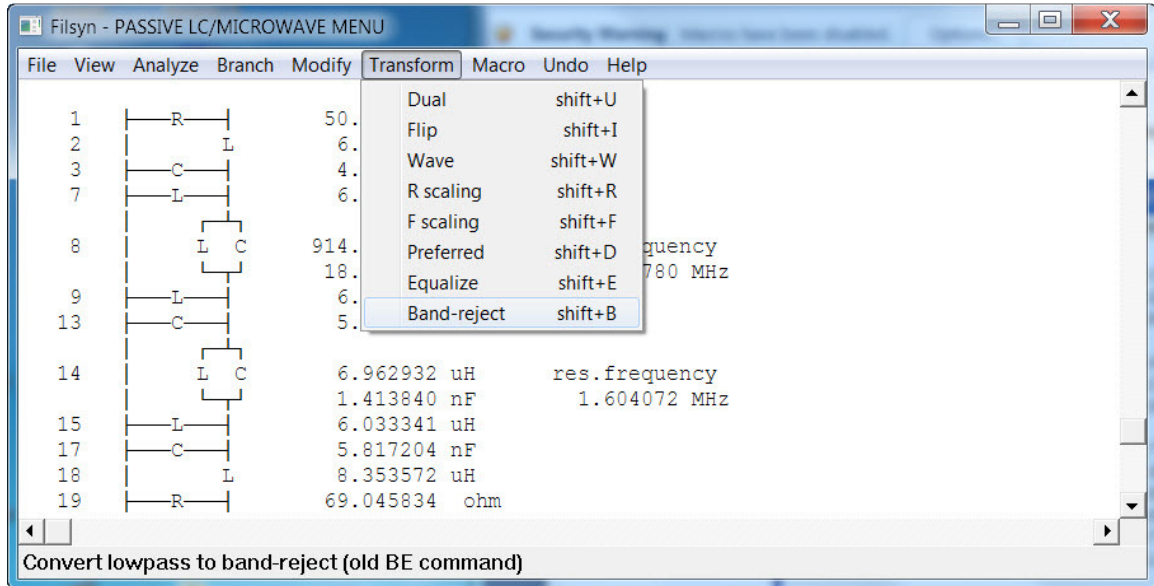
As one can see, all three shunt inductors have the same value, as desired.

e) **Multiple** is our last submenu option and this is used to apply a generalized version of the Norton transformation to make all shunt inductors or capacitors in a triplet or a quadruplet section equal. Instead of making these values all equal, we may specify two or three transformation ratios, but this option is not very useful.

The next set of main menu items is simpler, they all involve operations where the whole circuit is changed:

4.6 The 'Transform' menu options

1. **Dual** simply converts the circuit into its dual form. If the filter contains bridged-T and/or twin-T sections, the dual is useless and is not available.
2. **Flip** turns the circuit end-for-end.
3. **Wave** converts a microwave filter into a *wave-digital* form. See the chapter on digital filter analysis for details.
4. **R scaling** performs an impedance scaling operation.
5. **F scaling** performs a frequency scaling operation.
6. **Preferred** replaces all capacitors by the nearest value from a set of preferred values.
7. **Equalize** invokes the delay equalization process.
8. **Band-reject** converts a lowpass filter to a band-reject filter. This option offers the only way to design band reject (band elimination) filters. The only additional data needed are the upper end of the lower passband and the lower end of the upper one.



If we have a set of specifications for a band reject filter, we must design the proper reference lowpass filter first. Assuming the requirements are of the form shown in Fig. 3 below, we must first calculate the values:

$$B_1 = F_{AS} * (F_B - F_A) / (F_A * F_B - F_{AS} * F_{AS})$$

$$B_2 = F_{BS} * (F_B - F_A) / (F_{BS} * F_{BS} - F_A * F_B)$$

Next we specify a lowpass filter with an arbitrary f_p passband edge frequency and a stopband edge that is the product of this passband edge and the smaller of B_1 and B_2 . The loss requirements are the same as those of the band reject filter.

Note that these equations are valid only for analog filters (passive LC, active RC), while for digital and microwave filters they become quite a bit more complex. However, the **filscript.exe** menu script file offers the design of band reject filters in the **Butterworth**, **Chebyshev** and **Elliptic** menu options and use the proper expressions for both analog and digital filters.

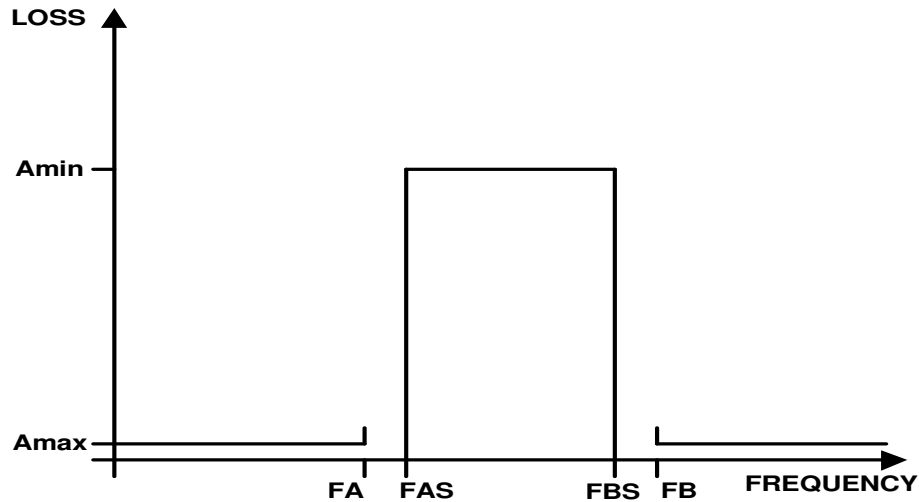
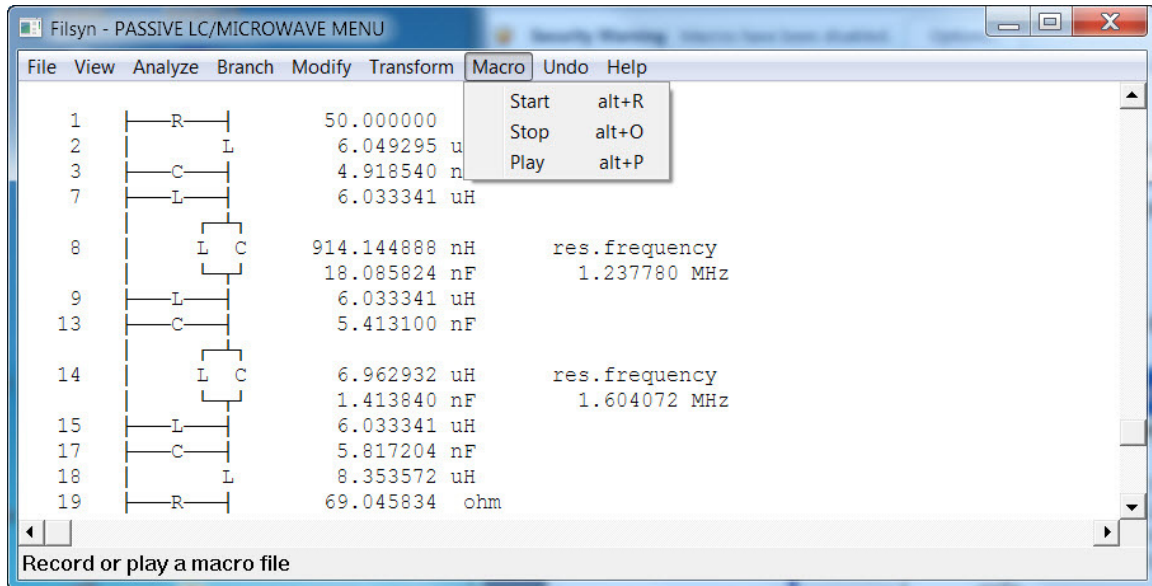


Fig. 3 Band reject filter requirements

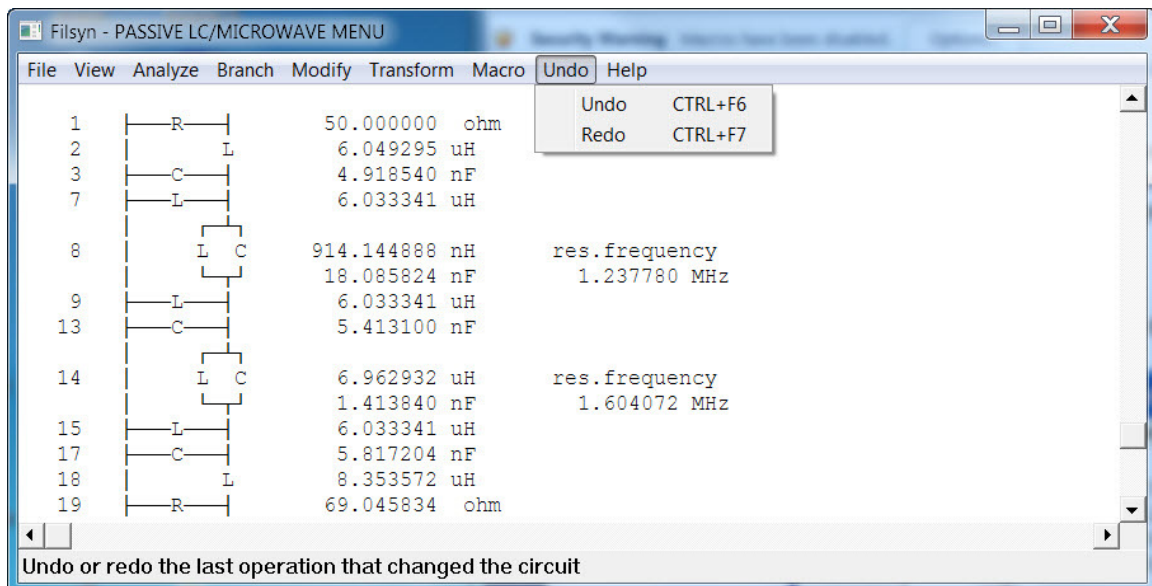
4.7 The 'Macro' menu options

Macro records in a text file all the actions that modify the circuit. This file can then be replayed and the actions performed again automatically. These files can then be edited and used also as script files. **Start** will start the recording, **Stop** will stop it. Separate segments will be concatenated in the file. **Play** will of course rerun the operations.

Note however than these **macro** operations require the presence of the *WinBach.exe* file and we do NOT supply a copy of this executable.



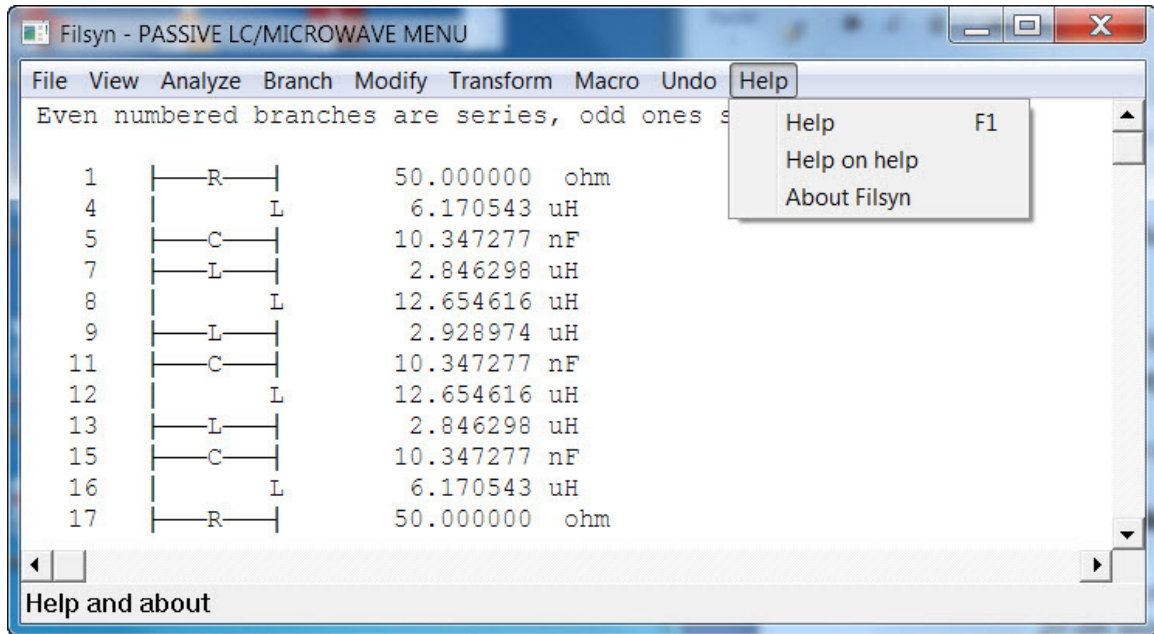
4.8 The 'Undo' menu option



Undo has two functions, undo up to ten of the last operations that changed the current filter, or redo any of the ones we have undone, in reverse sequence. If we undid more than one, then all but the first undo can be redone. Operations that do not change the circuit are all ignored here.

4.9 The 'Help' menu options

Finally the **Help** menu item contains the usual items:



Some of these menu items have already been demonstrated, others will be deferred to other chapters, still others need no further explanations.

Consider a few additional examples to illustrate some of these menu items.

4.10 Additional examples

The first example is a band reject filter with a stopband from 2.5 KHz to 4.5 KHz and a 20 dB minimum loss, while the passbands are below 2 KHz and above 6 KHz with a 0.1 dB Passband ripple. First we must compute the B_1 and B_2 values shown earlier, yielding:

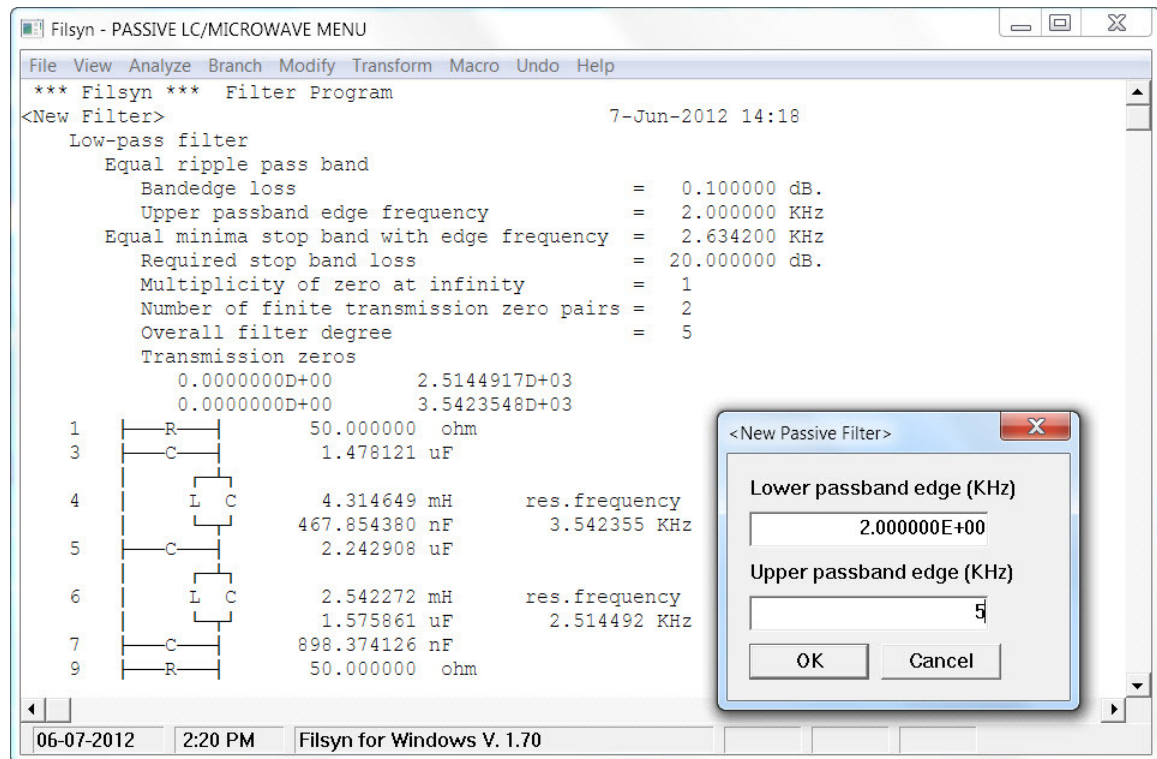
$$B_1 = 2.0$$

$$B_2 = 1.3171$$

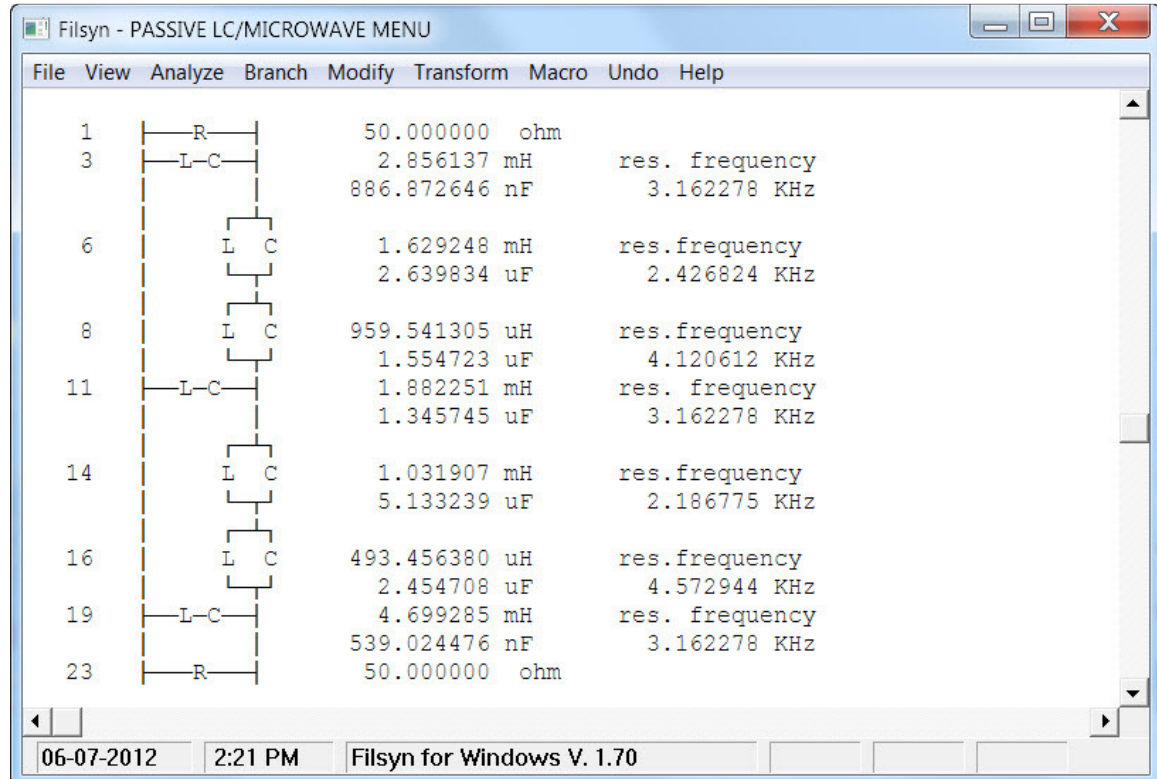
Selecting arbitrarily 2 KHz as the passband edge of the reference lowpass, the stopband edge will be B_2 times that (the lower of the two values above times two yielding 2.6342), hence the data entry screen shows the following:

Selecting the default computer selected structure, we obtain:

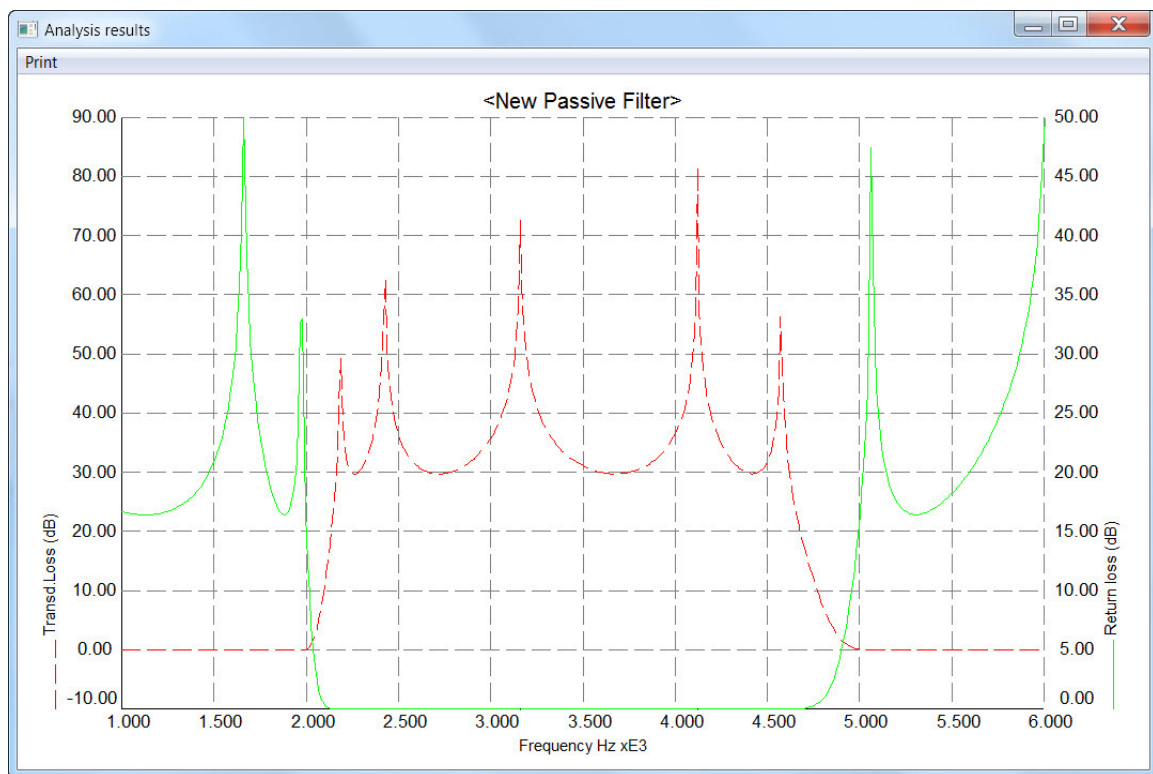
Using the **Transform->Band-reject** menu option, as shown above, we can next enter the passband edge frequencies:



The resulting band-reject filter is shown immediately:



The analysis results show a perfect performance:

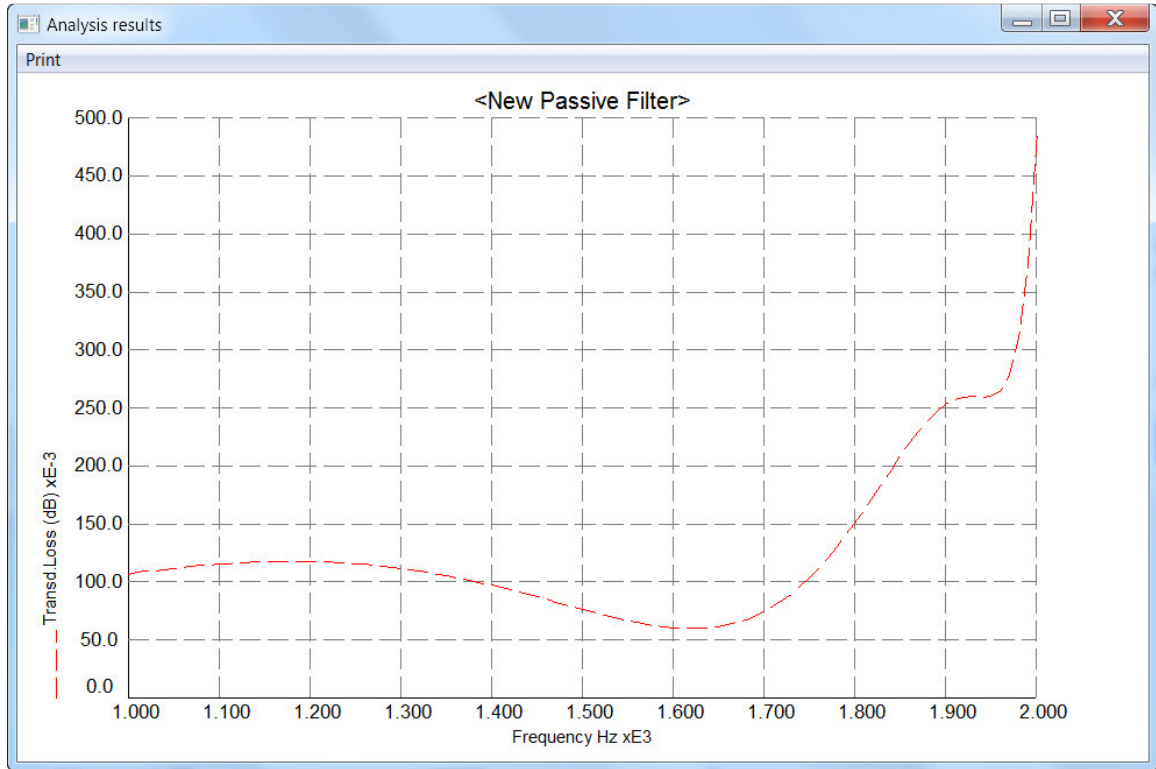


If we wish to take into account the fact that the LC elements usually have a finite Q value, we can repeat the analysis:

The figure shows a dialog box titled "<New Passive Filter>". It contains the following fields and buttons:

- Start and end frequencies (KHz)**: Two input fields with values $1.000000\text{E}+00$ and $6.000000\text{E}+00$.
- Increment (KHz)**: An input field with value 0.00000 .
- or no. of frequency points**: A spin box with value 501 .
- Inductor and capacitor Q's**: Two input fields with values 300 and 0.00000000 .
- Buttons**: OK, Cancel, and Next.

The next prompt, not shown, ask us if these values are constant (frequency independent), linearly varying with frequency, or abort the operation. If we select varying linearly with frequency, then the values given are those at the (upper) passband edge. The overall plot is indistinguishable from the one above, but if we zoom in on the upper edge of the lower passband, we can see the effect of the lossy components:



The effect is now visible, although still quite small.

Our next example is a simple microwave lowpass, with the following specifications: Quarter-wave frequency: 10 GHz, passband edge: 5 GHz with 0.2 dB ripple and three zeros at the quarter-wave frequency and two more contributing unit elements:

<New Filter>

Filter kind

- ☐ LC
- ☒ Microwave
- ☐ Active RC
- ☐ IIR Digital

Quarter-wave/Sampling freq (GHz)
10.000000E+00

Filter type

- ☒ Low pass
- ☐ High pass
- ☐ Band pass

Lower passband freq (GHz)
0.00000

Upper passband freq (GHz)
5.000000E+00

Passband type

- ☐ Max. flat
- ☒ Equal ripple
- ☐ Functional
- ☐ Sloping

Band-edge loss/return loss (dB)
0.20000000

Loss Slope (dB/oct)
6.00000000

Flat loss (dB)
0.00000000

Function Type

- ☒ E
- ☐ F

Multiplier
0.00000000

Stopband

- ☐ Monotonic
- ☐ Equal min
- ☐ Placer
- ☒ Specified

Lower stopband freq (GHz)
0.00000

Loss (dB)
50.00000000

Upper stopband freq (GHz)
0.00000

Loss (dB)
50.00000000

of zeros at zero
0

of zeros at FQ or FS/2
3

of unit elements
2

Parametric

- ☒ Conventional
- ☐ Parametric
- ☐ Matching

R1
50.000000E+00

R2
50.000000E+00

ZS parameter
-1

Q for predistortion
0.00000000

Predistortion

- ☒ Passband only
- ☐ Pass/stop bands

Shifted bandpass trans.
Center frequency (GHz)
0.00000

☐ Odd parametric

☐ Save design data

OK Cancel

After the computer-controlled synthesis we have:

Filsyn - PASSIVE LC/MICROWAVE MENU

File View Analyze Branch Modify Transform Macro Undo Help

*** Filsyn ***

<New Filter>

Low-pass filter

Equal ripple

Band-edge loss/return loss (dB) = 0.200000 dB.

Max. flatness = 1.538553

Upper passband frequency (GHz) = 5.000000 GHz

Quarter-wave frequency = 10.000000 GHz

Specified stop band

Multiplicity of zero at Quarter-wave fr. = 3

Number of unit elements = 2

Number of finite transmission zero pairs = 0

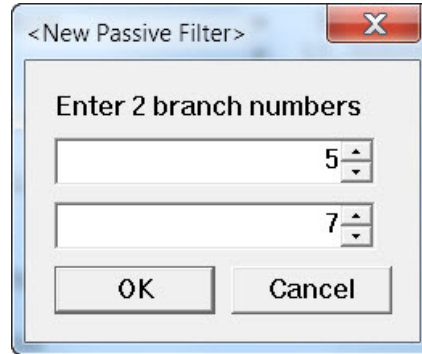
Overall filter degree = 5

**** All values are impedances ****

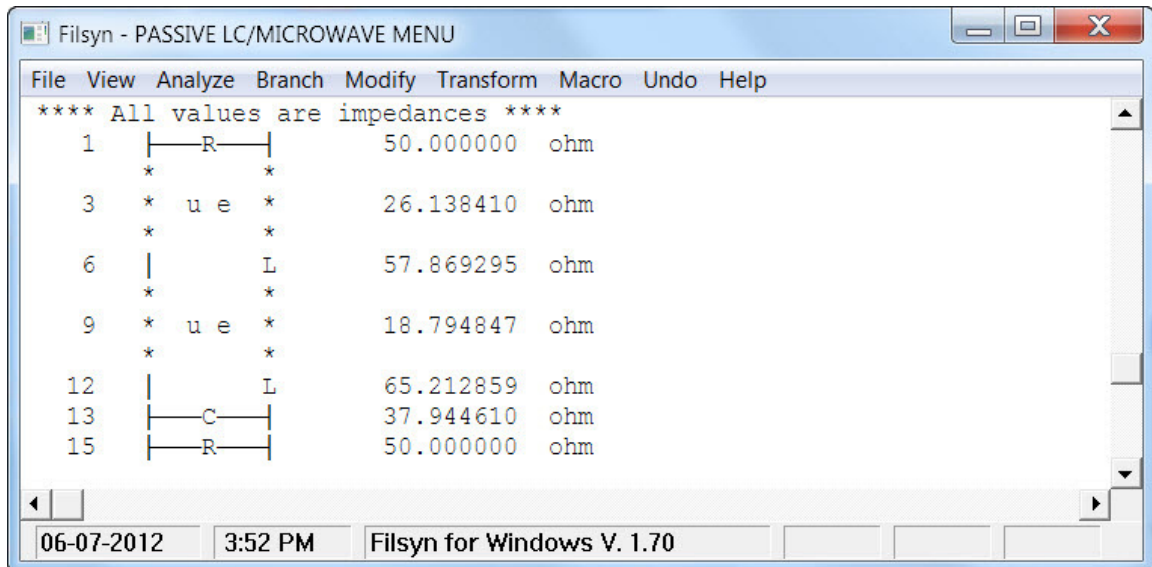
1	—R—	50.000000 ohm
3	* u e *	26.138410 ohm
5	* u e *	76.664142 ohm
7	—C—	24.899055 ohm
8	—L—	65.212859 ohm
9	—C—	37.944610 ohm
11	—R—	50.000000 ohm

Interchange two branches (old IB command)

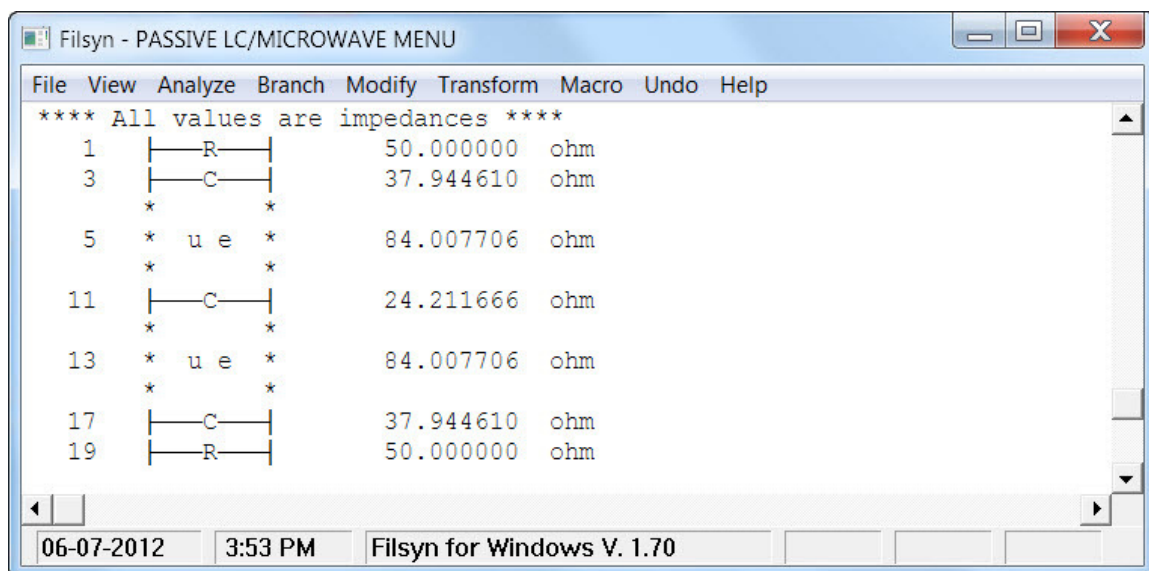
The computer-generated structure is not quite what we need, hence we have to rearrange the branches. First we interchange branches 5 and 7 as shown above and below. As we mentioned before, this is actually the use of one of the Kuroda's identities and after this prompt that specifies the branches:



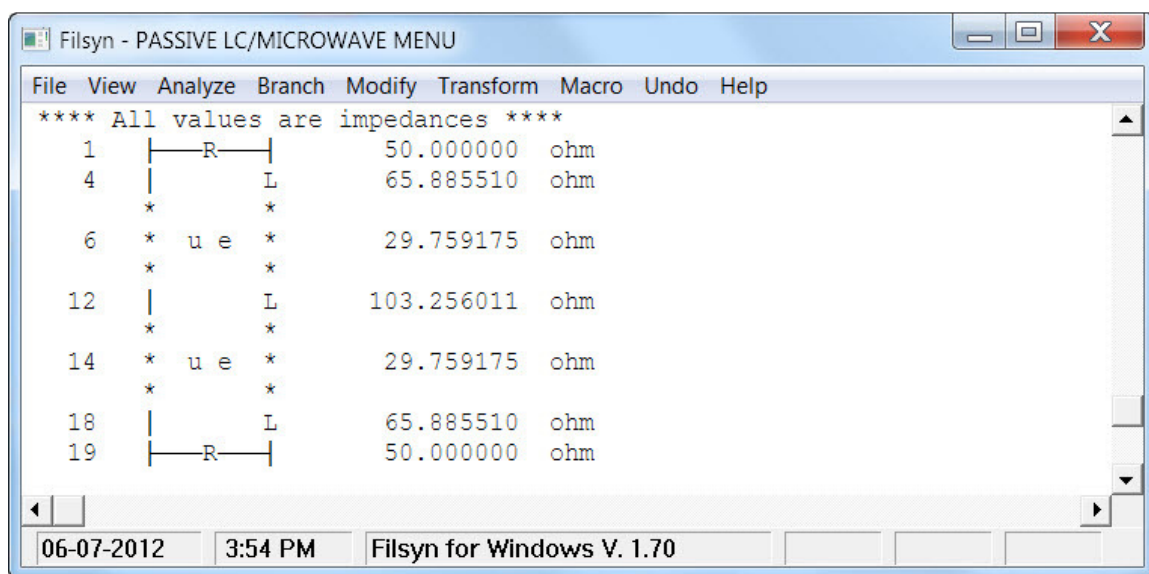
we get:



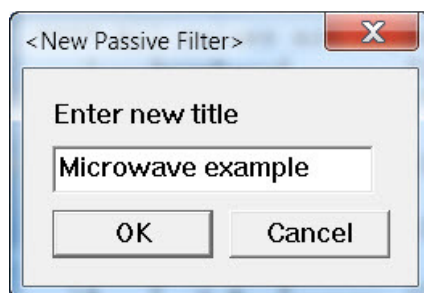
Interchanging branches 9 and 12 and then branches 3 and 6, that is using two more Kuroda's identities yields the circuit we need:



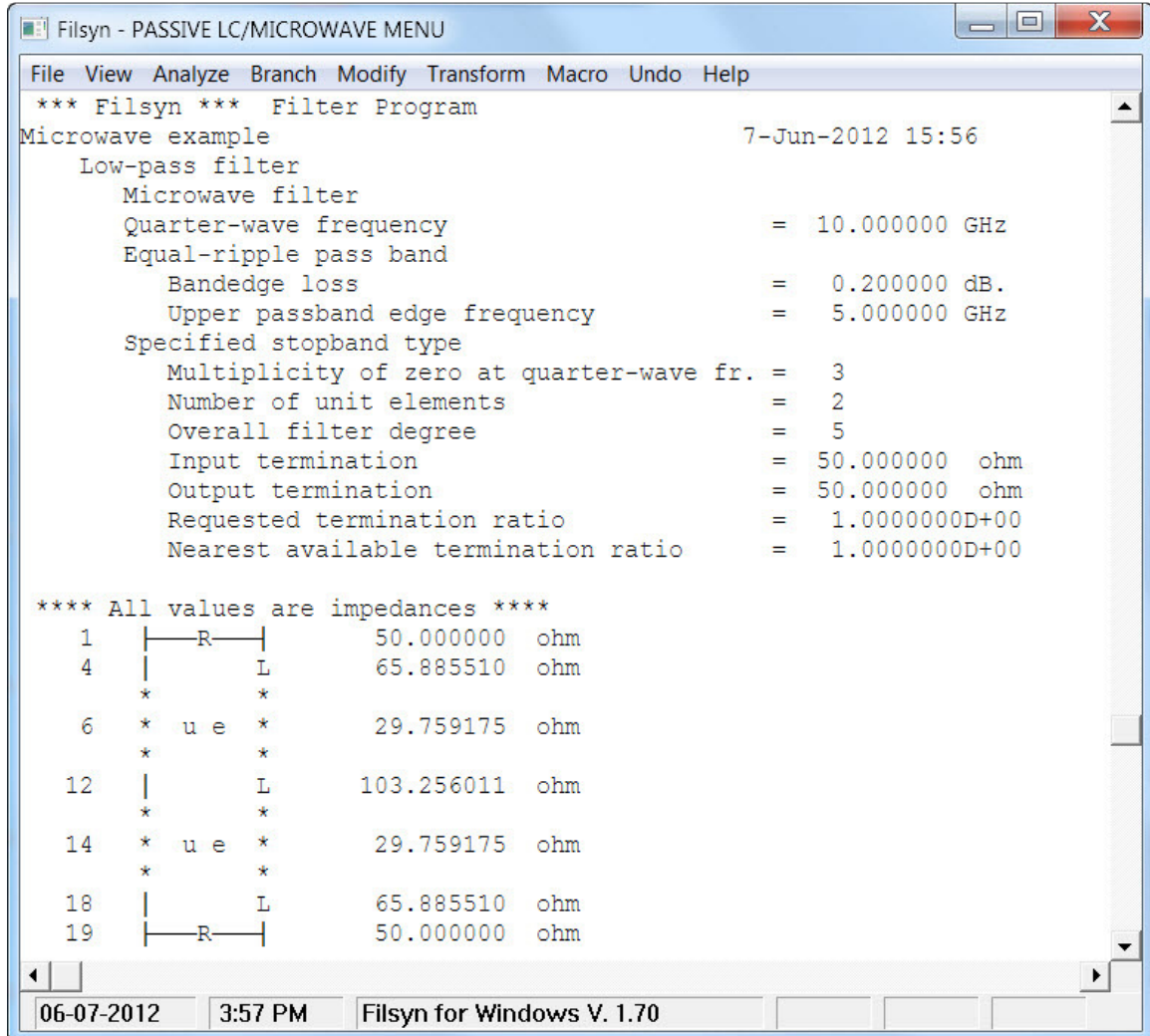
Using the **Transform->Dual** menu item yields the dual circuit, if we need that:



If we wish to change the title of the design, we can use the **View->Change title** menu option, leading to the entry screen:



and using the **View->Show** shows us the new title:



Selecting the **File->Text file** menu item generates a text file, which we can display in any text editor, or even in the Main menu of **Filsyn** by using the **File->Open** menu item and select the file we just wrote:

```

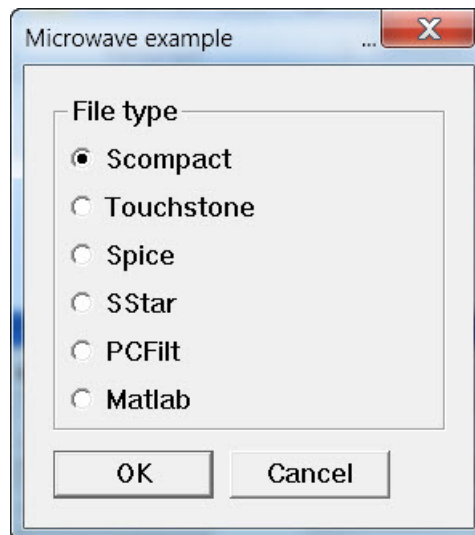
*** Filsyn ***  Filter Program
Microwave example                                     7-Jun-2012 16:06
  Low-pass filter
    Microwave filter
      Quarter-wave frequency                        = 10.000000 GHz
      Equal-ripple pass band
        Bandedge loss                             = 0.200000 dB.
        Upper passband edge frequency              = 5.000000 GHz
      Specified stopband type
        Multiplicity of zero at quarter-wave fr.  = 3
        Number of unit elements                   = 2
        Overall filter degree                     = 5
        Input termination                         = 50.000000 ohm
        Output termination                       = 50.000000 ohm
        Requested termination ratio                = 1.0000000D+00
        Nearest available termination ratio        = 1.0000000D+00

  1  |-----R-----|      50.000000 ohm
  4  |                   L      65.885510 ohm
    *                   *
  6  *   u e   *      29.759175 ohm
    *                   *
 12  |                   L      103.256011 ohm
    *                   *
 14  *   u e   *      29.759175 ohm
    *                   *
 18  |                   L      65.885510 ohm
 19  |-----R-----|      50.000000 ohm

```

Note that this option only displays the file, it cannot be edited or printed.

If we wish to write a transfer file to generate a netlist readable by other programs, we can use the **File->Transfer** menu option that brings up the prompt:



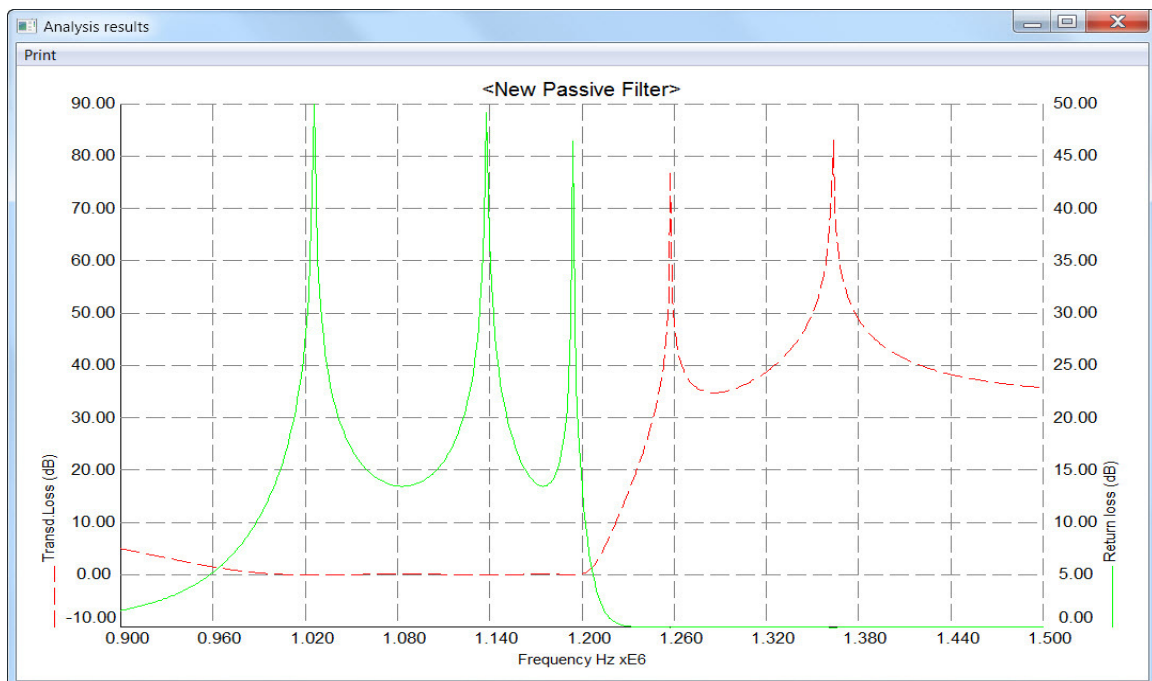
and the resulting *SuperCompact* file is shown here:

```
* Microwave example
f1 : 1.000000E+10
el : 90.00
blk
* res 1 0 r= 5.000000E+01
sst 2 1 z = 6.5885513E+01 e=e1 f=f1
trl 3 2 z = 2.9759174E+01 e=e1 f=f1
sst 4 3 z = 1.0325601E+02 e=e1 f=f1
trl 5 4 z = 2.9759174E+01 e=e1 f=f1
sst 6 5 z = 6.5885513E+01 e=e1 f=f1
* res 6 0 r= 5.000000E+01
a: 2por 1 6
end
freq
* put frequencies here
step 1.250000E+08 6.249999E+09 1.224999E+08
end
*
out
pri a r1 = 5.000000E+01 r2 = 5.000000E+01 s
end
```

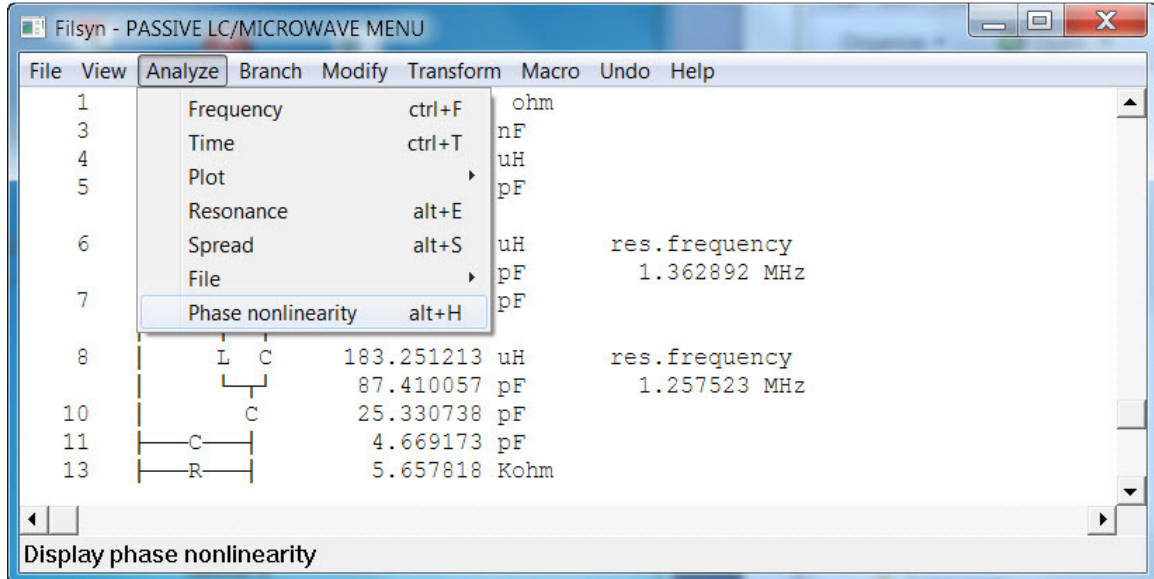
7-Jun-2012 16:00

The explanation of this file is of course, left to the *SuperCompact* manual. *Spice*, *SuperStar* and *Matlab* files are not available for microwave filters.

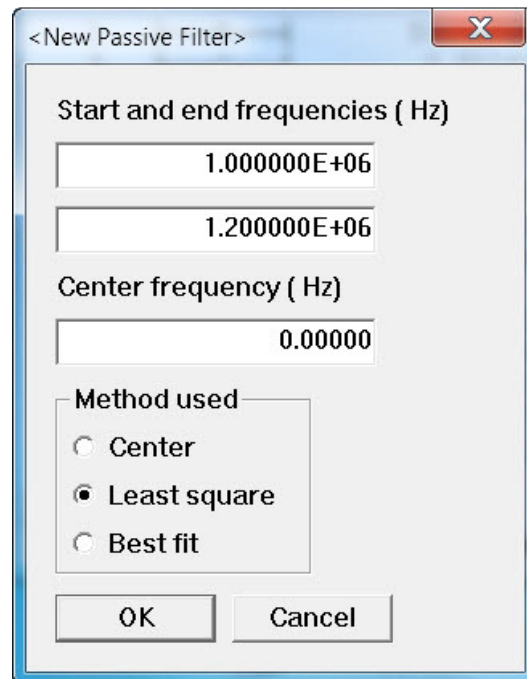
Recently we have added another submenu item to the **Analyze** menu, called **Phase nonlinearity**. As an example, consider the following bandpass filter with a passband from 1 MHz to 1.2 MHz with a performance shown:



The new submenu item is:



Clicking on this item we get the selection menu:

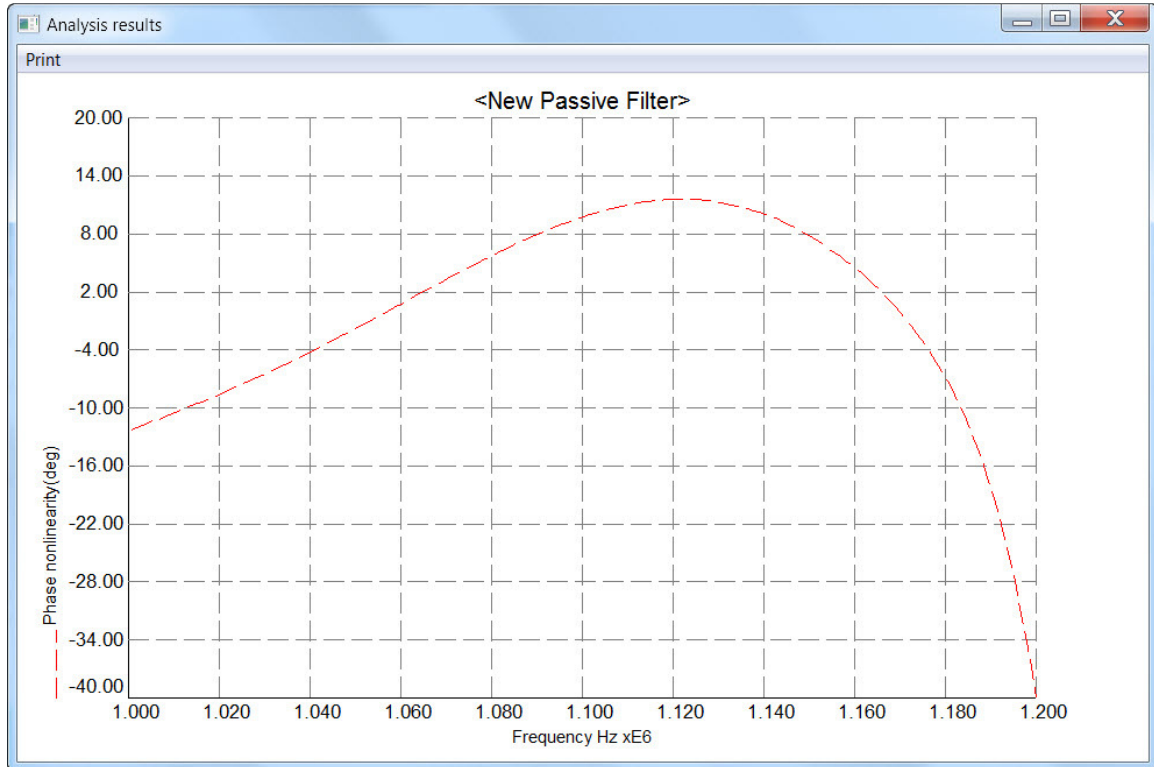


Naturally, the filter must have been analyzed prior to using this submenu, so that the phase has been computed. Here we specify the frequency band of interest and select one of three options for the method used in computing the nonlinearity.

The nonlinearity is defined as the phase deviation from a linear phase, where the linear phase can be any of the following:

- osculating linear phase at the center frequency (if undefined, as above, it is assumed to be half way between the two end frequencies)
- the linear phase that has the least mean square error
- the linear phase that goes through the two end points.

Here we have selected the least square error linear phase and the results are shown next:



No need to tabulate the results, the graphical display together with our ability to read off specific values if needed, is quite adequate.

Notes:

a) In a π section where the two shunt elements Z_1 and Z_3 are unequal, and you wish to apply a Norton transformation to make them equal, the transformation value used should be:

$$t = \sqrt{(Z_3(Z_1 + Z_2))/(Z_1(Z_2 + Z_3))}$$

or its inverse. Of course, if Z_1 and Z_3 are already equal, this expression yields unity. Also note that Z_2 could be either a single element branch of the same type as the shunt ones, or it could be a parallel resonant circuit and the value used for Z_2 above is the value of the same type of element as the shunt branches. This expression is used in several of the script files, where we aim for all shunt L's to be equal.

b) In microwave filters if we have finite transmission zeros, these are implemented as parallel or series resonant circuits as they are in lumped LC filters. But in microwave filters they may be replaced by cascaded line segment as shown in the figure:

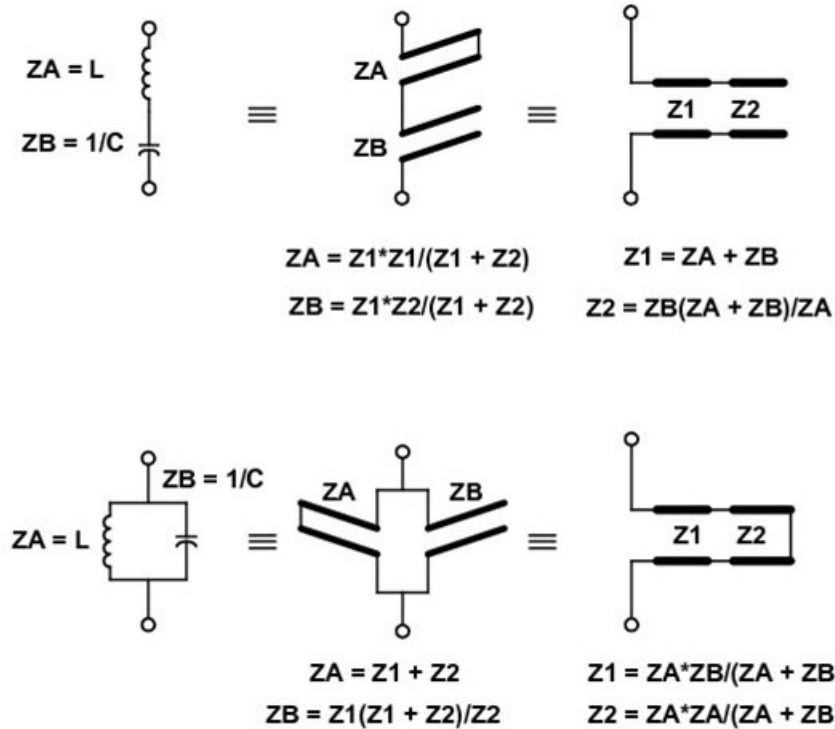


Fig. 4 Microwave equivalences

In our display of microwave circuits in addition to displaying the Z_A and Z_B values, we are also showing the Z_1 and Z_2 impedances for your information. These may be more convenient both in values as well as configurations. Finally similar equivalences are available for three and four element branches.

c) Another step a microwave filter designer may encounter is a combination of a unit element with an open series stub on one side, like the segment in the top left corner of Fig. 2. What we'd like to do is to split the stub (capacitor in our shorthand notation) into two and transfer one of the pieces to the other side of the unit element, using the proper Kuroda identity, such that the two stubs are identical.

At the present time this can be performed by a two step operation. First, we use the **Modify -> Norton -> All -> All** menu item and using the branch numbers of the unit element and the series open stub. This computes the impedance level of the open stub that is to be used in the transformation and splits it off from the stub that is present. Then we use the **Branch -> Interchange** menu to interchange this part of the stub with the unit element. In fact this performs the required Kuroda identity. The resulting circuit will have the required symmetrical structure.

Performing the same operation using a shunt shorted stub instead of the series open one can be done by taking the dual circuit first, performing the steps outlined above and finally taking the dual again.

5. BANDPASS IMPEDANCE MATCHING

It is well-known, that bandpass structures (as well as microwave highpasses containing unit elements, which are in fact bandpass filters) can accommodate unequal terminations without any special procedure. Occasionally, however, we need to match unequal terminations with a structure that looks like a lowpass filter. A lowpass-like structure can be designed to match two unequal resistors within a finite frequency band, which does *not* include zero frequency. This can be done in lumped or in microwave forms. We shall illustrate these designs here.

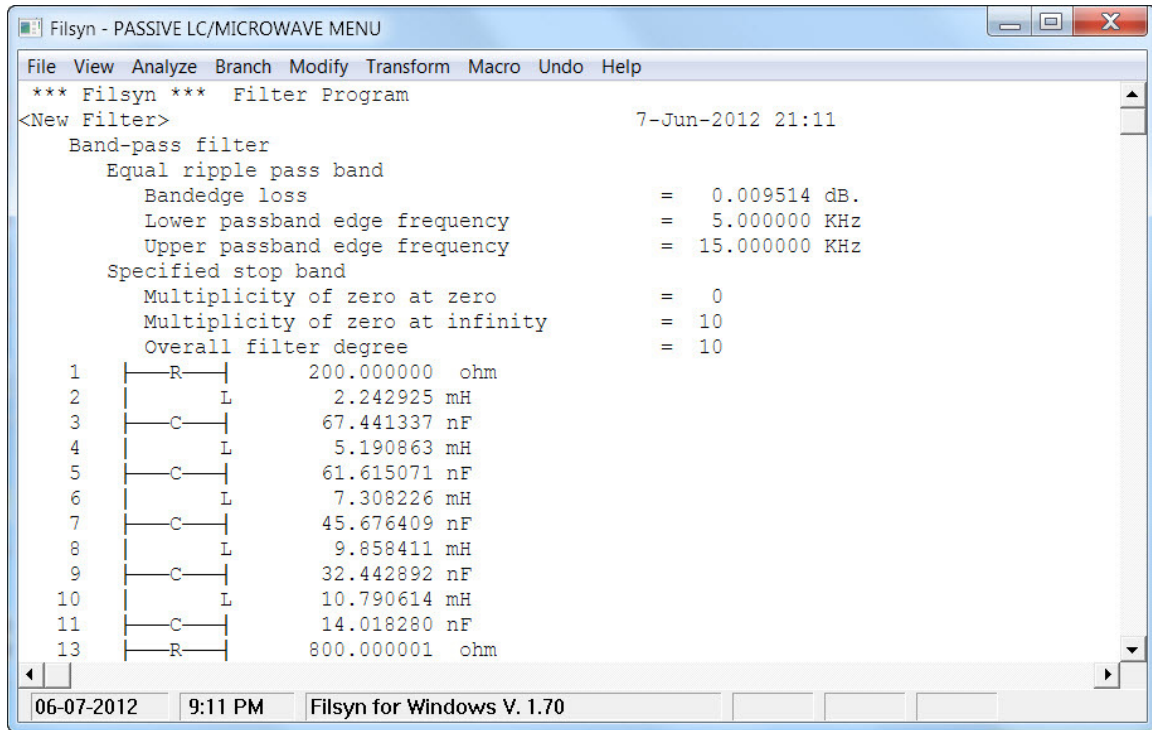
We start with the lumped LC case and request a bandpass that matches 50 ohm termination on one side to 200 ohm on the other over a band from 5 KHz to 15 KHz in an equal ripple manner with not more than 0.01 dB loss ripple. The input data screen is consequently as follows:

The screenshot shows the 'New Filter' dialog box with the following settings:

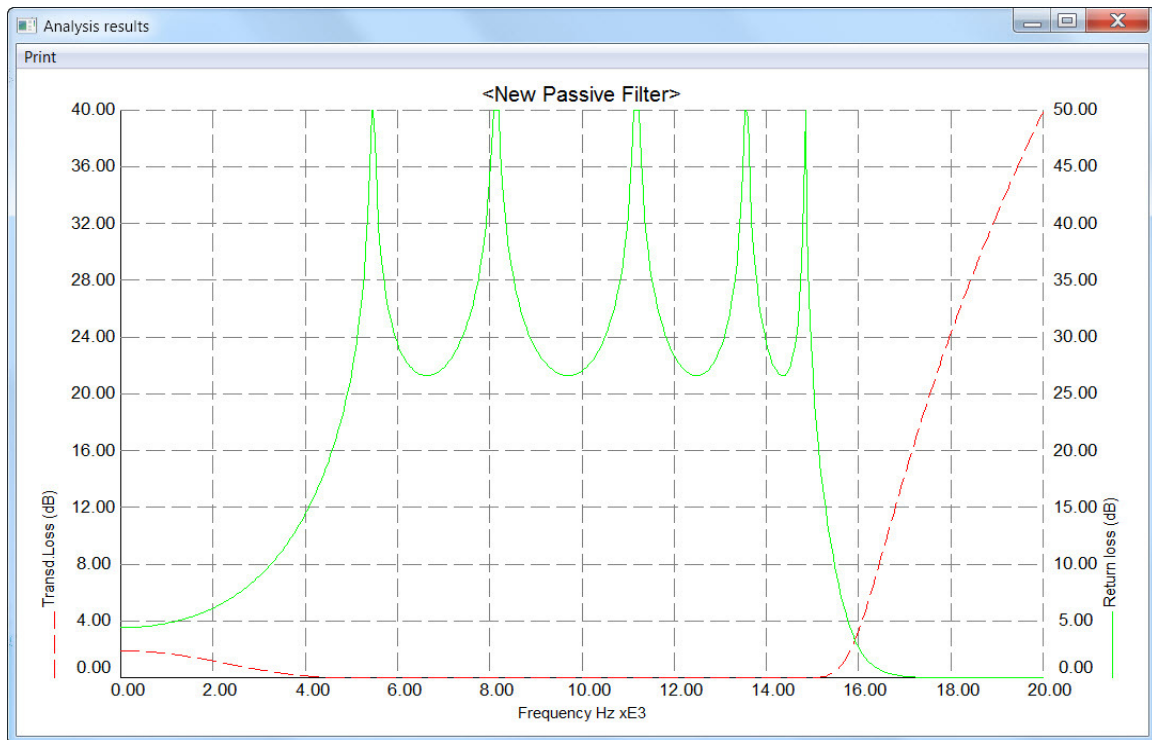
- Filter kind:** LC (selected)
- Filter type:** Band pass (selected)
- Quarter-wave/Sampling freq (Hz):** 0.00000
- Lower passband freq (Hz):** 5.000000E+03
- Upper passband freq (Hz):** 15.000000E+03
- Passband type:** Equal ripple (selected)
- Band-edge loss/return loss (dB):** 0.100000000E-01
- Loss Slope (dB/oct):** 6.00000000
- Flat loss (dB):** 0.00000000
- Function Type:** E (selected)
- Multiplier:** 0.00000000
- Stopband:** Specified (selected)
- Lower stopband freq (Hz):** 0.000000
- Loss (dB):** 50.00000000
- Upper stopband freq (Hz):** 0.000000
- Loss (dB):** 50.00000000
- ZS parameter:** -1
- Q for predistortion:** 0.00000000
- Predistortion:** Passband only (selected)
- Shifted bandpass trans. Center frequency (Hz):** 0.000000
- Parametric:** Matching (selected)
- R1:** 200.000000E+00
- R2:** 50.000000E+00
- Buttons:** OK, Cancel

Note that the stopband type is selected to be **Specified**, but this is immaterial since the other options are all inapplicable and the program ignores it. The critical selection is **Matching**. Also we know nothing about the degree of the filter, the program will compute it.

Clicking on the **OK** button and later selecting the computer configuration, we get the circuit shown below. The impedance level is off by a factor of 4 because the program is using the input impedance R_1 as the primary impedance level; this can be corrected easily using the **Transform -> R scaling** menu item.



The passband loss ripple is 0.0095 dB, satisfying the requirements and when we plot the computed response, we find that all other specifications are also met.



Considering now the microwave case, we have there the additional option of adding unit elements to the implementation. Hence we have now the following possibilities:

1. No unit elements. This essentially leads to a very similar solution, the only difference between the LC and the microwave cases will be in the nature of the components, L's and C's versus their distributed parameter equivalents.
2. A specified number of unit elements. This can be useful, if we wish to avoid the use of series shorted stubs. For instance, considering a very similar set of specifications to the LC design above, we can request 5 unit elements in addition to whatever else is needed to meet the specifications. The corresponding input data screen is:

<New Filter>

Filter kind

- ☐ LC
- ☒ Microwave
- ☐ Active RC
- ☐ IIR Digital

Quarter-wave/Sampling freq (Hz)

1.000000E+09

Filter type

- ☐ Low pass
- ☐ High pass
- ☒ Band pass

Lower passband freq (Hz)

5.000000E+06

Upper passband freq (Hz)

15.000000E+06

Passband type

- ☐ Max. flat
- ☒ Equal ripple
- ☐ Functional
- ☐ Sloping

Band-edge loss/return loss (dB)

0.10000000E-01

Loss Slope (dB/oct)

6.00000000

Flat loss (dB)

0.00000000

Function Type

- ☒ E
- ☐ F

Multiplier

0.00000000

Stopband

- ☐ Monotonic
- ☐ Equal min
- ☐ Placer
- ☒ Specified

Lower stopband freq (Hz)

0.00000

Loss (dB)

50.0000000

Upper stopband freq (Hz)

0.00000

Loss (dB)

50.0000000

Detail Parameters

of zeros at zero

0

of zeros at FQ or FS/2

0

of unit elements

5

Parametric

- ☐ Conventional
- ☐ Parametric
- ☒ Matching

R1

200

R2

50.000000E+00

ZS parameter

-1

Q for predistortion

0.00000000

Predistortion

- ☒ Passband only
- ☐ Pass/stop bands

Shifted bandpass trans. Center frequency (Hz)

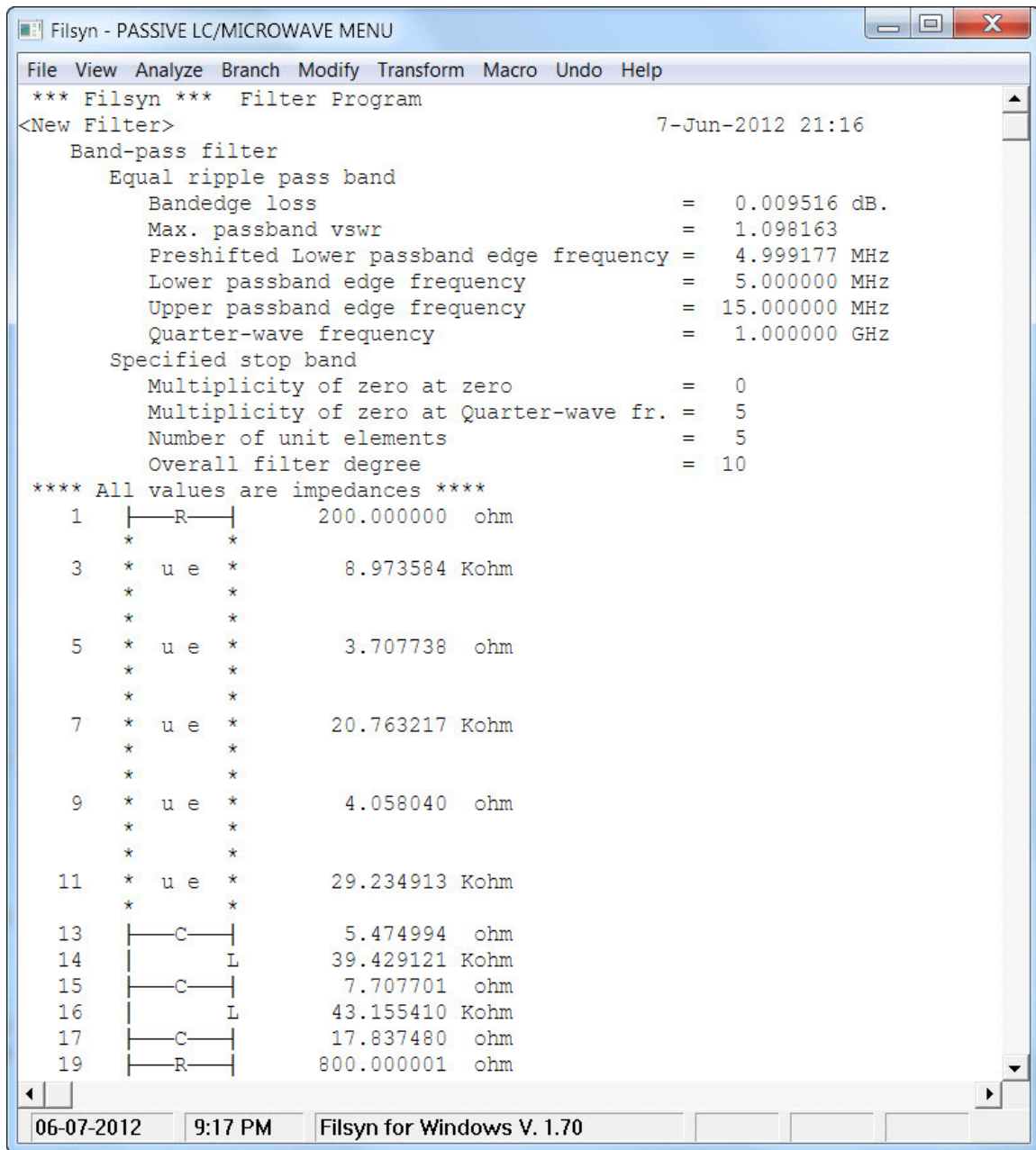
0.00000

☐ Odd parametric

☐ Save design data

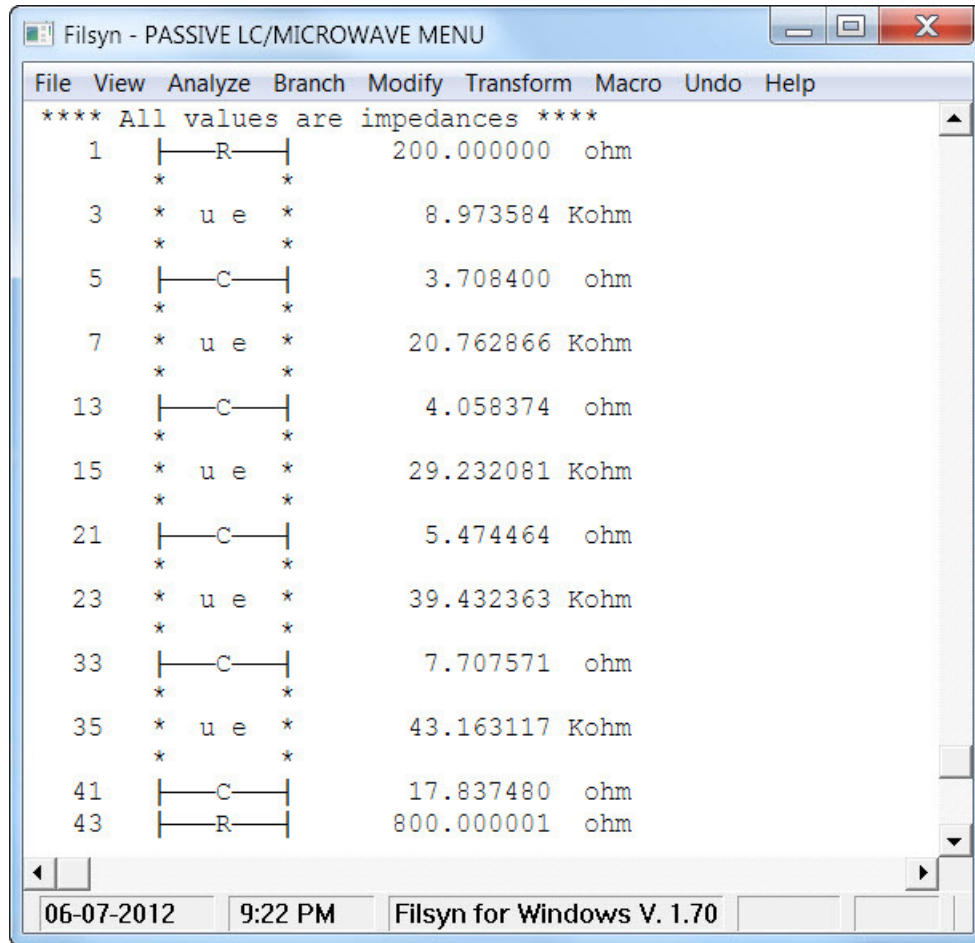
OK Cancel

The resulting circuit is shown here next:

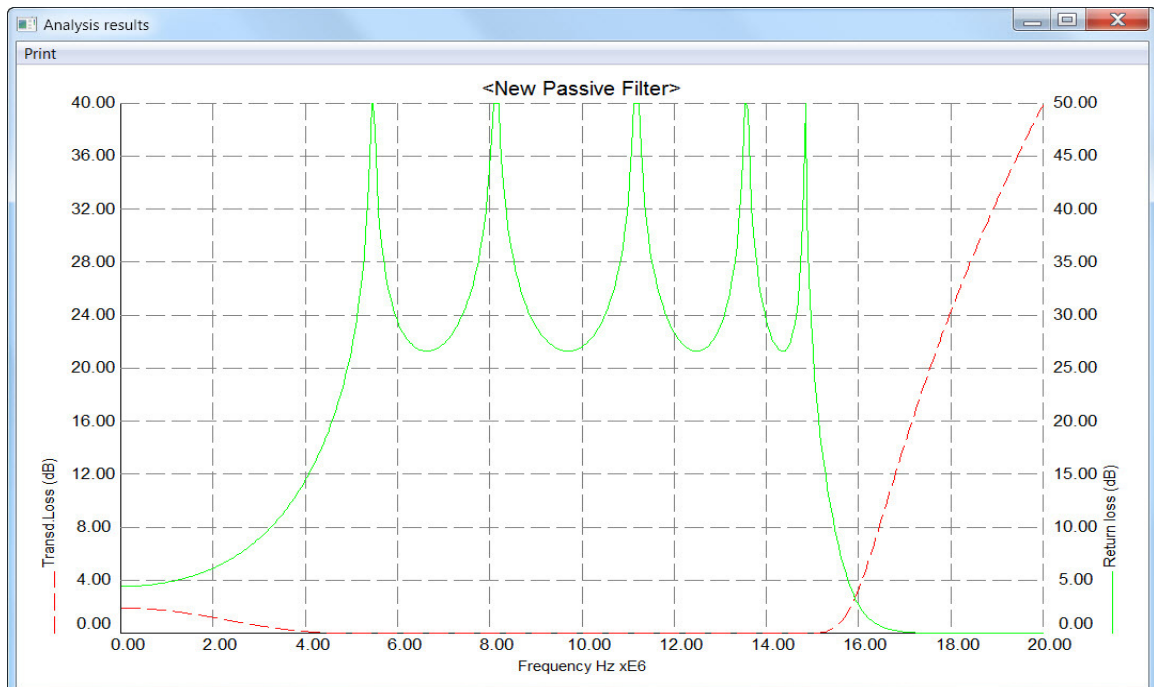


We can easily distribute the stubs to be between the unit elements by using the **Branch -> Interchange** menu option. The first step would be to interchange branches 11 and 13, which takes the C_{13} branch and converts it into a series L on the other side of the unit element. This is not a simple interchange, but the application of one of Kuroda's identities, as we have indicated before.

Performing a number of these steps, we can obtain the final result shown. The impedance level can easily be changed using the **Transform-> R scaling** menu item.



The analysis shows perfect behavior:



This structure is considerably simpler to implement, than the one without the unit elements. The performance of this circuit is essentially identical (save for the frequency scale factor) to the LC case shown above.

3. The third possibility is to use *only* unit elements. The way to do this is to specify -1 as the number of unit elements. Since this is clearly nonsense, the program takes this as the request to find the number of unit elements necessary to meet the specifications. The corresponding data input screen would look like:

<New Filter>

Filter kind

- ☐ LC
- ☒ Microwave
- ☐ Active RC
- ☐ IIR Digital

Quarter-wave/Sampling freq (Hz)

1.000000E+09

Filter type

- ☐ Low pass
- ☐ High pass
- ☒ Band pass

Lower passband freq (Hz)

10.000000E+06

Upper passband freq (Hz)

20.000000E+06

Passband type

- ☐ Max. flat
- ☒ Equal ripple
- ☐ Functional
- ☐ Sloping

Band-edge loss/return loss (dB)

0.500000000E-01

Loss Slope (dB/oct)

6.00000000

Flat loss (dB)

0.00000000

Function Type

- ☒ E
- ☐ F

Multiplier

0.00000000

Stopband

- ☐ Monotonic
- ☐ Equal min
- ☐ Placer
- ☒ Specified

Lower stopband freq (Hz)

0.00000

Loss (dB)

50.0000000

Upper stopband freq (Hz)

0.00000

Loss (dB)

50.0000000

of zeros at zero

0

of zeros at FQ or FS/2

0

of unit elements

-1

Parametric

- ☐ Conventional
- ☐ Parametric
- ☒ Matching

R1

200

R2

50.000000E+00

ZS parameter

-1

Q for predistortion

0.00000000

Predistortion

- ☒ Passband only
- ☐ Pass/stop bands

Shifted bandpass trans. Center frequency (Hz)

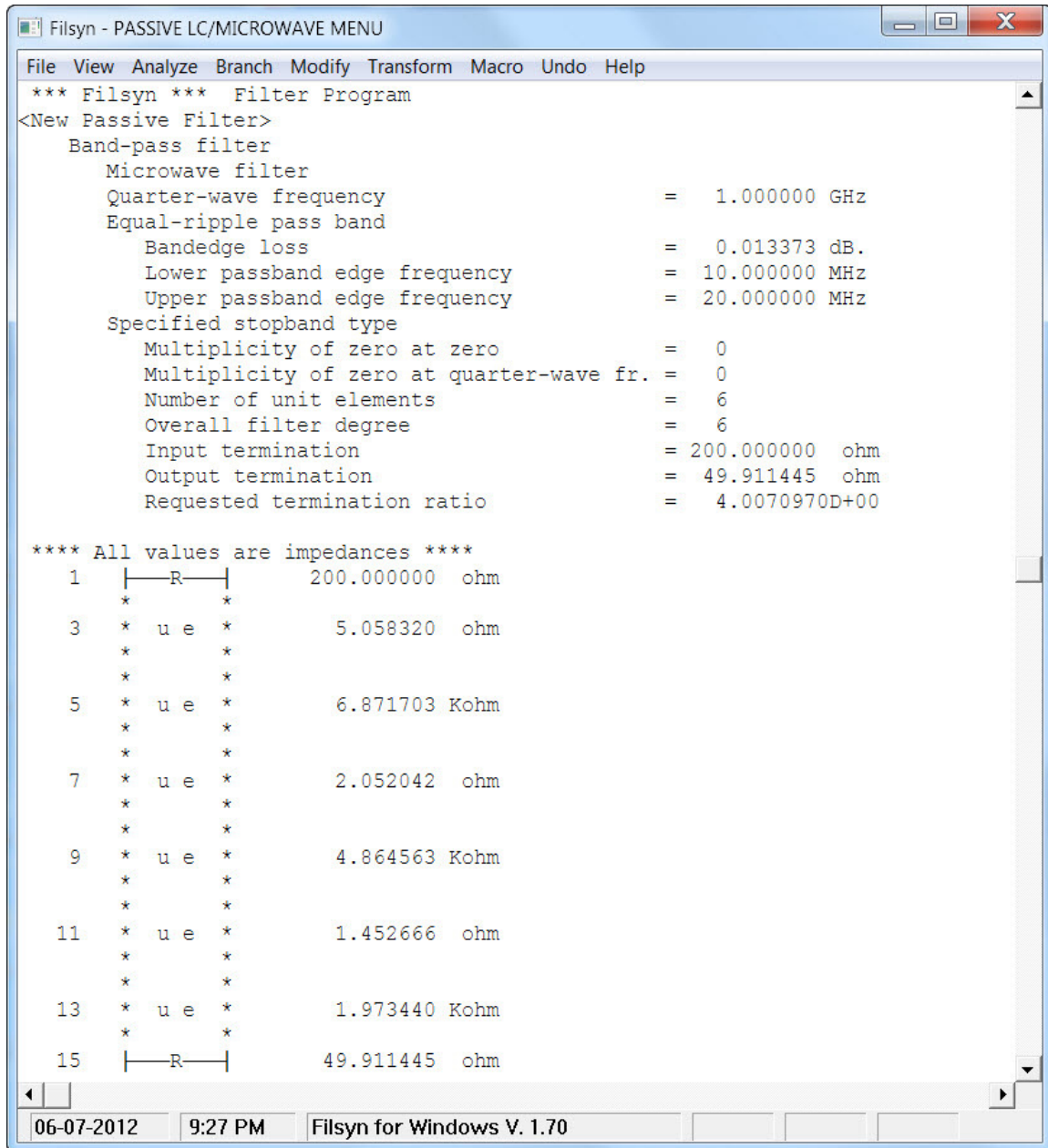
0.00000

☐ Odd parametric

☐ Save design data

OK Cancel

We reduced the bandwidth here to 10 MHz centered at 1 GHz in order to get a smaller circuit. The resulting structure is shown below and the analysis indicates perfect results again.



4. In the case of microwave filters we have still another option and that is to use a highpass filter containing only unit elements. A structure containing unit elements only is impossible to classify as a lowpass, highpass or bandpass without actually analyzing it. But such a matching highpass is in fact, very useful and goes by the name of *quarter-wave transformer*.

The data input screen gives us a moment of difficulty, however, since if we select a highpass structure, we won't be able to select the **Matching** option, because that is active only when **Bandpass** is specified. The way around this problem is to select **Bandpass** first, then click on **Matching** and then go back to the beginning and change the selection back to **Highpass**. In fact this is one of the very few instances where the data entry sequence involves backtracking.

Also, since this is a highpass, the quarter-wave frequency should be selected to be the band center and the passband edge is actually the lower end of the band of interest. Again specifying -1 as the number of unit elements, we are all set. The band is from 5 MHz to 15 MHz and the required ripple is to be less than 0.01 dB:

<New Filter>

Filter kind

- ☐ LC
- ☒ Microwave
- ☐ Active RC
- ☐ IIR Digital

Quarter-wave/Sampling freq (Hz)

10.000000E-03

Filter type

- ☐ Low pass
- ☒ High pass
- ☐ Band pass

Lower passband freq (Hz)

5.000000E-03

Upper passband freq (Hz)

0.00000

Passband type

- ☐ Max. flat
- ☒ Equal ripple
- ☐ Functional
- ☐ Sloping

Band-edge loss/return loss (dB)

0.100000000E-01

Loss Slope (dB/oct)

6.00000000

Flat loss (dB)

0.00000000

Function Type

- ☒ E
- ☐ F

Multiplier

0.00000000

Stopband

- ☐ Monotonic
- ☐ Equal min
- ☐ Placer
- ☒ Specified

Lower stopband freq (Hz)

0.00000

Loss (dB)

50.0000000

Upper stopband freq (Hz)

0.00000

Loss (dB)

50.0000000

Detail Parameters

of zeros at zero

0

of zeros at FQ or FS/2

0

of unit elements

-1

Parametric

- ☐ Conventional
- ☐ Parametric
- ☒ Matching

R1

200.000000E+00

R2

50.000000E+00

ZS parameter

-1

Q for predistortion

0.00000000

Predistortion

- ☒ Passband only
- ☐ Pass/stop bands

Shifted bandpass trans.

Center frequency (Hz)

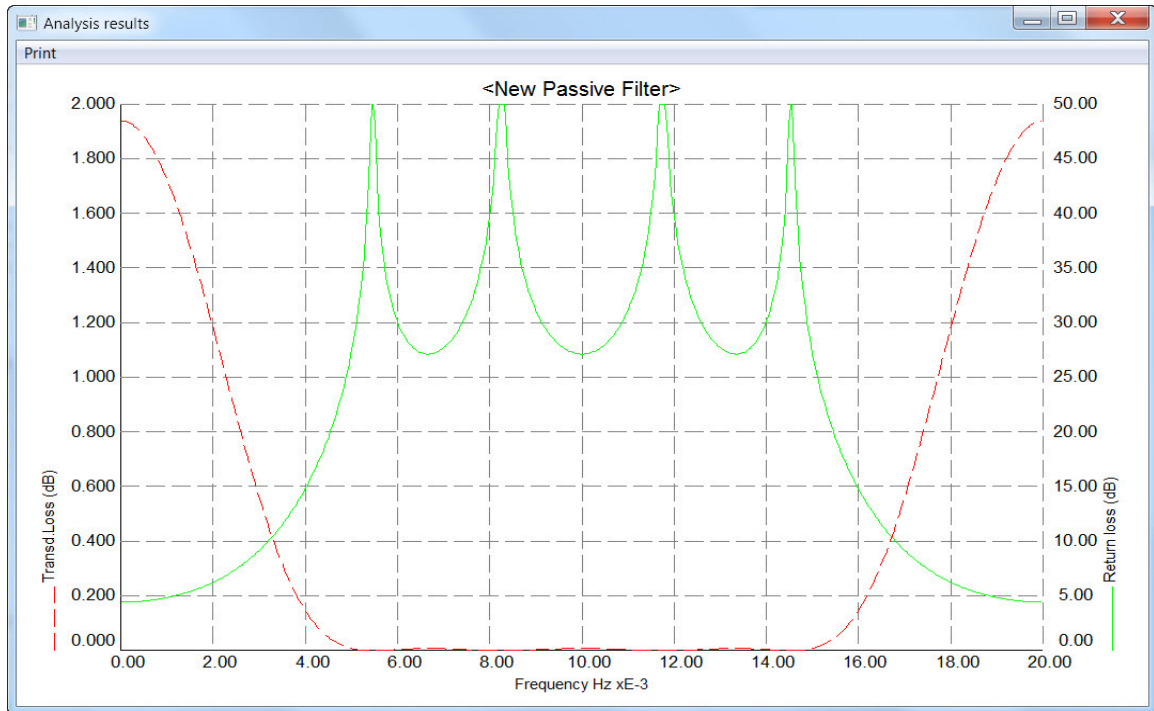
0.00000

☐ Odd parametric

☐ Save design data

OK Cancel

Note that while **Matching** does not seem active, it is still selected and in fact, active. The resulting matching circuit is shown below together with the results of the analysis:



```

Filsyn - PASSIVE LC/MICROWAVE MENU
File View Analyze Branch Modify Transform Macro Undo Help
*** Filsyn *** Filter Program
<New Passive Filter>
  High-pass filter
    Microwave filter
    Quarter-wave frequency                = 10.000000 mHz
    Equal-ripple pass band
      Bandedge loss                      = 0.008445 dB.
      Lower passband edge frequency      = 5.000000 mHz
    Specified stopband type
      Multiplicity of zero at zero       = 0
      Number of unit elements            = 4
      Overall filter degree              = 4
      Input termination                   = 200.000000 ohm
      Output termination                  = 50.000000 ohm
      Requested termination ratio         = 4.000000D+00

**** All values are impedances ****
  1  |---R---|      200.000000 ohm
    *   *
  3  * u e *      169.120632 ohm
    *   *
    *   *
  5  * u e *      122.252716 ohm
    *   *
    *   *
  7  * u e *      81.797774 ohm
    *   *
    *   *
  9  * u e *      59.129391 ohm
    *   *
 11  |---R---|      50.000000 ohm

```

06-07-2012 9:35 PM Filsyn for Windows V. 1.70 CAPS

We had to do a rescaling operation here also, to get the correct impedance level.

One might ask the question as to which of these solutions is preferable. This is difficult to answer. The all-unit element bandpass needs about twice as many unit elements than the highpass (quarter-wave transformer), but those unit elements will be substantially shorter, since the quarter-wave frequency is much higher. The highpass solution has fewer, but longer unit elements and the impedance values of the highpass circuit are substantially more convenient all being in between the two terminating R values. Also, if you rely on the filtering properties of the circuit as well, then the bandpass solution has the advantage, since the next passband could be much further away.

6. ACTIVE RC ANALYSIS

6.1 Available structures

Once the filter requirements are specified in the Main menu of the **Filsyn** program with an active RC filter type selected, we enter the active RC analysis segment of the program. First, if our filter was a lowpass one and we need a band reject, here is the place to convert it. This is followed by the selection of the implementation we wish to use.

We have three options here:

1. Cascaded second order sections (Fig. 5). Each of these sections implements a second order factor from the overall numerator and the denominator polynomials. The sequence and pairing of factors is done automatically using a semi-empirical optimal procedure, but it can be changed later by performing a permutation procedure.
2. Cascaded second order sections with feedback paths from the outputs of all but the first section to a common input summing point (Fig. 6). This structure is called the *Follow the leader* (FLF) feedback. The individual numerators still are factors of the overall numerator, but the denominators – in this implementation of the design procedure – are different and have zeros on the imaginary axis, except for the very first section. The sequence of numerator factors can be selected by the user or can be left for the program. However, once selected, this sequence cannot be changed later, since any permutation of these would change the overall transfer function drastically. If the overall filter degree is odd, the first section will have a first order transfer function instead of a quadratic one.
3. The last structure is similar, except the feedback paths go from all but the first section output to the input of two sections back (Fig. 7). This arrangement is called the *Leapfrog* (LF) feedback structure. Again the numerator factors are factors of the overall numerator and their sequence may be selected by the user, or left to the program to select and may not be changed later. All denominators except those of the end sections, have zeros on the imaginary axis. In the case of an odd overall degree, the first order section will be either the first or the last section.

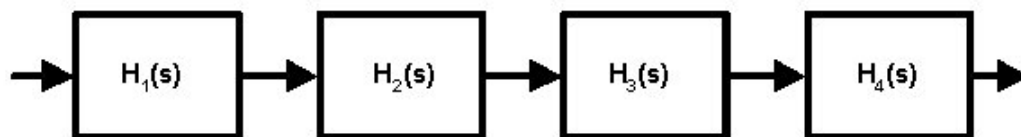


Fig. 5 Cascaded second order sections

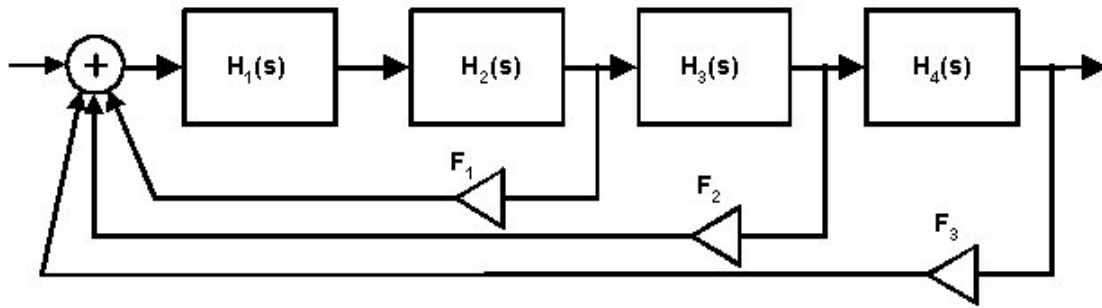


Fig. 6 Follow-the-leader feedback structure

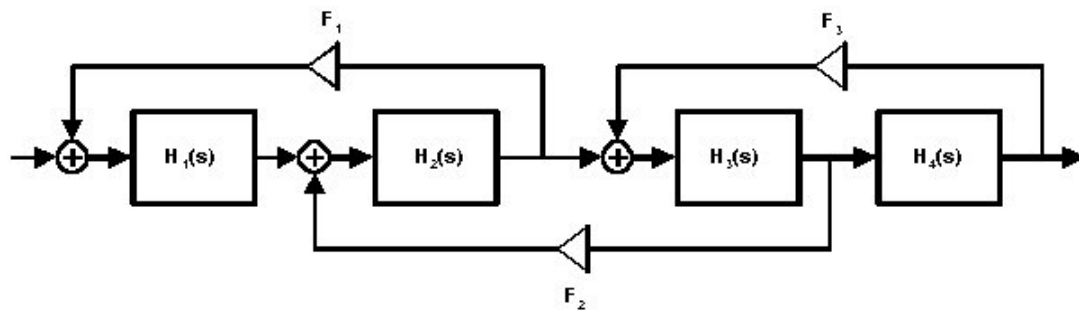


Fig. 7 Leapfrog feedback structure

Consider the following quasi-elliptic highpass filter:

<New Filter>

Filter kind

- ☐ LC
- ☐ Microwave
- ☒ Active RC
- ☐ IIR Digital

Quarter-wave/Sampling freq(KHz)

0.00000

Filter type

- ☐ Low pass
- ☒ High pass
- ☐ Band pass

Lower passband freq (KHz)

3.000000E+00

Upper passband freq (KHz)

0.00000

Passband type

- ☐ Max. flat
- ☒ Equal ripple
- ☐ Functional
- ☐ Sloping

Band-edge loss/return loss (dB)

0.200000000

Loss Slope (dB/oct)

6.000000000

Flat loss (dB)

0.000000000

Function Type

- ☒ E
- ☐ F

Multiplier

0.000000000

Stopband

- ☐ Monotonic
- ☐ Equal min
- ☒ Placer
- ☐ Specified

Lower stopband freq (KHz)

0.00000

Loss (dB)

50.00000000

Upper stopband freq (KHz)

0.00000

Loss (dB)

50.00000000

Detail Parameters

of zeros at zero

2

of zeros at infinity

0

of unit elements

0

Parametric

- ☒ Conventional
- ☐ Parametric
- ☐ Matching

R1

50.000000E+00

R2

50.000000E+00

ZS parameter

-1

Q for predistortion

0.000000000

Predistortion

- ☒ Passband only
- ☐ Pass/stop bands

Shifted bandpass trans.

Center frequency (KHz)

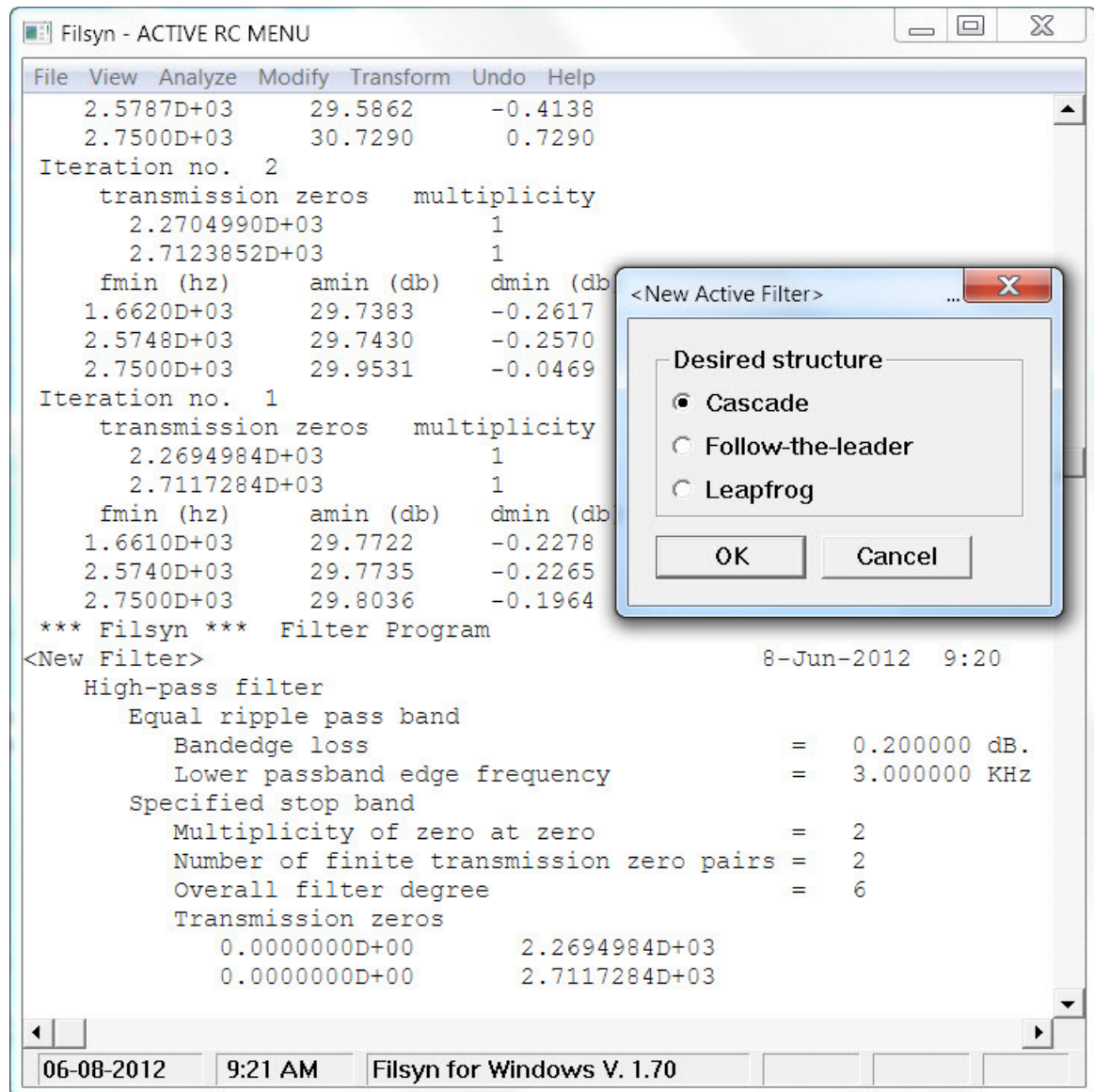
0.00000

☐ Odd parametric

☐ Save design data

OK Cancel

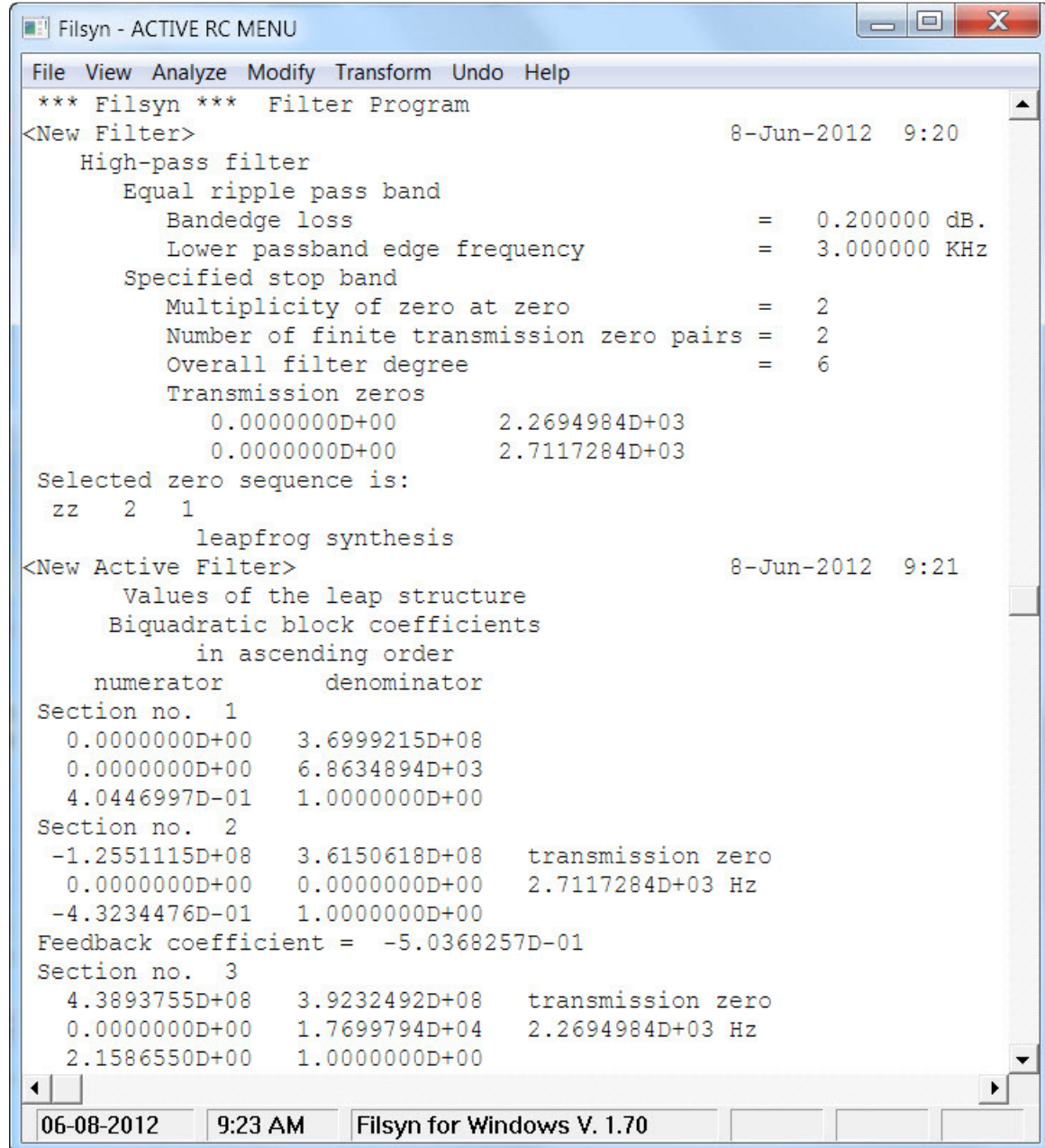
where the **Detail Parameters** contain a single breakpoint at 2.75 KHz and 30 dB, while specifying two finite transmission zeros. After the stopband optimization is finished, we reach the point with results that are acceptable:



and selecting the **Leapfrog** option, the next step is:



Opting for the computer-selected zero sequence, we have the complete transfer function:



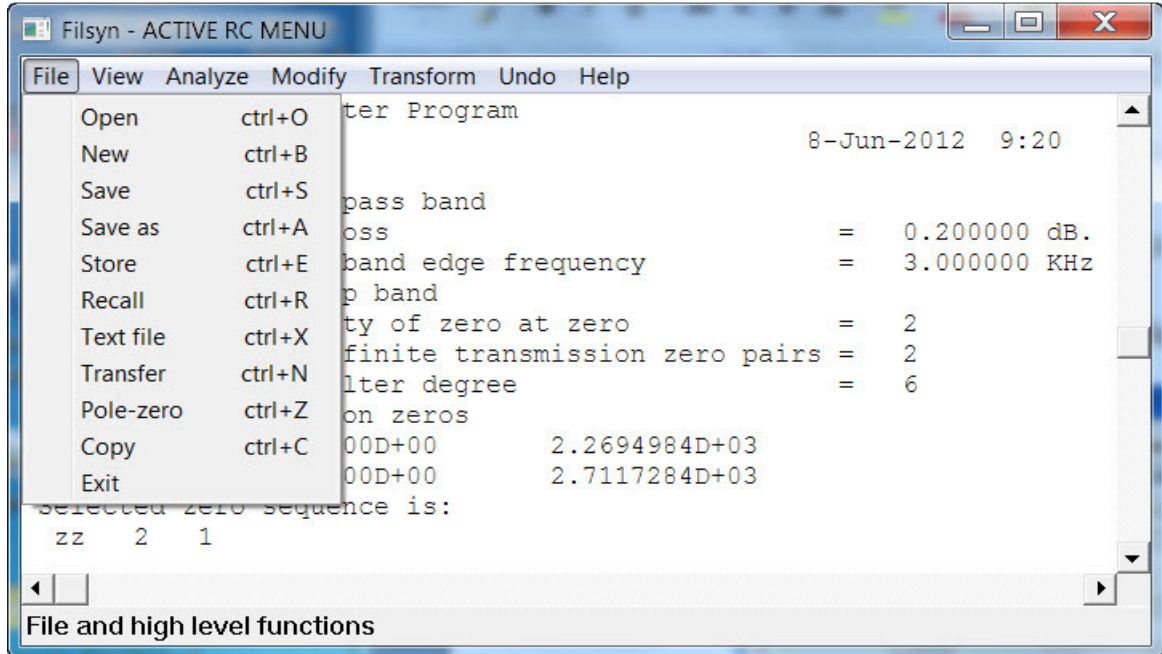
```

Filsyn - ACTIVE RC MENU
File View Analyze Modify Transform Undo Help
*** Filsyn *** Filter Program
<New Filter> 8-Jun-2012 9:20
  High-pass filter
    Equal ripple pass band
      Bandedge loss = 0.200000 dB.
      Lower passband edge frequency = 3.000000 KHz
    Specified stop band
      Multiplicity of zero at zero = 2
      Number of finite transmission zero pairs = 2
      Overall filter degree = 6
      Transmission zeros
        0.000000D+00 2.2694984D+03
        0.000000D+00 2.7117284D+03
    Selected zero sequence is:
      zz 2 1
        leapfrog synthesis
<New Active Filter> 8-Jun-2012 9:21
  Values of the leap structure
  Biquadratic block coefficients
    in ascending order
      numerator denominator
Section no. 1
  0.0000000D+00 3.6999215D+08
  0.0000000D+00 6.8634894D+03
  4.0446997D-01 1.0000000D+00
Section no. 2
  -1.2551115D+08 3.6150618D+08 transmission zero
  0.0000000D+00 0.0000000D+00 2.7117284D+03 Hz
  -4.3234476D-01 1.0000000D+00
Feedback coefficient = -5.0368257D-01
Section no. 3
  4.3893755D+08 3.9232492D+08 transmission zero
  0.0000000D+00 1.7699794D+04 2.2694984D+03 Hz
  2.1586550D+00 1.0000000D+00
06-08-2012 9:23 AM Filsyn for Windows V. 1.70
  
```

In the printout above the computer selected zero sequence is shown and here 'zz' represents a double zero at zero frequency, while the other integers represent the serial numbers of the transfer function zeros the corresponding section implements.

Let us see now the possible operations available here. Some will be familiar, others are unique to the active RC filter segment. The **File** menu option has a large number of submenu items:

6.2 The 'File' menu options

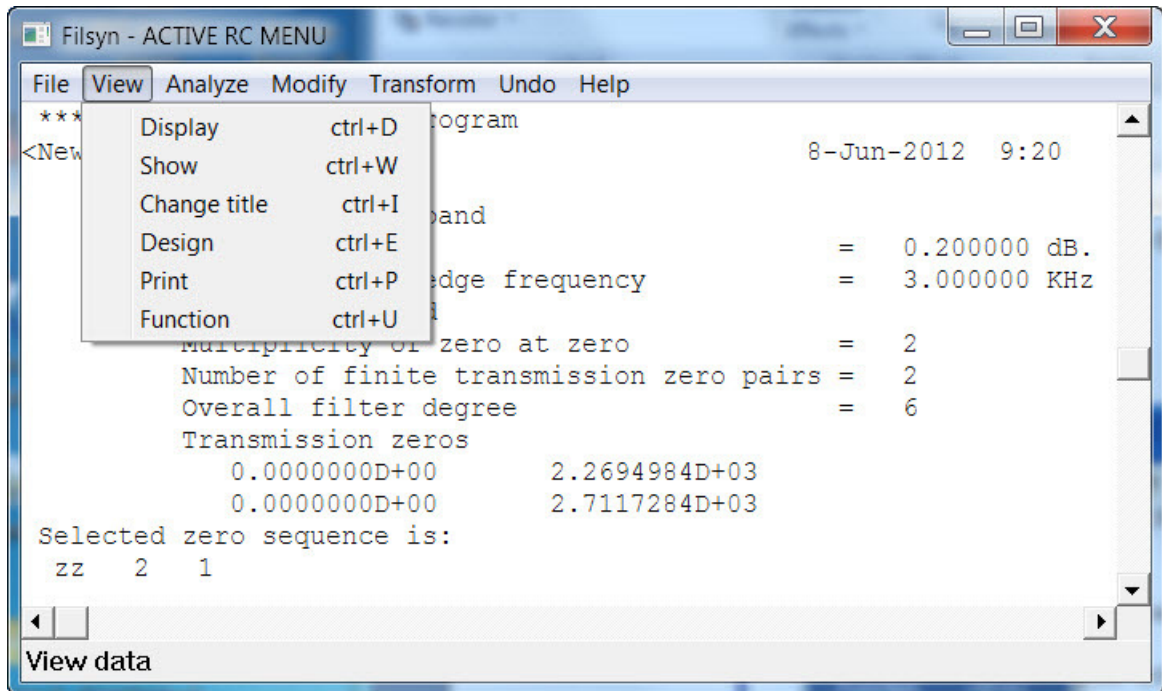


1. The **Open** option recalls a previously saved set of filter data, but the current filter data will be overwritten and lost. If we select this option, the warning about this will be prominently displayed before action can be taken.
2. The **New** option permits us to restart the synthesis or start a new one without the need for stopping the program and restarting it.
3. **Save** and **Save as** are the usual menu options saving the current data in a permanent file, while **Store** and **Recall** save the data in temporary file and recall that data respectively. If we close the program, the temporary data is lost.
4. The **Text file** option will write the design data to a file that can be displayed and edited in any text editor and used for documentation purposes only. It may not be read back into the program. If a circuit implementation has not yet been selected, or is incomplete, the program warns us about it, since that information would also be written to this text file.
5. The **Transfer** option permits us to write text files for the purpose of transferring the circuit diagram to other programs for further analysis. Currently we can write these netfiles for *Spice* and *SuperStar* programs, or, if the filter has been converted to a switched-capacitor form (see later), to the *Switcap* and *Scasy* programs.
6. **Pole-zero** option lets us write the transfer function data to a text file again for documentation purposes. If the circuit contains *no* delay equalizer sections, then the file will have a “.pz” extension and may be read back into **Filsyn** using the

Functional input option. However if it *does* contain delay equalizers, the file will have a “.dat” extension and may be read back into **Filsyn** only through the **Analysis** option. The data format in these files will be the appropriate one for the application and clearly indicated.

7. The **Copy** option allows us to select any area of the output by the usual mouse operations and copy it to the clipboard, while the **Exit** option of course, terminates the program.

6.3 The ‘View’ menu options

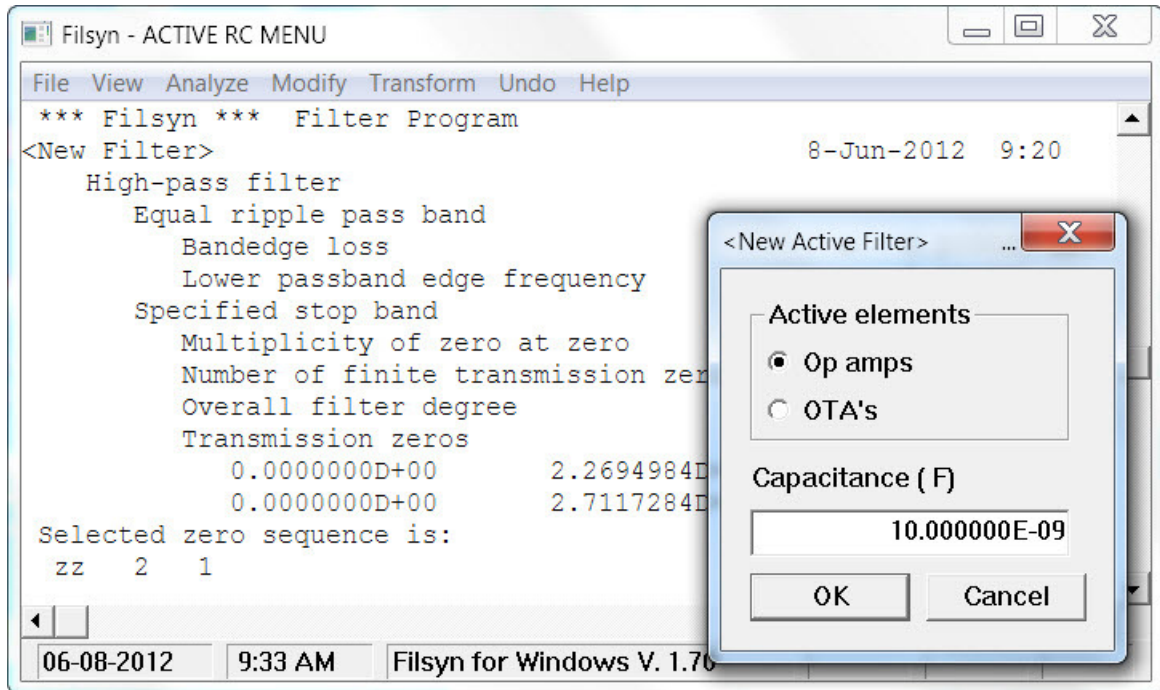


The **View** menu option has the following submenu items:

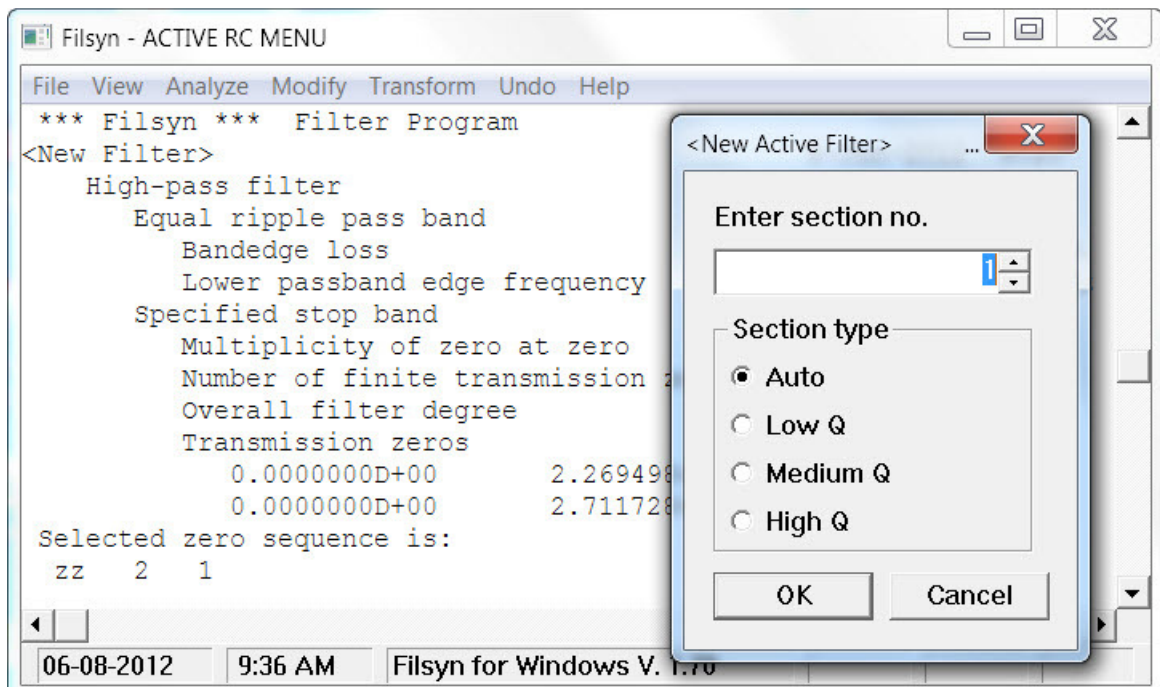
1. The **Display** option prints the filter data that is seen on this screen, to the screen again, while the **Show** option prints the transfer function to the screen again.
2. **Change title** does just what it says.
3. The **Design** option is one of the most significant here in that it provides us with actual circuit diagrams and element values implementing the filter. Here we first must select the type of active element we wish to use. The two possibilities are:
 - a) Operational amplifiers, that are represented as ideal voltage-controlled voltage sources with essentially infinite gains, or
 - b) Operational transconductance amplifiers (OTA), represented by ideal voltage-controlled current sources having a specified conductance (current/voltage ratio) value.

We must also specify a preferred capacitance value and the program will select as many to have this value as possible.

The **Design** option therefore leads to a submenu:



Selecting the operational amplifier option, we have to perform the design on a section-by-section basis, since for each second order section we will have more than one possible implementation:



The 'Q value' here is the Q value of the pole of the second order transfer function of the section we are implementing and is computed as:

$$Q = (\sqrt{a_2})/a_1$$

if the denominator is of the form:

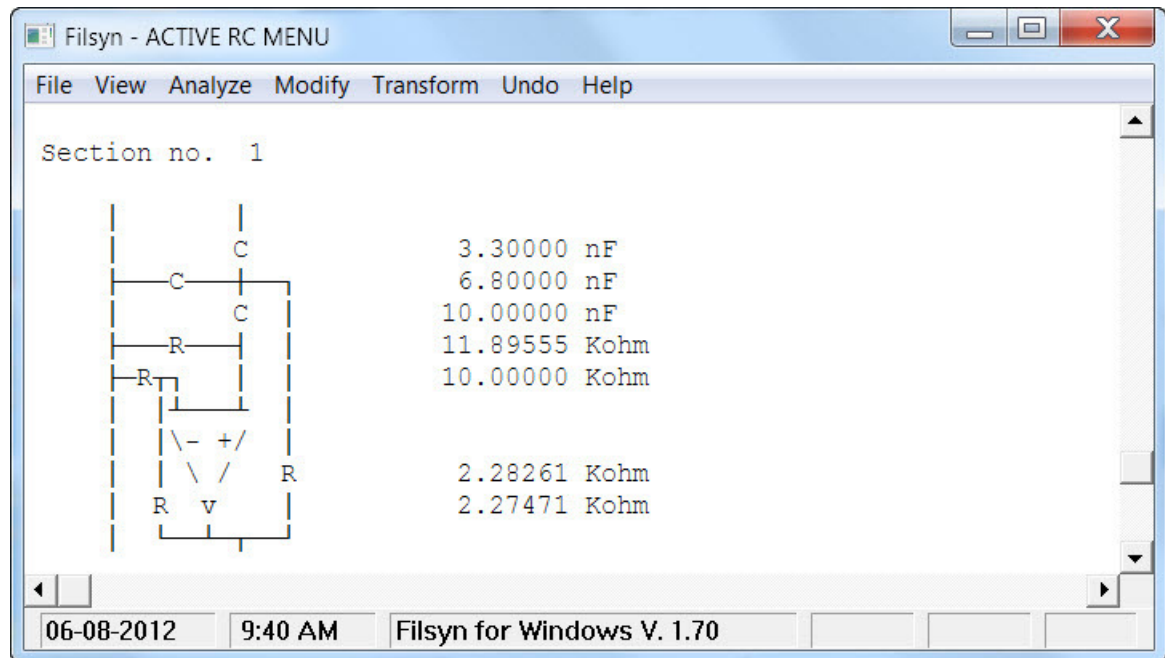
$$S^2 + a_1S + a_2$$

The ranges are specified as:

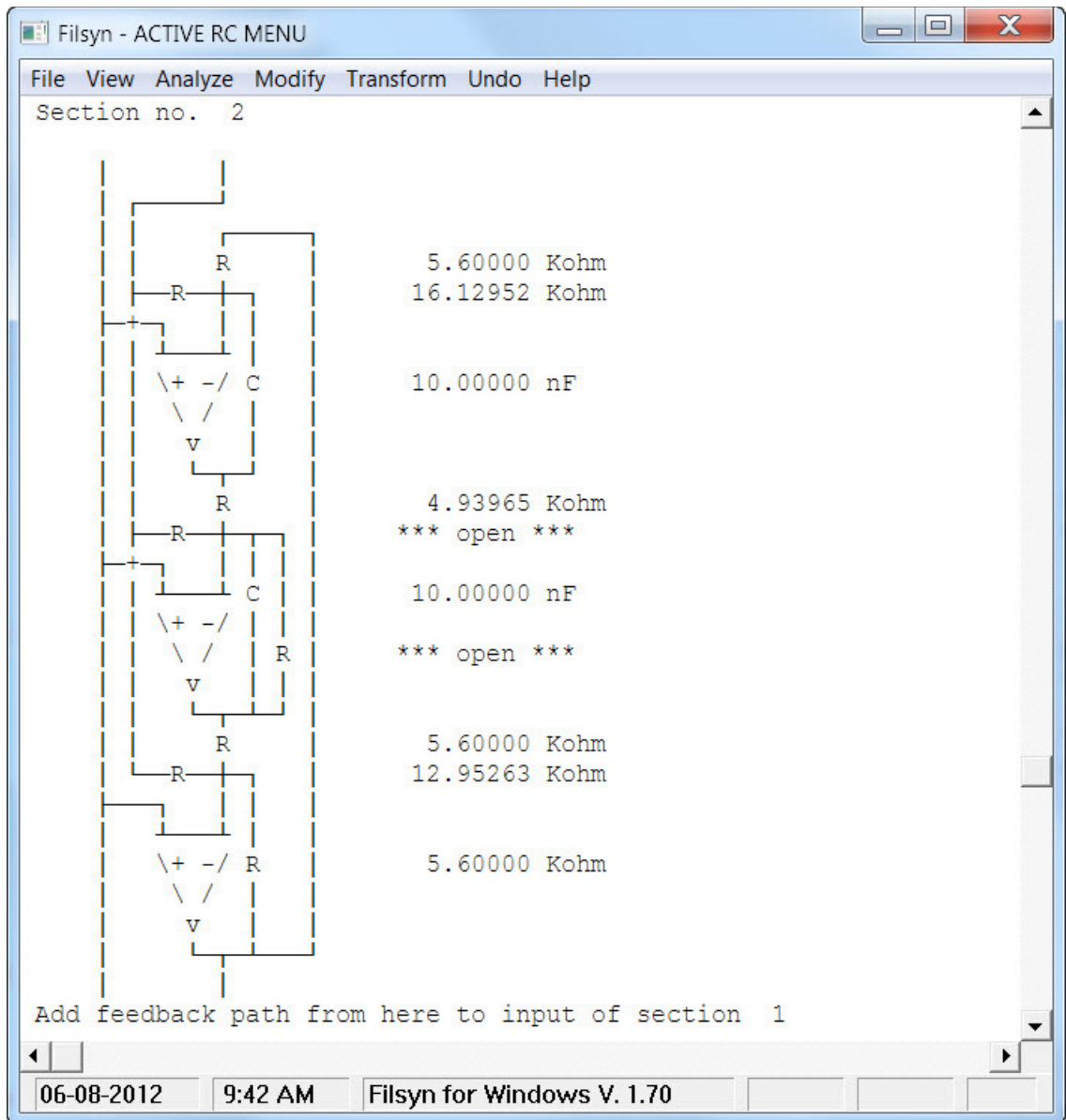
Low Q:	$0 < Q < 2.0$
Medium Q:	$2.0 < Q < 20.0$
High Q:	$20.0 < Q$

The **Auto** selection performs this computation and selects the proper Q value range, but we can override it by selecting whichever other option we wish. Note however, that sometimes our choice is not available, in which case the Auto selection automatically replaces it.

In this particular case, the **Auto** selection in fact, selected the low-Q implementation for us:

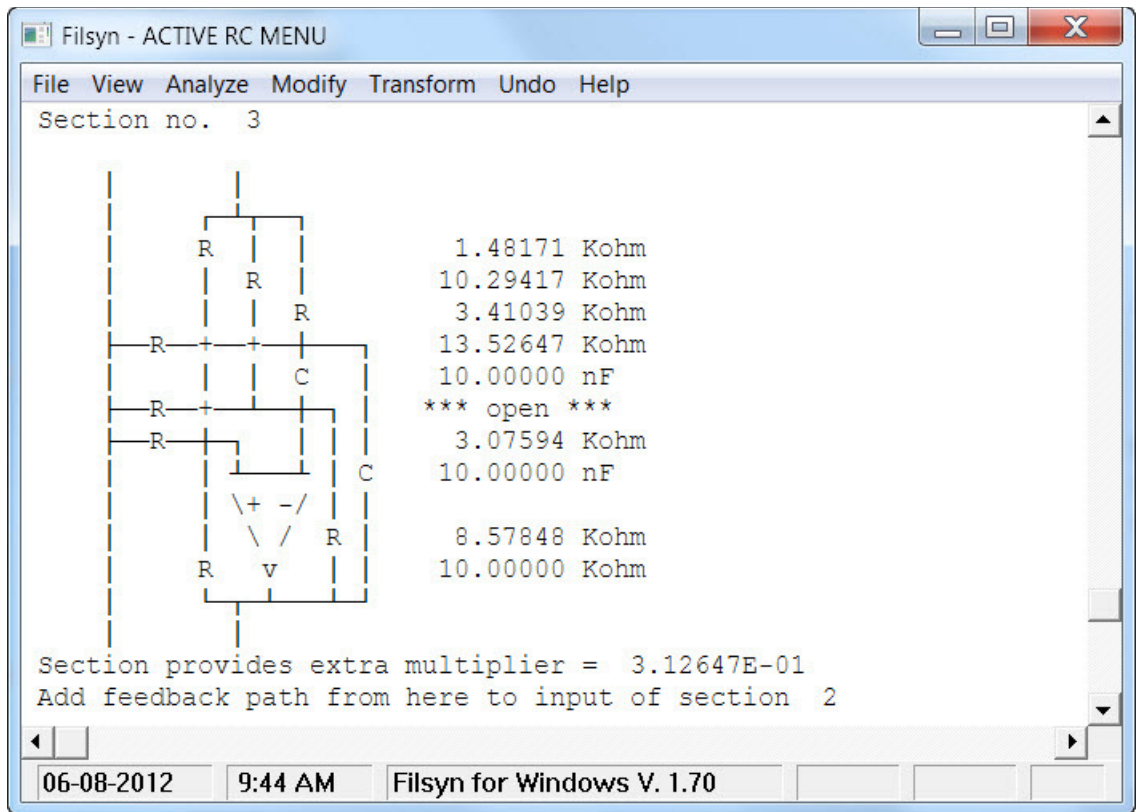


The next section has its pole on the imaginary axis, hence its Q value is infinite. Therefore, whatever we select, the program will use the high-Q implementation as the only one available:



Note the comment at the end about the feedback path. The actual circuitry implementing this feedback is not shown, simply because that depends too much on other factors, like the implementation of the first section (which, although known at this time, might be changed later to something else).

The last section will be a low-Q section again:



This implementation creates the transfer function with a change in the multiplier. This must also be taken into account in the feedback path and may also be compensated for by introducing the inverse multiplier in one of the other sections.

At this stage we have completed the actual implementation (save for the feedback paths) and could write one of the transfer files. That would have to be edited later to include the necessary circuitry for the feedback paths. More about that later.

At this stage we wish to point out, that once the active element has been selected, we are committed to its use. If we wish to try the other active element, we must save the filter data, exit the program, reenter it and recall the saved filter. Having done that, we can indeed select the **View->Design** menu option and select the OTA as the active element. Without actually displaying the resulting circuitry here (it would take up a large amount of space), we find that there are no additional options and the complete filter implementation is done all at once. The feedback paths are still not shown on the screen, but if we write a transfer file, we would see, that in fact the feedback paths are all there and no additional editing is required to implement them.

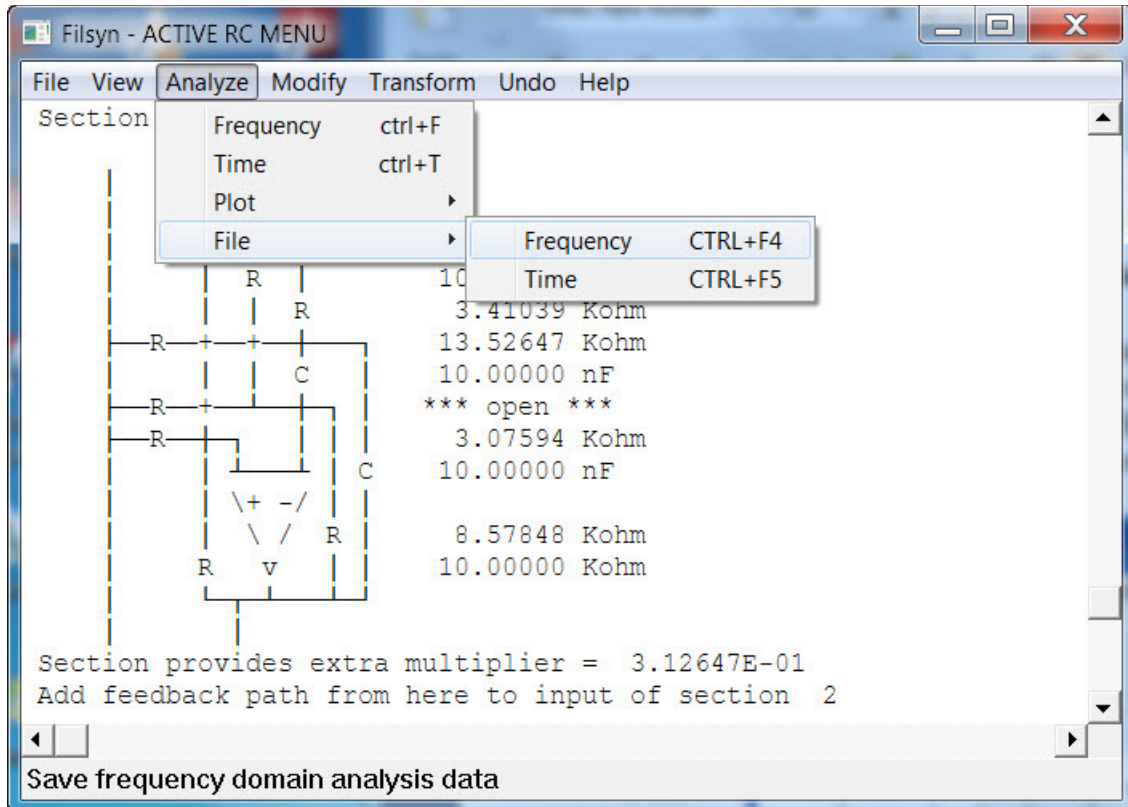
Two more submenu options are available under the *View* menu item:

4. **Print** is a switch to activate or deactivate the print option, which directs everything that shows up on the output screen also to the printer or a text file.

- The last item, **Function** displays the overall transfer function, if available, on the screen.

6.4 The 'Analyze' menu options

Next comes the **Analysis** menu item, with the usual array of frequency domain and time domain analyses. Note however, that the analysis is done using the transfer function, *not* the circuit implementation. That is why it is essential to provide a path to the *Spice* program or equivalent, for the proper analysis of the actual circuitry.



Both the frequency and the time domain analyses are restricted to a maximum of 501, equidistant points. The frequency domain analysis yields the loss, phase and delay functions, while in the time domain we compute the impulse and step responses. The results can be displayed on the screen and any one or two results can be plotted.

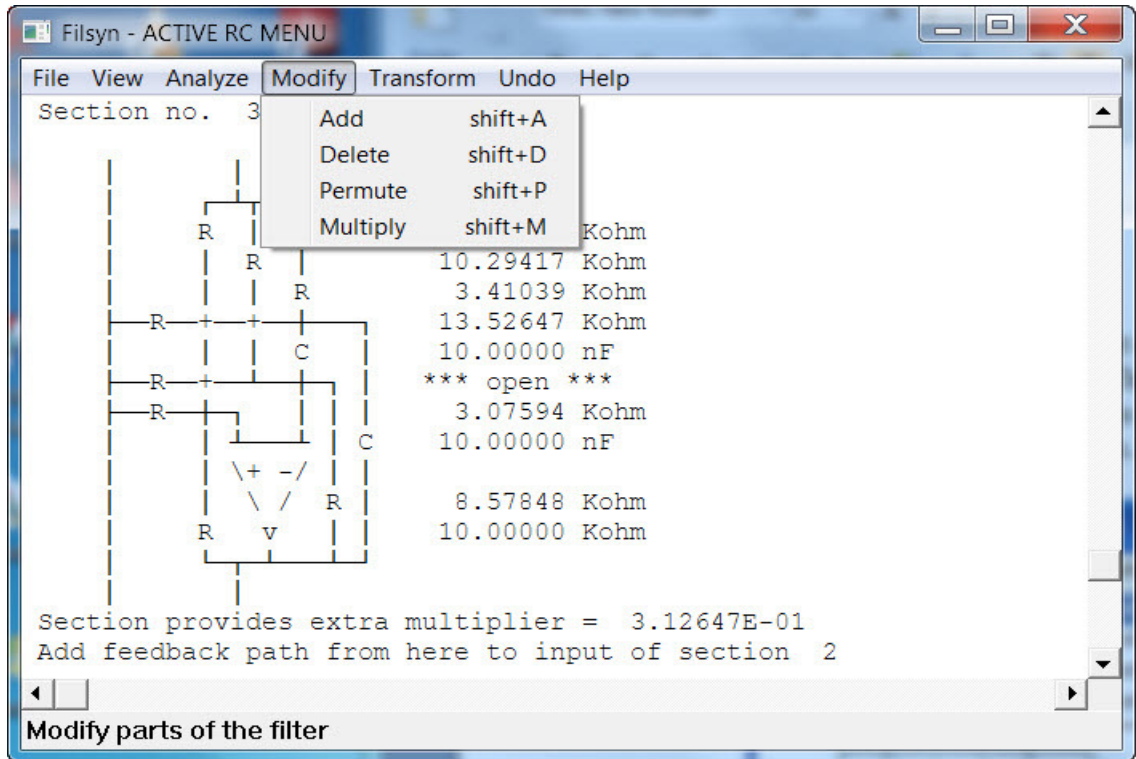
The last **File** submenu item prints the analysis results to a text file for documentation purposes.

6.5 The 'Modify' menu options

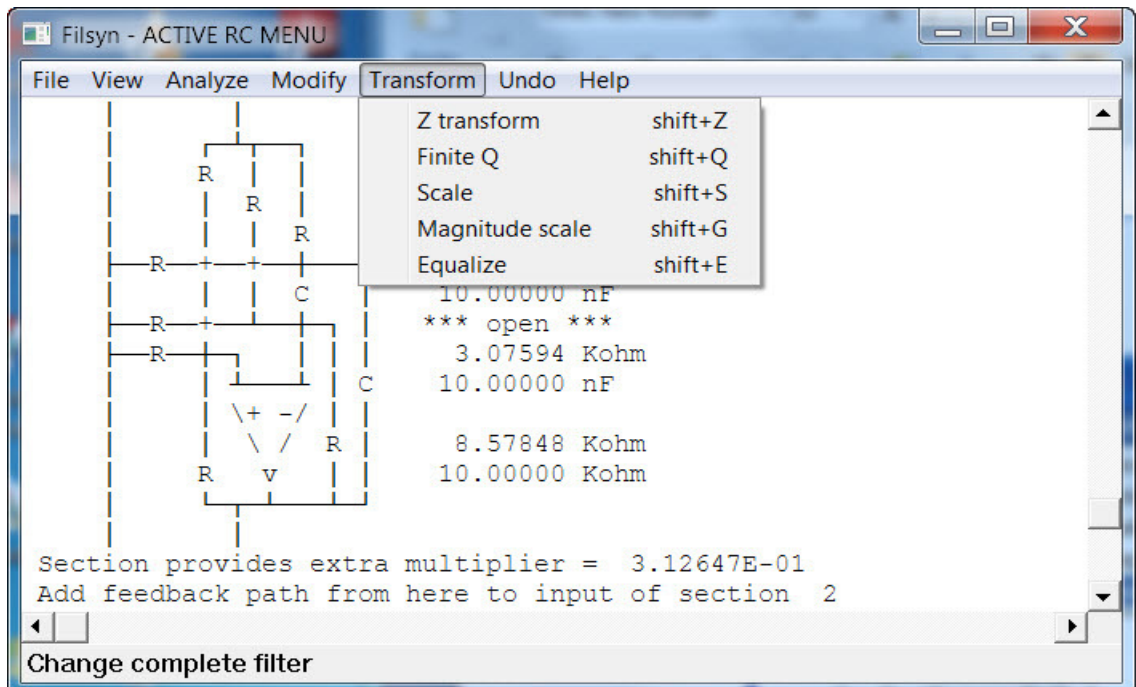
Next comes the **Modify** menu option with submenu items as follows:

These submenu options are fairly simple. **Add** and **Delete** will add a second order section or deletes one, respectively, while **Permute** will permute the numerator or the

denominator factors. This last one works only on cascade implementations, not on the FLF or LF feedback forms. The last item permits us to multiply one of the second order section transfer functions by a constant, thereby compensating for the multiplier introduced by another section. See the section shown in the background, for instance.

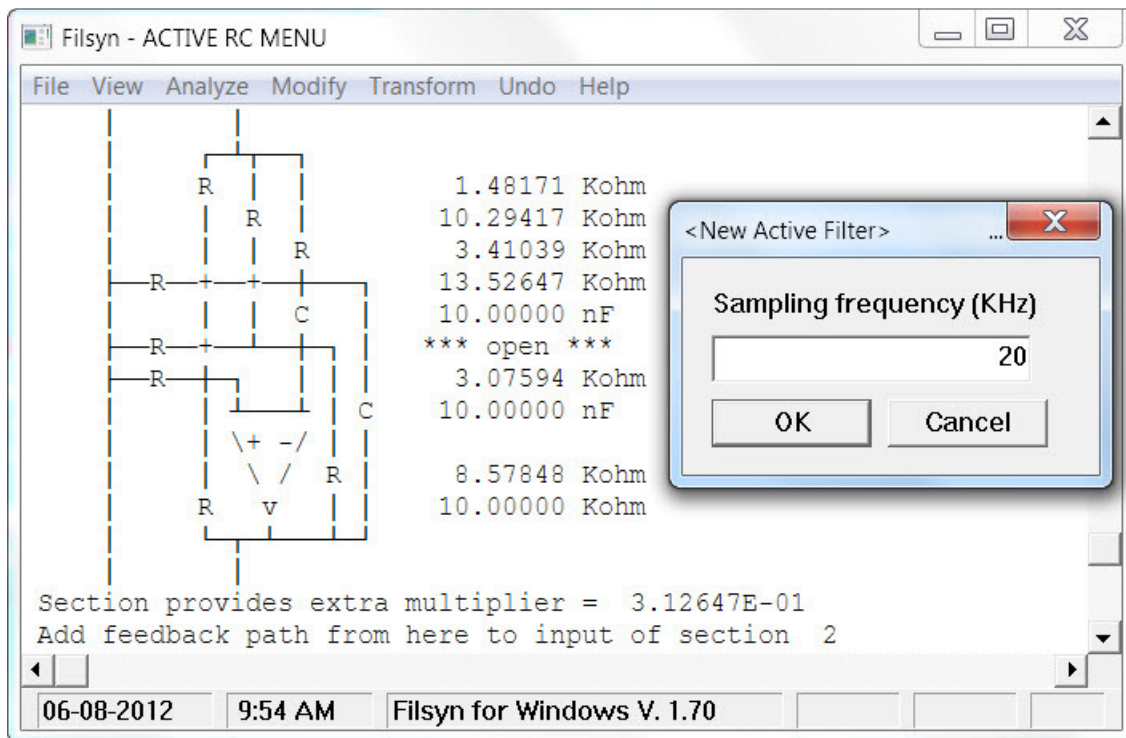


6.6 The 'Transform' menu option provides more interesting submenu options:

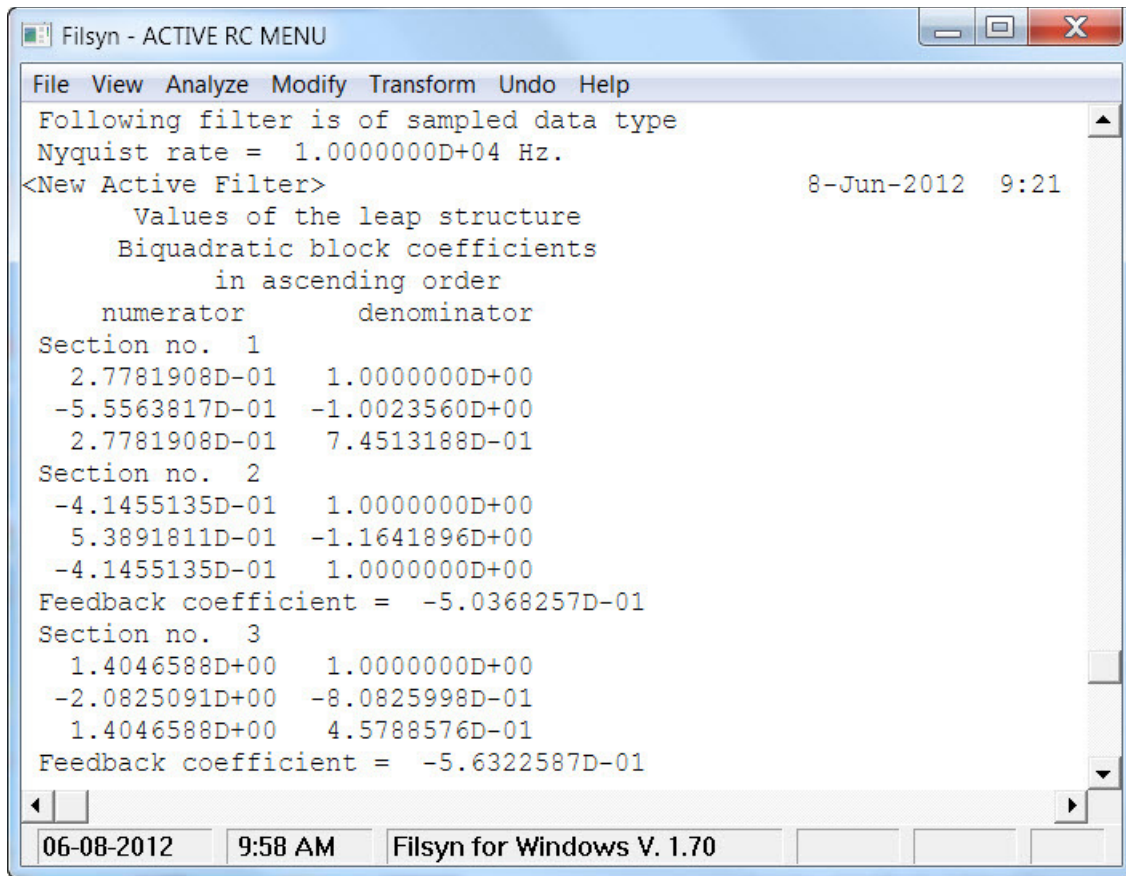


1. The **Z transform** option converts the analog transfer function into a digital one using the bilinear Z-transform method (see the IIR digital part of the manual for details). This can then be implemented as a switched-capacitor circuit. For the actual circuit design, we use the Fleischer-Laker procedure (see ref. [15]) as the most effective and stable one.

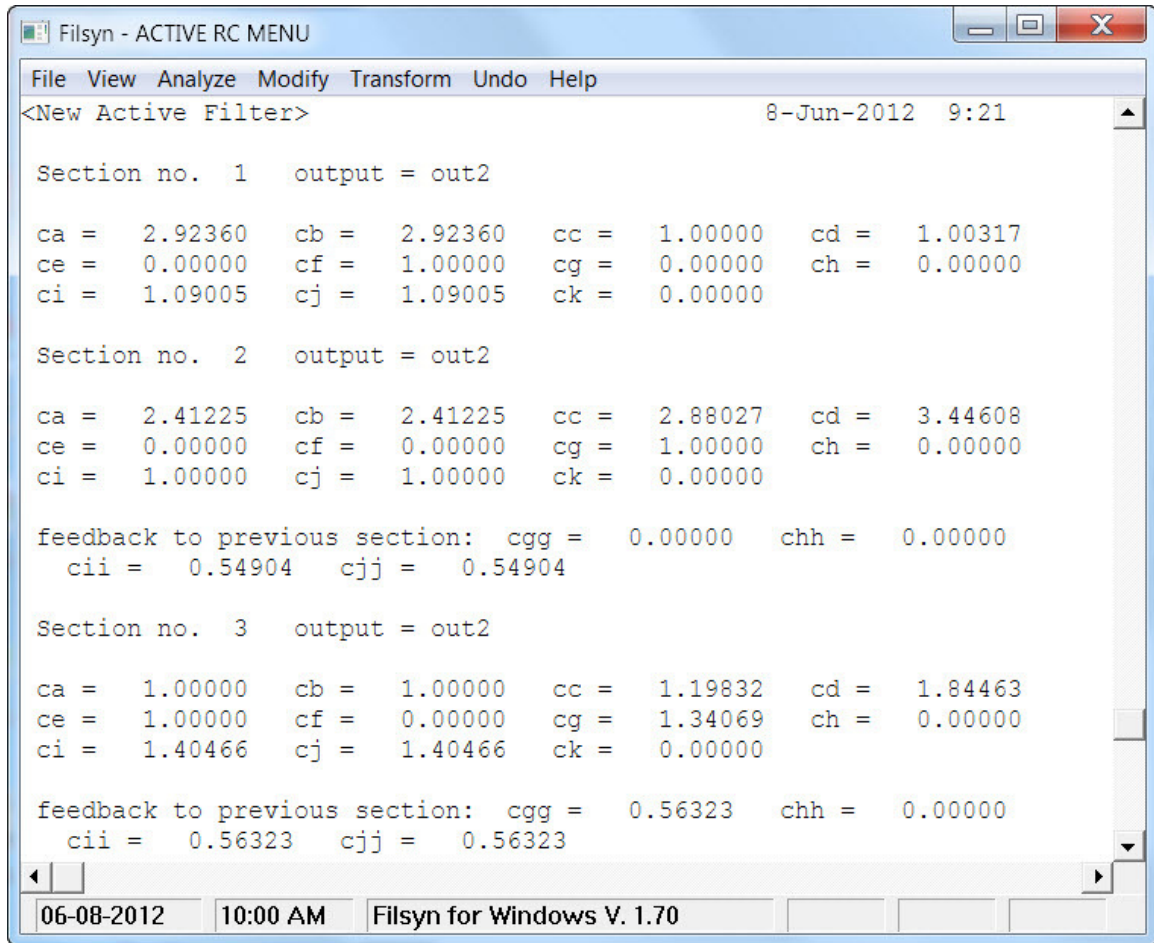
Let us demonstrate this using a 20 KHz sampling frequency, which will leave the filter specification pretty much unchanged.



The resulting transfer function is shown next:



Invoking the **View->Design** menu option again, we get the complete implementation without any further prompt:



The capacitor element values are all relative ones where the smallest nonzero value is the unit, with the exception of some of the feedback values. The largest element value spread is 6.28. The implementations for first and second order sections are shown below in Figs. 8 and 9.

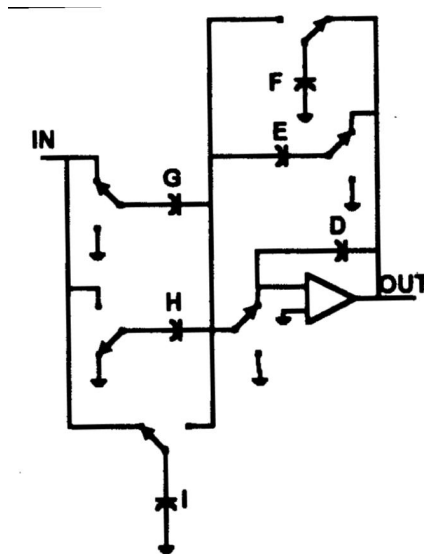


Fig. 8 First order switched capacitor section

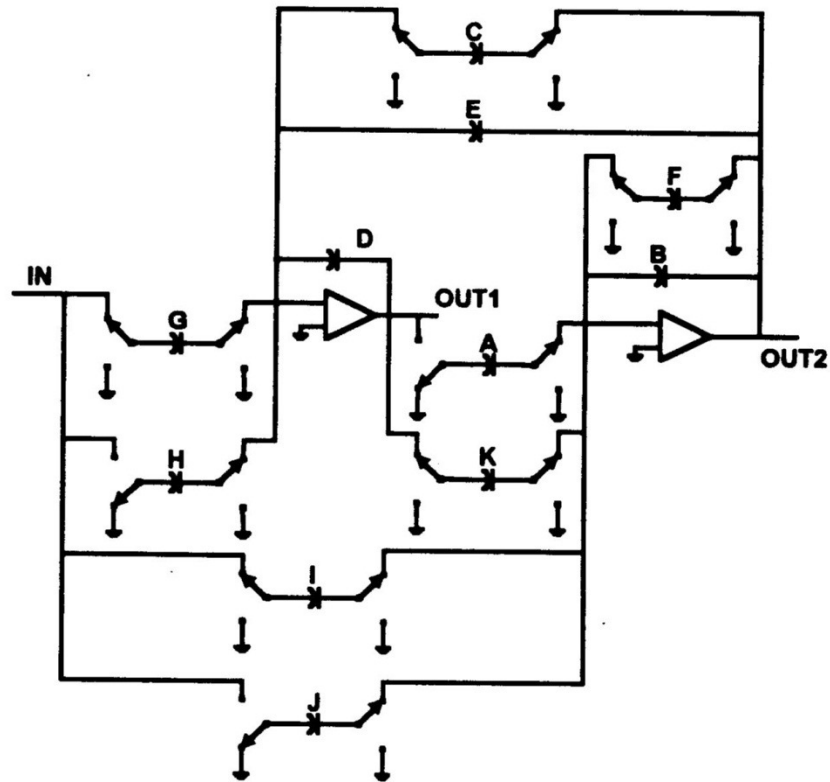
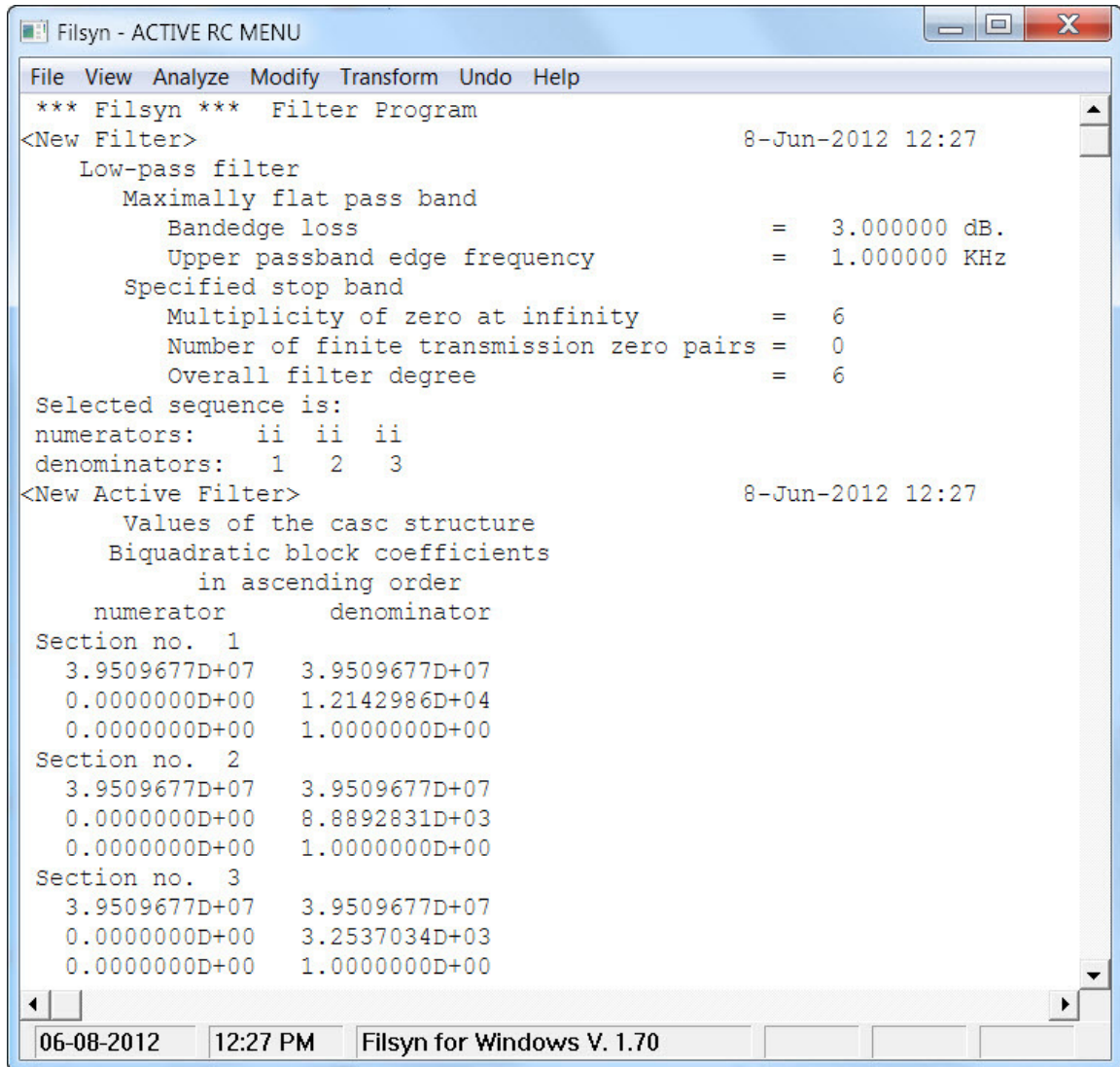
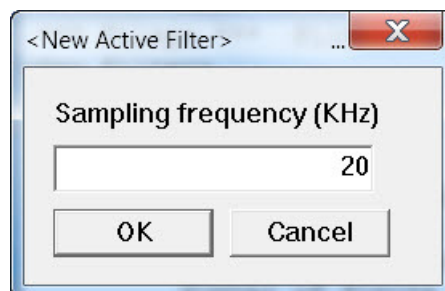


Fig. 9 Second order switched capacitor section

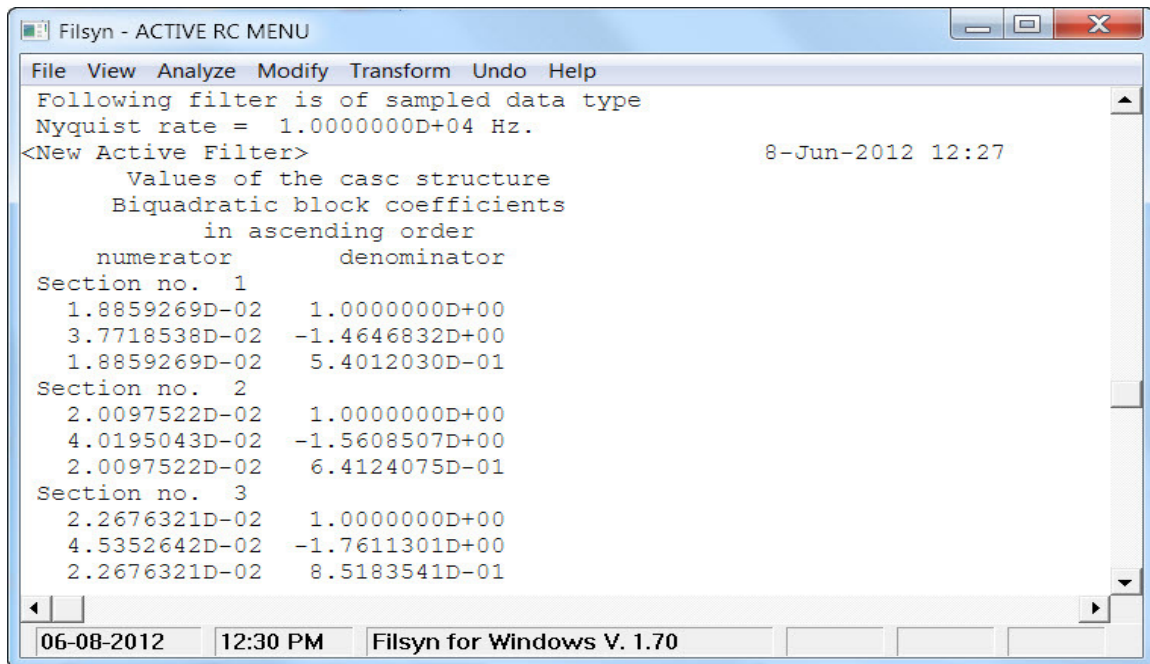
If, however, the implementation was not one of the feedback types, but the straightforward cascade of second order sections, we have an additional option. Consider, for example, a simple 6th order Butterworth lowpass:



We convert this to a switched-C design by using the z-transform option again:

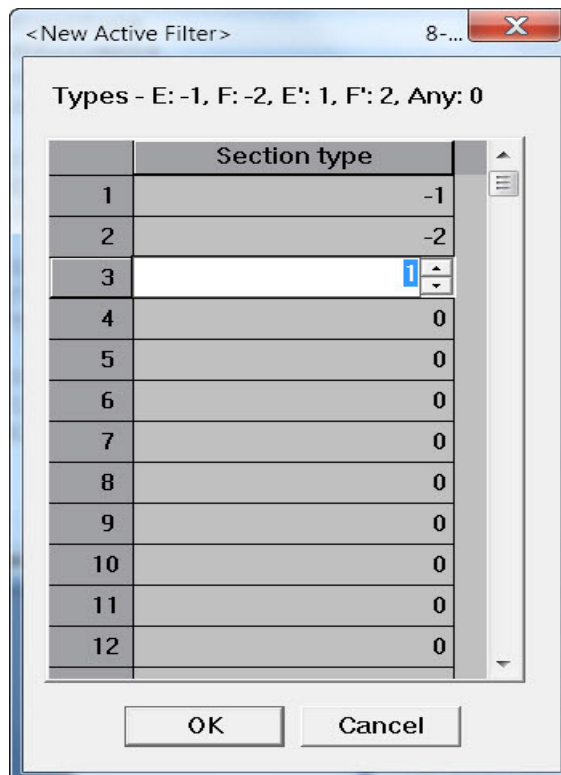


The results are shown below:

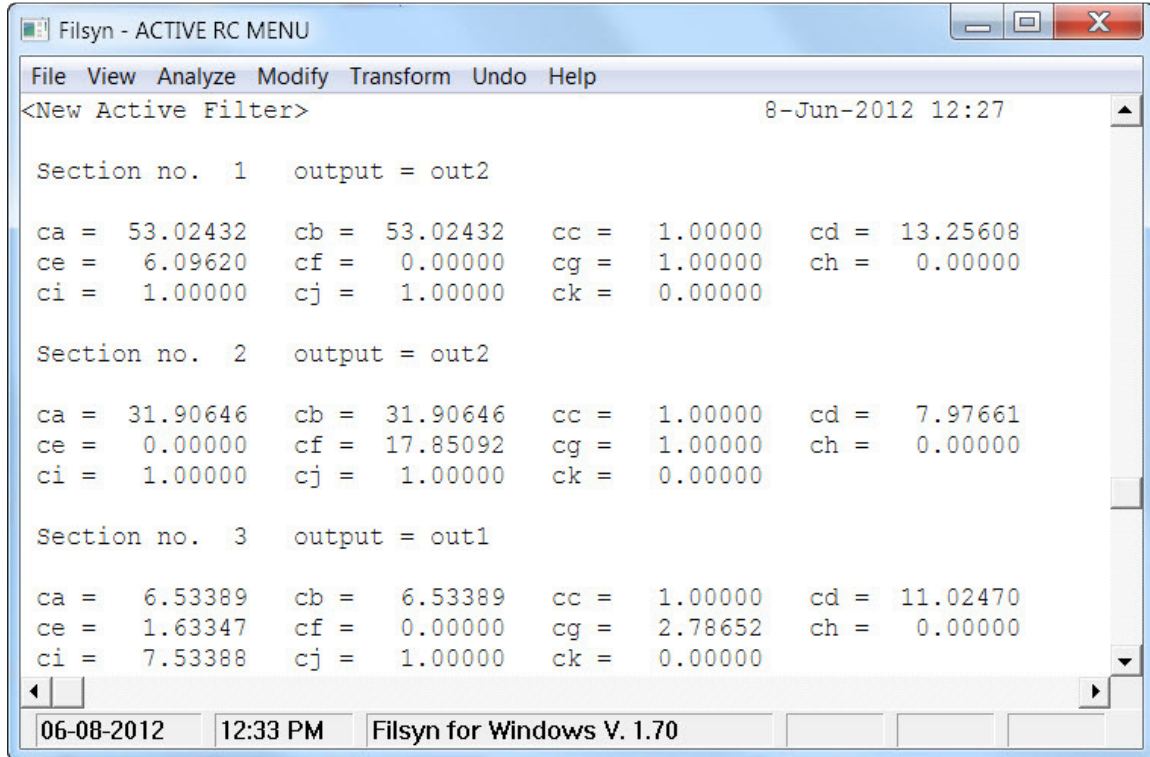


Now we are ready to implement this filter using the **View->Design** menu item, which brings up a new menu. Here we can select one of the four implementation options described in the Fleischer-Laker paper quoted above for each of the three sections. The four options are E, F, E' or F'. Selecting **Any** will make the program select one for you.

Here we selected E, F and E' for the three sections respectively:



The final design is shown below:



2. Going back to the various menu options, the **Finite Q** submenu option is useful for FLF or LF feedback circuits. As mentioned above, these structures contain second order sections with poles on the imaginary axis (with the exceptions of the first and possibly last sections), which are therefore nominally unstable, even though the overall circuit is stable. In order to ease the problem of tuning these sections by themselves, we can replace their infinite Q values by a finite, but preferably high, value.

3. The **Scale frequency** option is simply a frequency scaling operation, while the

4. **Magnitude scale** item performs an automatic section-by-section magnitude scaling operation that optimizes the dynamic ranges of all of them. This operation does *not* work on leapfrog feedback circuits.

5. Finally, the **Equalize** option invokes the usual delay equalizer subprogram.

6.7 The 'Undo' menu option

The next menu item, **Undo** again contains the usual **Undo** and **Redo** submenu options. We can undo the last ten operations in reverse sequence and redo any of them except the first that was undone.

6.8 The 'Help' menu option

The last item, **Help** contains the usual options.

7. IIR DIGITAL DESIGN

This chapter describes the design of IIR digital filters and includes a number of examples to clarify matters.

Filsyn is capable of designing infinite-impulse-response (IIR) type digital filters in a number of different ways. The resulting transfer function $H(z)$ will be a rational fractional function (a ratio of two polynomials) of the variable z^{-1} . The only restriction is that the numerator may be of no higher degree than the denominator.

7.1 Available transformations

The design procedure starts by finding an appropriate analog transfer function (a function of the complex frequency variable s) and then converts it into a function of z^{-1} by the use of any of the three commonly used transformations:

- impulse-invariant (“standard”) Z-transformation;
- matched Z-transformation;
- bilinear Z-transformation.

It is not appropriate to discuss the respective properties or merits of these transformations in this manual. However, it should be stated that only the bilinear Z-transformation can be expected to preserve unconditionally the filtering properties of the transfer function. This is done at the cost of a distortion (*warping*) of the frequency scale. The program will compensate for this by *prewarping* the requirements. This operation is useful even if other than the bilinear Z-transform is used and is therefore applied in all cases.

Once all requirements are entered, the program will perform the prewarping and calculate the proper initial analog transfer function. Subsequently, the digital synthesis segment will be entered automatically and one can then select the desired Z-transformation to be performed.

The resulting digital transfer function will be presented initially in factored form, convenient for the cascaded biquadratic form implementation of the filter. Additional forms are also available through the use of the appropriate menu items in the analysis segment (see below). In the impulse-invariant case the parallel form is also computed, because it is needed for the actual Z-transform computations.

At this stage, if the filter is a lowpass, we may apply a spectral (frequency) transformation to it. The available options are:

- lowpass-to-lowpass (rescaling to a new cutoff frequency with the same sampling rate),
- lowpass-to-highpass,
- lowpass-to-bandpass and
- lowpass-to-band reject.

The last option is unique. It is the only way to obtain a digital band-reject filter. The others, except for their inherent ability to change the apparent sampling rate, are either inefficient compared to a direct highpass or bandpass design, or completely equivalent to it. Hence these transformations are included here mainly for completeness.

Concerning the possible implementation structures, we have the following options:

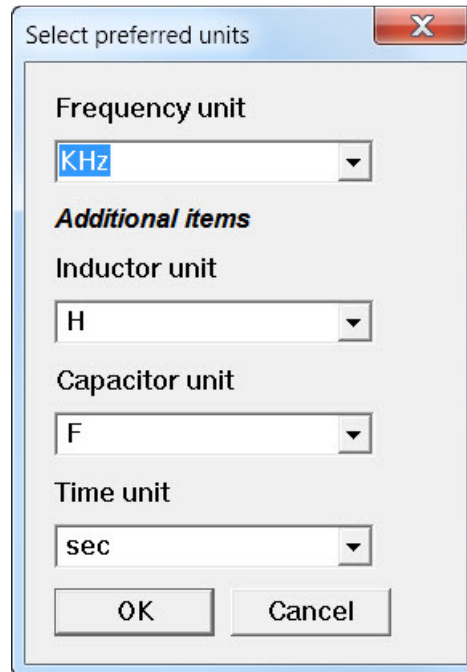
- Cascaded second order sections
- Parallel second order sections
- Direct form
- Gray-Markel lattice (2-multiplier form)
- Differential allpass form (when it exists)

We will find the cascade form upon entering this segment and if we need them, the others are obtainable by the use of the appropriate menu option.

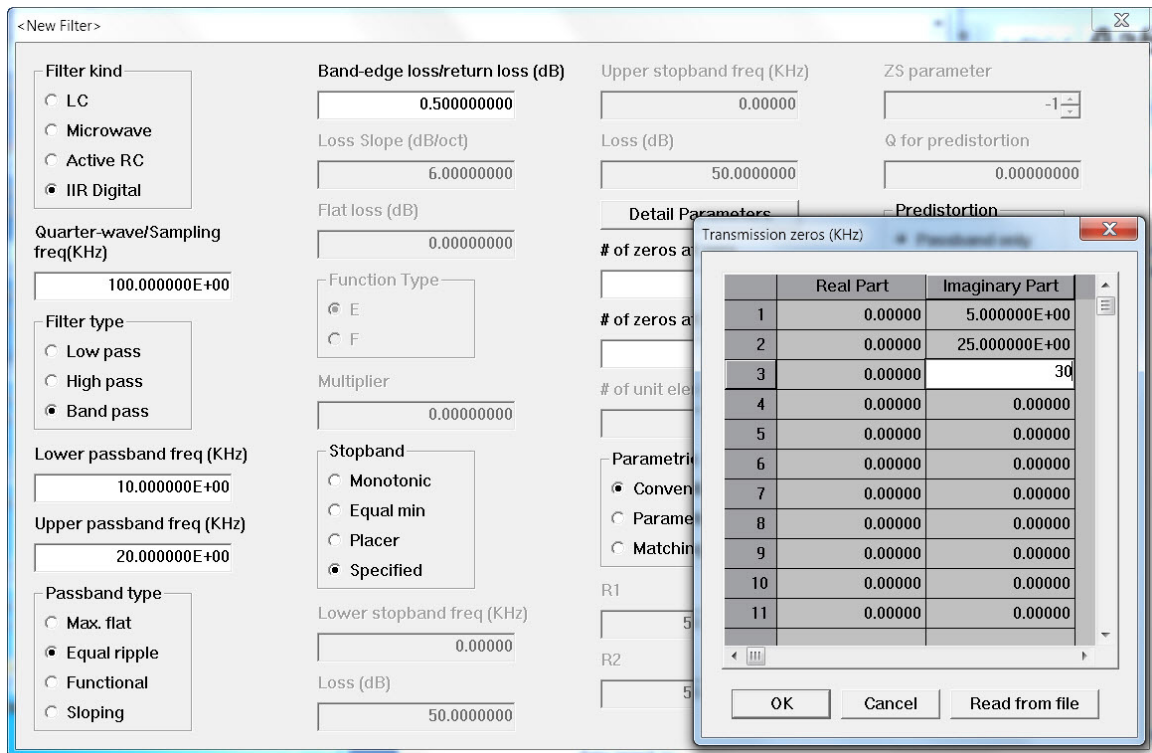
Various **wave-digital** forms are also available through the passive segment of the program. These will be demonstrated at the end of this chapter.

7.2 Example 1: Bilinear Bandpass Filter

As was done in the preceding chapters, the design of IIR digital filters will be explained by the use of examples. First we select KHz as the global unit using the **File -> Global unit** menu item:

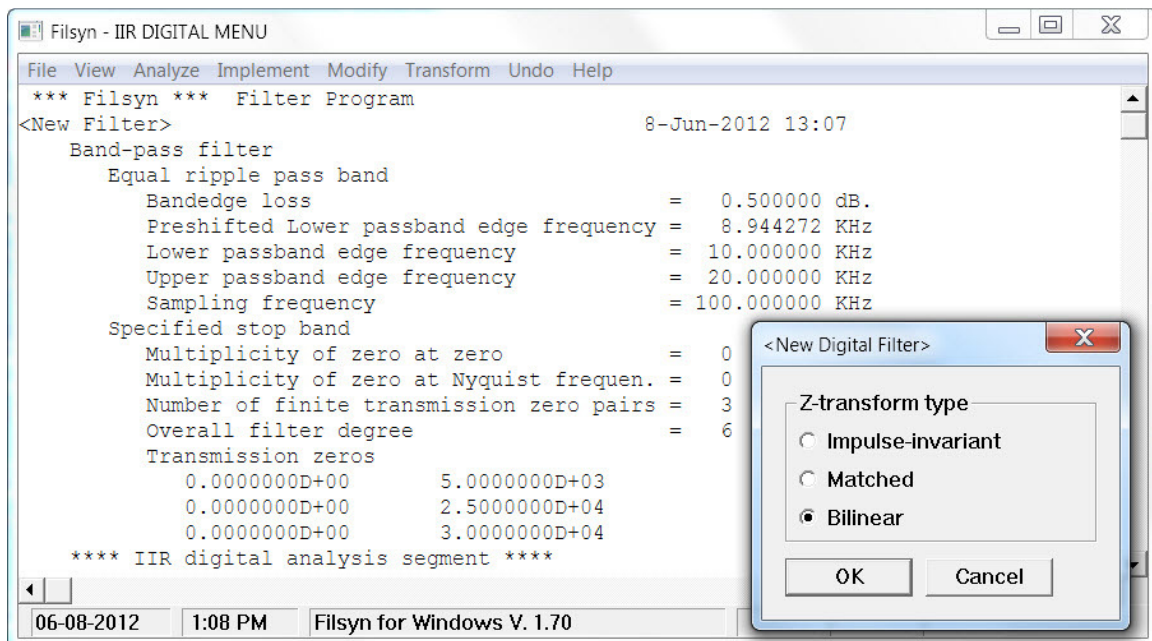


We will design a bilinear Z-transformed bandpass with specified transmission zeros. The data input is:

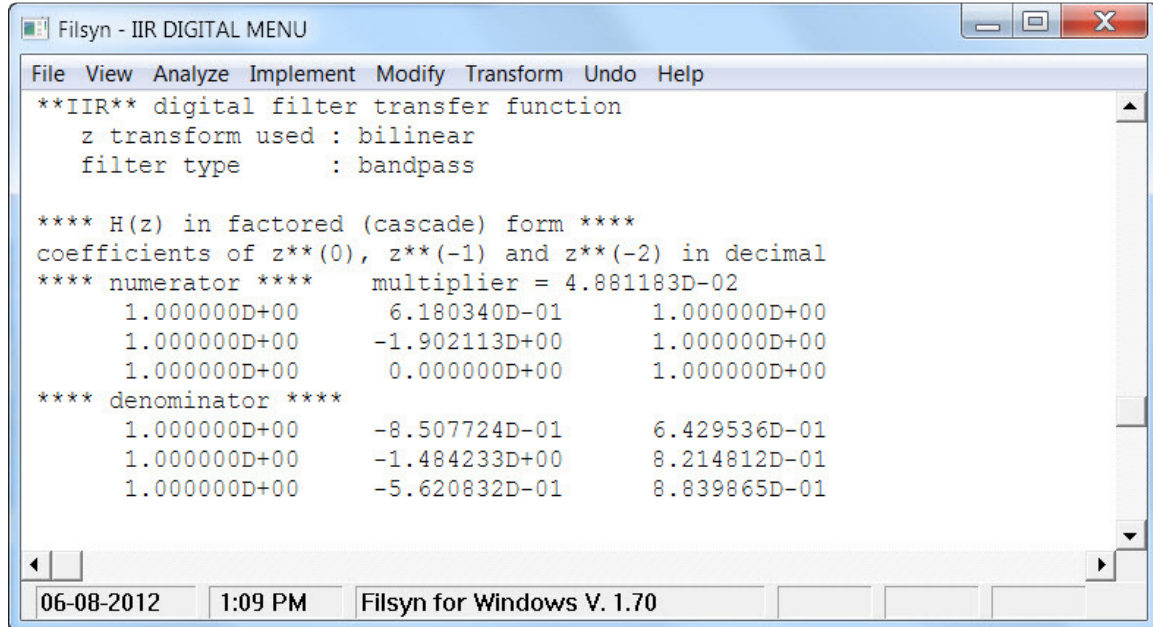


While **parametric** bandpass is available, it does not have the advantage over that of the **conventional** type, which it had in the passive LC case. Since terminations are meaningless for digital filters, those items are blanked out.

The summary presented below shows the *preshifted passband edge* as used by the program for prewarping. Otherwise the listing is a summary of what we entered:



Selecting the default **Bilinear** case, we get the design:



```

Filsyn - IIR DIGITAL MENU
File View Analyze Implement Modify Transform Undo Help
**IIR** digital filter transfer function
  z transform used : bilinear
  filter type      : bandpass

**** H(z) in factored (cascade) form ****
coefficients of z**(0), z**(-1) and z**(-2) in decimal
**** numerator ****      multiplier = 4.881183D-02
    1.000000D+00      6.180340D-01      1.000000D+00
    1.000000D+00      -1.902113D+00      1.000000D+00
    1.000000D+00      0.000000D+00      1.000000D+00
**** denominator ****
    1.000000D+00      -8.507724D-01      6.429536D-01
    1.000000D+00      -1.484233D+00      8.214812D-01
    1.000000D+00      -5.620832D-01      8.839865D-01
  
```

06-08-2012 1:09 PM Filsyn for Windows V. 1.70

The function is in factored form which is appropriate for cascade implementation. Other implementations are available, but we will consider them in the Digital Analysis segment.

7.3 Example 2: Band Elimination Filter

Since we must start our design with a lowpass specification, it is imperative that we convert the requirements into lowpass form. These expressions are given here without derivation.

Parenthetically, we note here that the **Elliptic** design in the **filscript.exe** script processor will do all these computations for us automatically. Here we did these calculations to describe the process:

First we introduce the values:

$$C_A = \cos [\pi(f_B + f_A)/f_S] / \cos [\pi(f_B - f_A)/f_S]$$

where f_A and f_B are the passband edge frequencies of the band-reject filter and f_S is the sampling frequency, and:

$$C_B = \tan [\pi(f_B - f_A)/f_S] * \tan (\pi F_B/f_S)$$

where F_B is the (arbitrary) passband edge of the reference lowpass filter. For the stopband edge frequencies, we use the expression:

$$F_{\text{stop}} = (f_S/\pi) * \tan^{-1} [C_B * \sin(2\pi f_{\text{stop}}/f_S) / (\cos(2\pi f_{\text{stop}}/f_S) - C_A)]$$

Here f_{stop} is the stopband edge frequency of the band elimination filter and F_{stop} is the corresponding value for the reference lowpass. There are two stopband edges for the band elimination filter and we use the smaller value of the two F_{stop} values for the design of the reference lowpass.

As an example, consider the band-reject filter specification of sampling frequency of 10 KHz, passbands below 1 KHz and above 2 KHz with 0.1 dB loss ripple and stopband between 1050 Hz and 1950 Hz with a minimum of 40 dB loss.

Selecting $F_B = 1$ KHz, we compute:

$$C_A = 0.618034$$

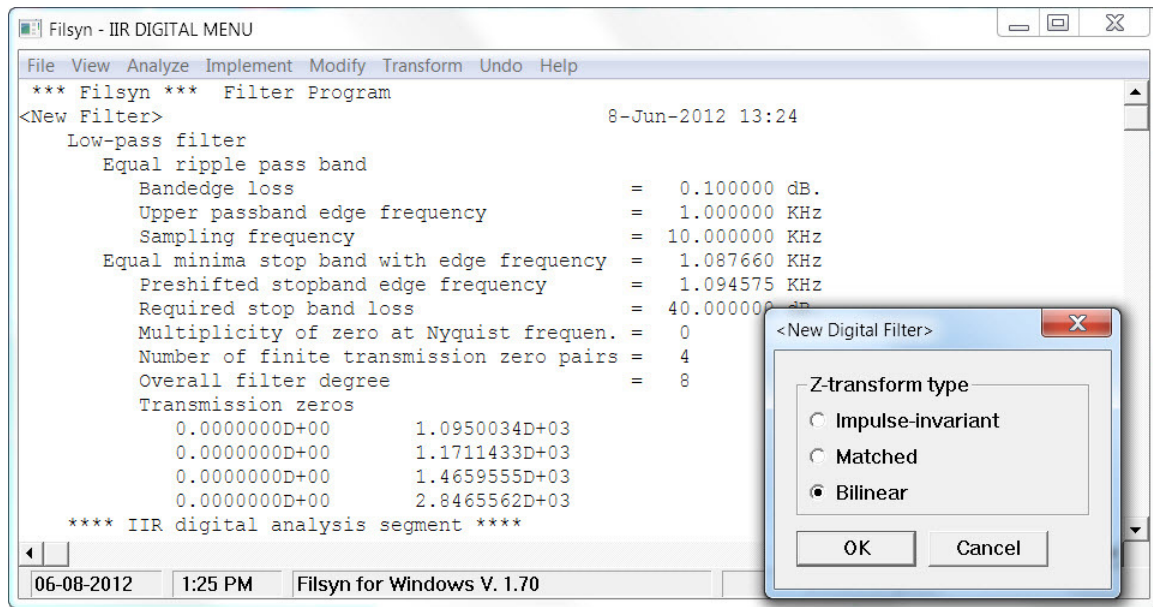
$$C_B = 0.105573$$

The lower value of the stopband frequencies is the one that corresponds to 1950 Hz:

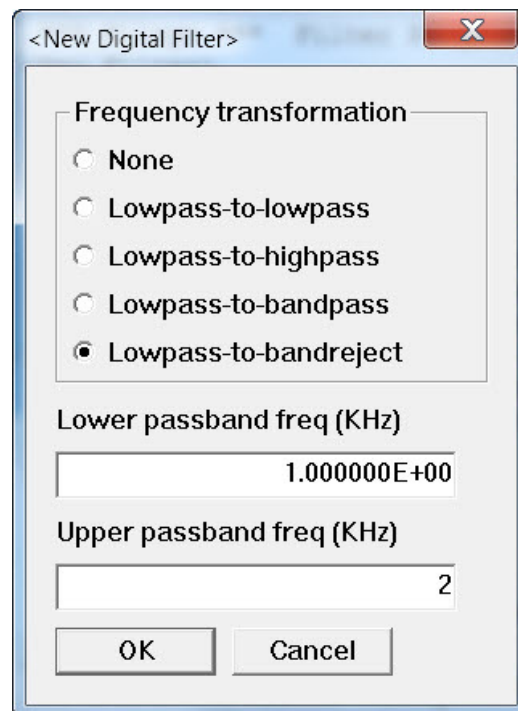
$$F_{\text{stop}} = 1087.66 \text{ Hz}$$

The input data screen is shown next:

Clicking on **OK**, we see that we need an 8th order lowpass. First we select the usual bilinear transform:



which we now convert to a 16th order band reject filter:



Yielding:

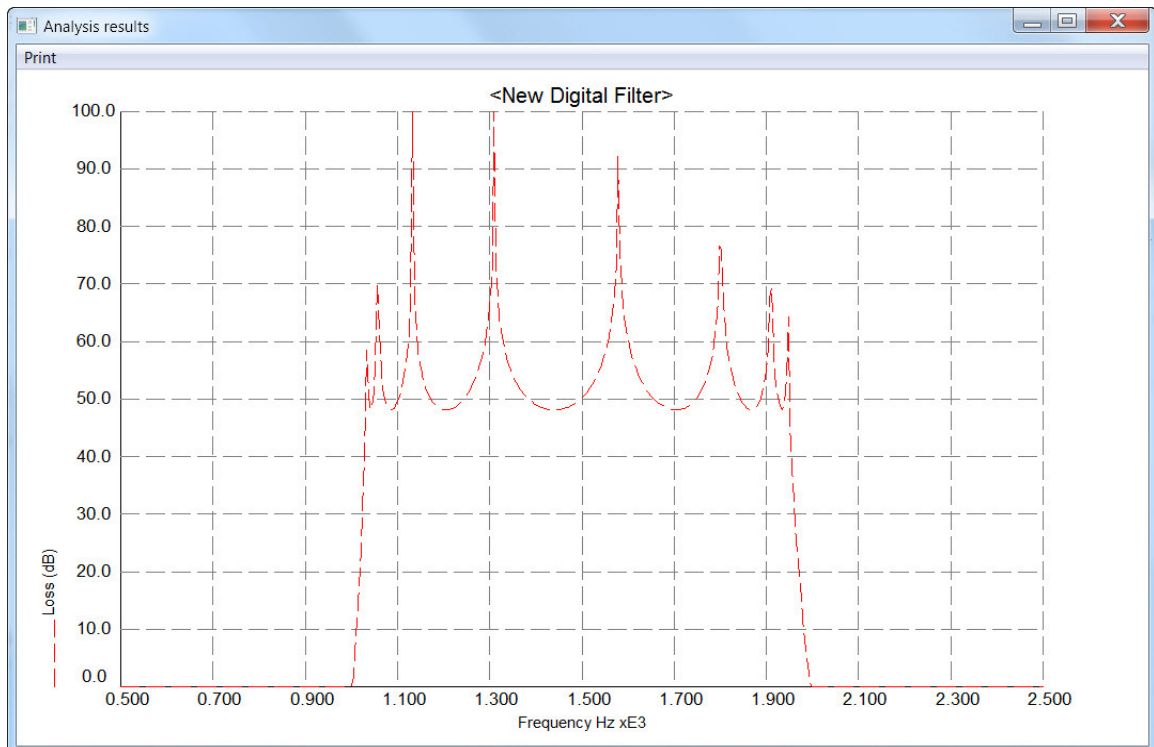
```

Filsyn - IIR DIGITAL MENU
File View Analyze Implement Modify Transform Undo Help
**IIR** digital filter transfer function
  z transform used : bilinear
  filter type      : band reject
  (obtained from digital lowpass by spectral transformation)
    lower passband edge frequency      = 1.000000 KHz
    upper passband edge frequency      = 2.000000 KHz

**** H(z) in factored (cascade) form ****
coefficients of z**(0), z**(-1) and z**(-2) in decimal
**** numerator ****      multiplier = 3.064393D-01
  1.000000D+00   -1.360346D+00   1.000000D+00
  1.000000D+00   -1.094153D+00   1.000000D+00
  1.000000D+00   -1.514124D+00   1.000000D+00
  1.000000D+00   -8.509198D-01   1.000000D+00
  1.000000D+00   -1.574003D+00   1.000000D+00
  1.000000D+00   -7.256430D-01   1.000000D+00
  1.000000D+00   -1.592590D+00   1.000000D+00
  1.000000D+00   -6.820143D-01   1.000000D+00
**** denominator ****
  1.000000D+00   -1.296059D+00   5.591375D-01
  1.000000D+00   -3.298671D-01   3.395727D-01
  1.000000D+00   -1.543065D+00   8.638505D-01
  1.000000D+00   -4.554469D-01   7.763854D-01
  1.000000D+00   -1.591310D+00   9.582961D-01
  1.000000D+00   -5.788767D-01   9.326251D-01
  1.000000D+00   -1.606851D+00   9.905861D-01
06-08-2012   1:28 PM   Filsyn for Windows V. 1.70

```

The performance is excellent, the minimum stopband loss is about 48 dB everywhere:



7.4 Wave Digital Filters

An entirely different set of implementations of IIR digital filters is available through the passive synthesis segment of the program. This is based on an initial design that is a microwave filter. This filter can then be converted into an IIR digital filter, essentially using the bilinear Z-transform, and consequently preserving the filtering properties of the structure. Prewarping is automatically performed in the design of the microwave filter, hence we need to pay no attention to it.

The conversion from microwave lattice or ladder structure to a digital filter is based upon the *voltage-wave* method, described by Dr. Fettweis in his exhaustive review of the subject (see ref. [12]). Without going into any details, the design procedure is as follows:

- a. We design the filter as a microwave one, where we substitute the Nyquist frequency (half the sampling frequency) every time the program asks for the “quarter-wave” frequency. The filter may be arbitrary and may contain unit elements, but no delay equalizer sections. It may have complex transmission zeros, but if it does, it may *not* have unit elements.
- b. If the filter has no unit elements and is symmetrical, i.e. the multiplicities of extreme transmission zeros, if nonzero, are both odd), we may ask for the lattice implementation and select the wave-digital form of this lattice, when offered.
- c. If the filter has no complex transmission zeros, we can obtain a ladder implementation of the filter, rearrange it in any way we wish, using the tools of the passive analysis segment. When we are satisfied with the results, use the **Transform->Wave** menu option to convert the filter into a wave-digital ladder.

All these structures will contain unit delay lines (shift registers), shown with the abbreviation ***T***, and two- and three-port parallel and series adaptors. These adaptors consist of adders and multipliers, and are clearly described in the reference quoted above; but for completeness, we show the simplest implementations of all we need below.

First the two-port adaptor implementation is shown in Fig. 10 below. The adaptor needs a single multiplier and three adders. Denoting the *terminating impedances* of the respective ports by $R(1)$ and $R(2)$, the multiplier constant is given by:

$$A = (R(1) - R(2)) / (R(1) + R(2))$$

For two-port adaptors, there is no difference between parallel and series adaptors. Next we show the general forms of the parallel and series three-port adaptors. Both of these require two multipliers and four adders with a sign change or two (see Figs. 2 and 3). Denoting the respective *terminating impedances* by $R(1)$, $R(2)$ and $R(3)$, the multiplier constants for the series adaptor are given by:

$$\begin{aligned} A &= 2 R(1) / (R(1) + R(2) + R(3)) \\ B &= 2 R(2) / (R(1) + R(2) + R(3)) \end{aligned}$$

For the parallel three-port adaptor, shown below, we calculate the multiplier constants from the *terminating admittances* $G(1)$, $G(2)$ and $G(3)$ as follows:

$$A = 2 G(1)/(G(1) + G(2) + G(3))$$

$$B = 2 G(2)/(G(1) + G(2) + G(3))$$

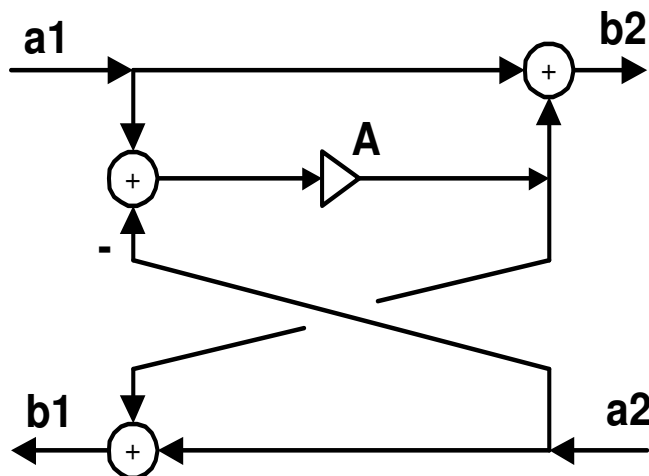


Fig. 10 Two-port adaptor

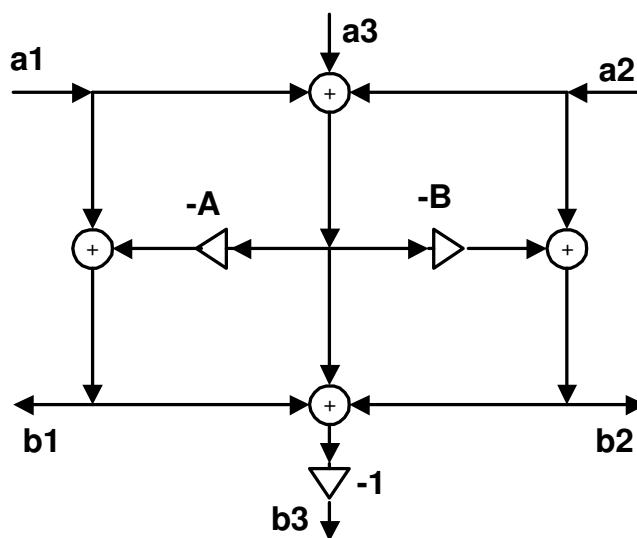


Fig. 11 Series three-port adaptor

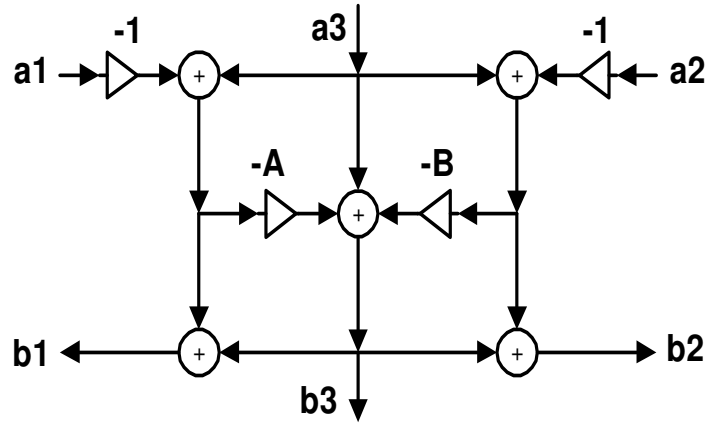


Fig. 12 Parallel three-port adaptor

In a number of cases, one of the three impedances or admittances may be selected arbitrarily. If, in such a case, we select $R(3)$ for the series adaptor as:

$$R(3) = R(1) + R(2)$$

or $G(3)$ for the parallel adaptor as:

$$G(3) = G(1) + G(2)$$

then these adaptors may be simplified and can be implemented by using only one multiplier per adaptor. Since the conversion procedure selects this option whenever possible, we show the corresponding simplified implementations as follows:

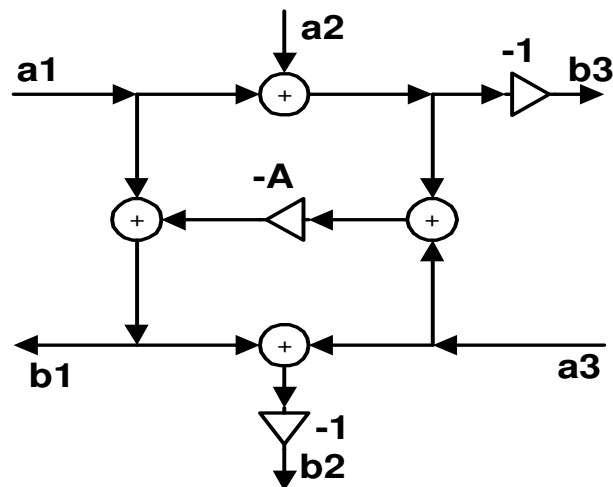


Fig. 13 Restricted three-port series adaptor

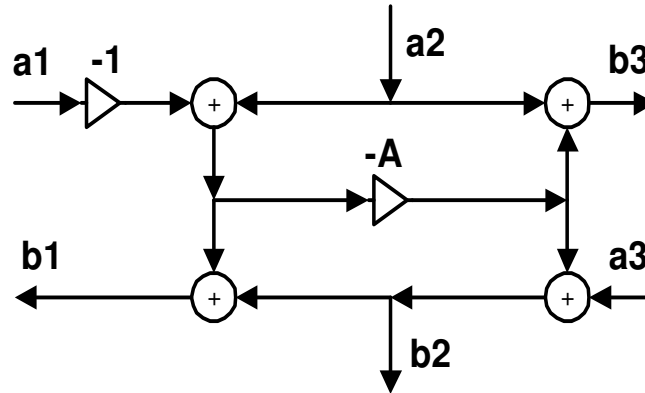


Fig. 14 Restricted three-port parallel adaptor

In these special cases, the reflected signal b_3 at port 3 is independent of the incident signal a_3 at the same port, hence we designate this case with the phrase: **port 3 matched** in the display. If we see the **port 1 matched** legend instead, we must naturally interchange ports 1 and 3.

All of the computations of the multiplier constants are performed automatically by the program. The quantities printed are the multipliers (A and B above) needed to implement the adaptors. A few examples will serve to illustrate the procedure as well as the resulting structures.

7.5 Example 3: Wave-Digital Lattice.

We consider a simple microwave elliptic lowpass filter of degree 5 with the following specification. The 36 kHz quarter-wave frequency will represent a 72 kHz sampling frequency after conversion:

<New Filter>

Filter kind

- ☐ LC
- ☒ Microwave
- ☐ Active RC
- ☐ IIR Digital

Quarter-wave/Sampling freq(KHz)

36.000000E+00

Filter type

- ☒ Low pass
- ☐ High pass
- ☐ Band pass

Lower passband freq (KHz)

0.00000

Upper passband freq (KHz)

3.000000E+00

Passband type

- ☐ Max. flat
- ☒ Equal ripple
- ☐ Functional
- ☐ Sloping

Band-edge loss/return loss (dB)

0.100000000

Loss Slope (dB/oct)

6.00000000

Flat loss (dB)

0.00000000

Function Type

- ☒ E
- ☐ F

Multiplier

0.00000000

Stopband

- ☐ Monotonic
- ☒ Equal min
- ☐ Placer
- ☐ Specified

Lower stopband freq (KHz)

0.00000

Loss (dB)

50.0000000

Upper stopband freq (KHz)

3.500000E+00

Loss (dB)

25

Detail Parameters

of zeros at zero

0

of zeros at FQ or FS/2

0

of unit elements

0

Parametric

- ☒ Conventional
- ☐ Parametric
- ☐ Matching

R1

50.000000E+00

R2

50.000000E+00

ZS parameter

-1

Q for predistortion

0.00000000

Predistortion

- ☒ Passband only
- ☐ Pass/stop bands

Shifted bandpass trans.
Center frequency (KHz)

0.00000

☐ Odd parametric

☐ Save design data

OK Cancel

The next step is selecting the **Lattice** option in the **Synthesis** menu, followed by the **Wave digital** option:

<Synthesis>

Structure

- ☒ Lattice
- ☐ Ladder

Configuration

- ☒ Auto
- ☐ Manual

Numerator

- ☒ Even sum
- ☐ Even difference

Form

- ☐ Microwave
- ☒ Wave digital

Direction

- ☒ Forward
- ☐ Reverse

Denominator

- ☒ Odd sum
- ☐ Odd difference

Format

- ☒ Decimal
- ☐ Hexadecimal

Impedance selection

- ☒ Auto
- ☐ Manual

This is an

- ☒ Impedance
- ☐ Admittance

Existing polynomials: es, os, od

☐ Print polynomials

OK Cancel

The final results are shown next:

```

Filsyn - LATTICE MENU
File View Analyze Help
<New Lattice Filter> 8-Jun-2012 13:48
*** Lattice wave-digital implementation ***

Realization of lattice branch no. 1
*t* indicates unit delay

2-port adaptor
a= 7.781103D-01
*t*

2-port adaptor
a=-9.296490D-01
*t*

2-port adaptor
a= 3.779702D-01
*t* (-1)

Realization of lattice branch no. 2
*t* indicates unit delay

2-port adaptor
a= 8.255646D-01
*t*

2-port adaptor
a=-7.094993D-01
*t*

06-08-2012 1:48 PM Filsyn for Windows V. 1.70

```

The numerical data can be printed either in decimal or in hexadecimal format, the latter providing more precision, if needed. These two lattice branches contain 3 and 2 (two-port) adaptors and the same number of delay lines respectively in the following configuration (the boxes represent the two-port adaptors). Note the sign inversion at the end of branch 1 (see Fig. 15). The complete lattice consists of these two branches in the configuration of Fig. 16. This needs an additional summer and a divide-by-two circuit.

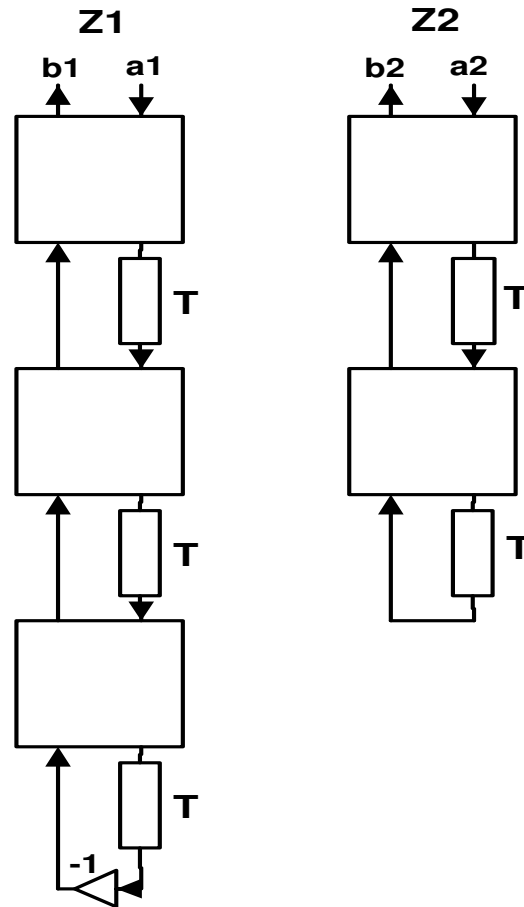


Fig. 15 Lattice branches

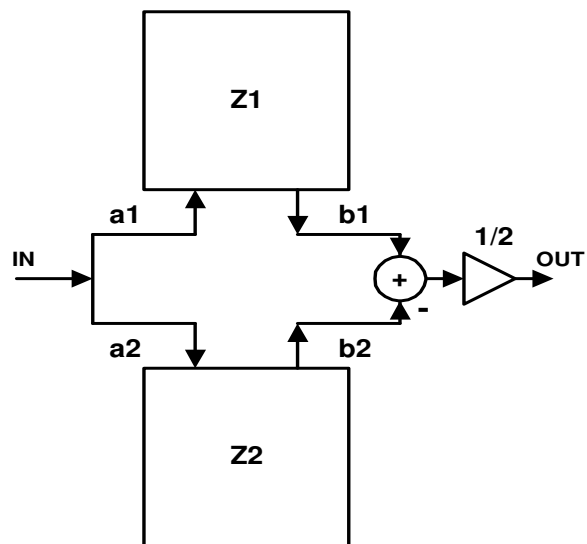


Fig. 16 Complete digital lattice

7.6 Example 4: Wave-Digital Ladder

The next example is a simple microwave bandpass filter with the following specifications:

<New Filter>

Filter kind

- ☐ LC
- ☒ Microwave
- ☐ Active RC
- ☐ IIR Digital

Quarter-wave/Sampling freq(KHz)

36.000000E+00

Filter type

- ☐ Low pass
- ☐ High pass
- ☒ Band pass

Lower passband freq (KHz)

2.000000E+00

Upper passband freq (KHz)

3.000000E+00

Passband type

- ☐ Max. flat
- ☒ Equal ripple
- ☐ Functional
- ☐ Sloping

Band-edge loss/return loss (dB)

0.500000000

Loss Slope (dB/oct)

6.00000000

Flat loss (dB)

0.00000000

Function Type

- ☒ E
- ☐ F

Multiplier

0.00000000

Stopband

- ☐ Monotonic
- ☐ Equal min
- ☐ Placer
- ☒ Specified

Lower stopband freq (KHz)

0.000000

Loss (dB)

50.0000000

Upper stopband freq (KHz)

0.000000

Loss (dB)

50.0000000

Detail Parameters

of zeros at zero

1

of zeros at FQ or FS/2

1

of unit elements

0

Parametric

- ☒ Conventional
- ☐ Parametric
- ☐ Matching

R1

50.000000E+00

R2

50.000000E+00

ZS parameter

-1

Q for predistortion

0.00000000

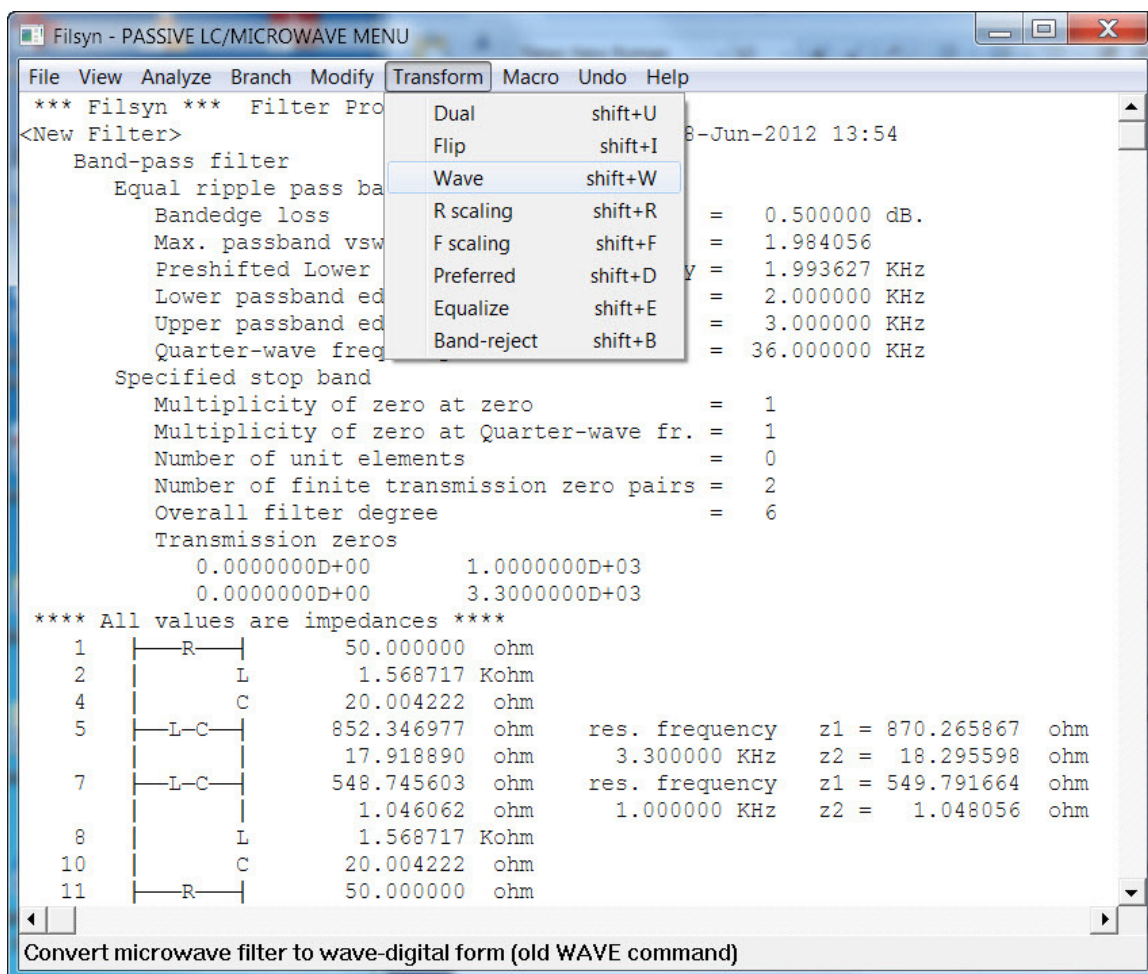
Predistortion

Transmission zeros (KHz)

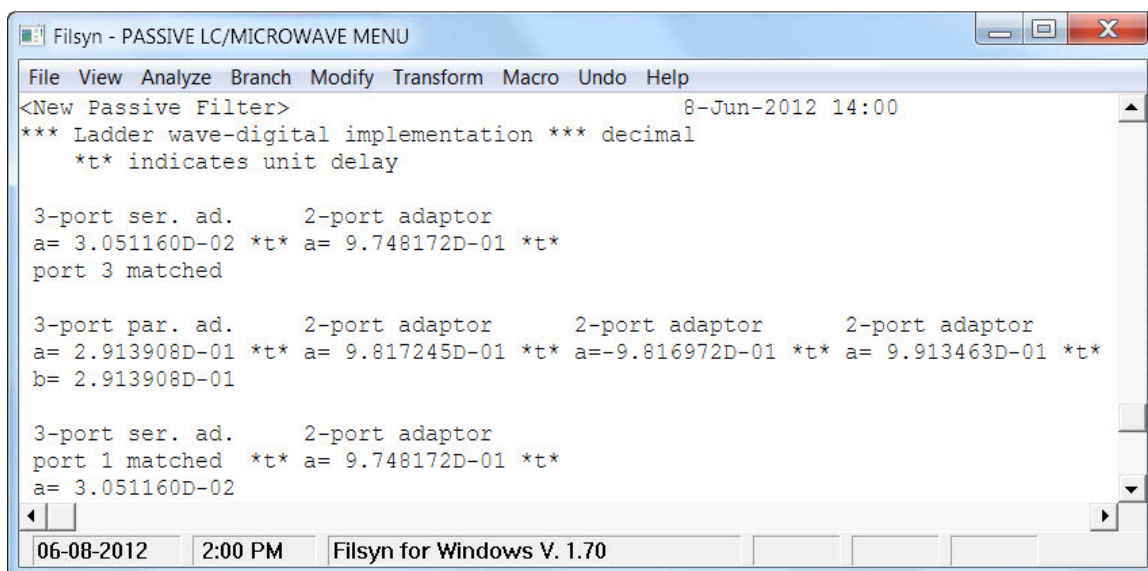
	Real Part	Imaginary Part
1	0.00000	1.000000E+00
2	0.00000	3.3
3	0.00000	0.00000
4	0.00000	0.00000
5	0.00000	0.00000
6	0.00000	0.00000
7	0.00000	0.00000
8	0.00000	0.00000
9	0.00000	0.00000
10	0.00000	0.00000
11	0.00000	0.00000

OK Cancel Read from file

We used 36 kHz quarter-wave frequency again to yield 72 kHz sampling rate. This time we selected the ladder realization of this filter:



This is followed by the application of the **Transform->Wave** menu option to convert the filter to the ladder wave-digital form:



Again we can have the coefficients printed in decimal or hexadecimal forms. To print the other form, just repeat the menu selection and select the hexadecimal print format:

```

Filsyn - PASSIVE LC/MICROWAVE MENU
File View Analyze Branch Modify Transform Macro Undo Help
<New Passive Filter> 8-Jun-2012 13:56
*** Ladder wave-digital implementation *** hexadecimal
    ** indicates unit delay

3-port ser. ad.      2-port adaptor
a= 0.07cf9b      ** a= 0.f98d9e      **
port 3 matched

3-port par. ad.      2-port adaptor      2-port adaptor      2-port adaptor
a= 0.4a9896      ** a= 0.fb524c      ** a=-0.fb5081      ** a= 0.fdc8de      **
b= 0.4a9896

3-port ser. ad.      2-port adaptor
port 1 matched      ** a= 0.f98d9e      **
a= 0.07cf9b

06-08-2012 1:59 PM Filsyn for Windows V. 1.70
  
```

The implementation needs 5 two-port adaptors and 3 three-port adaptors, two of which use the restricted form, since one of their ports is matched. The structure is shown graphically below:

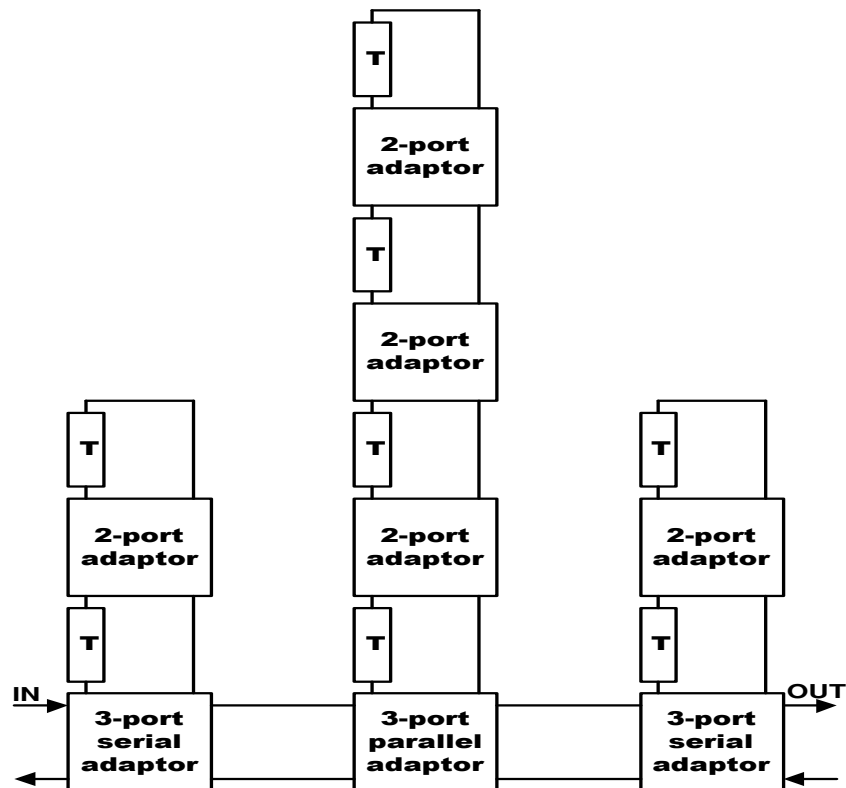
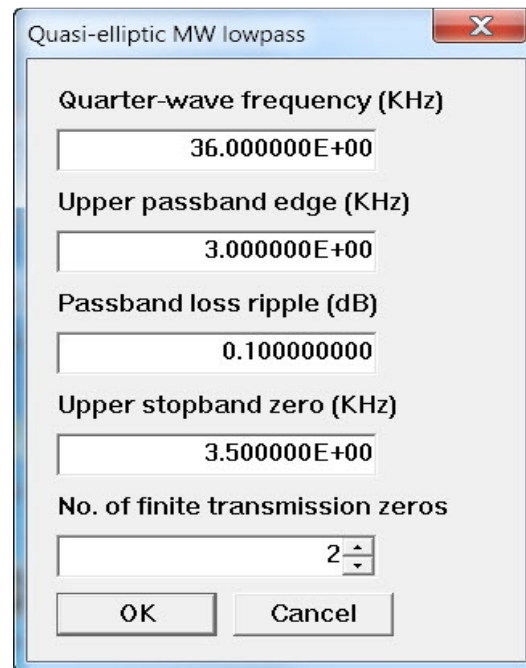


Fig. 17 Ladder wave-digital filter structure

7.7 Example 5: Wave-Digital Ladder with Unit Elements

This example is a bit more elaborate, obtained through the use of the **filscript.exe** script processor, where we used the **Microwave low-, high-passes->Quasi-elliptic lowpass filter** menu item from its **Start menu**. The specification is as follows:



Quasi-elliptic MW lowpass

Quarter-wave frequency (KHz)
36.000000E+00

Upper passband edge (KHz)
3.000000E+00

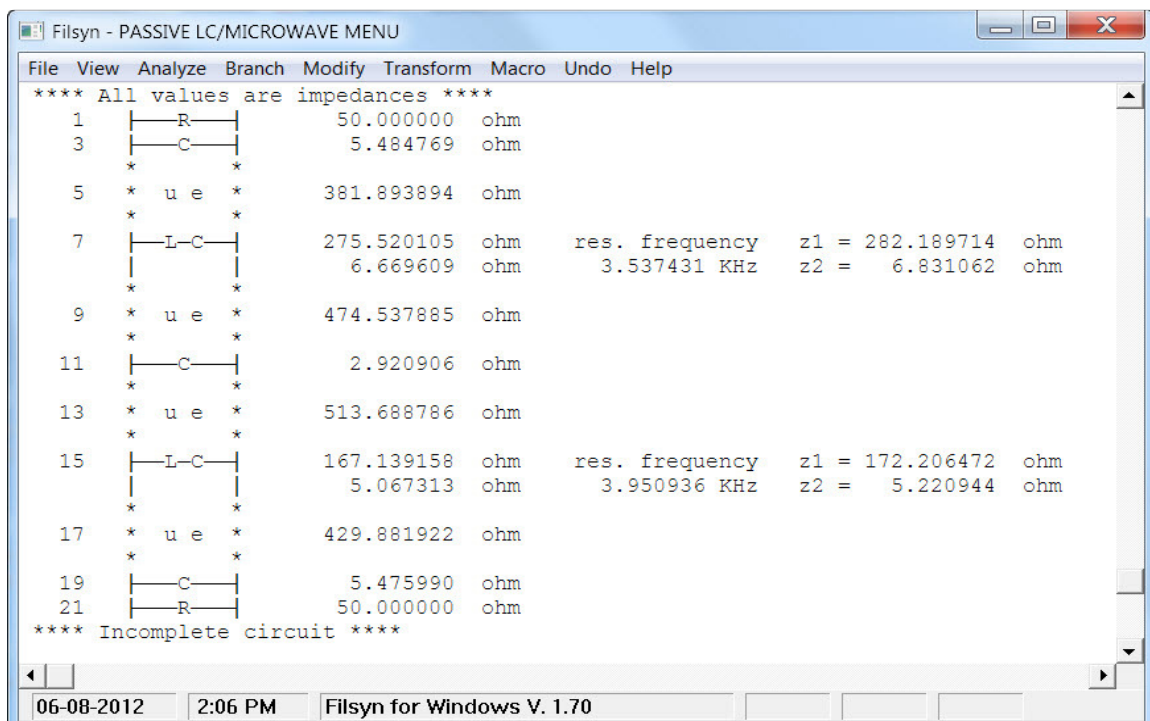
Passband loss ripple (dB)
0.100000000

Upper stopband zero (KHz)
3.500000E+00

No. of finite transmission zeros
2

OK Cancel

The resulting circuit, after a large number of steps of automatic circuit modifications:



Filsyn - PASSIVE LC/MICROWAVE MENU

File View Analyze Branch Modify Transform Macro Undo Help

**** All values are impedances ****

1	—R—	50.000000	ohm			
3	—C—	5.484769	ohm			
5	* u e *	381.893894	ohm			
7	—L—C—	275.520105	ohm	res. frequency	z1 = 282.189714	ohm
		6.669609	ohm	3.537431 KHz	z2 = 6.831062	ohm
9	* u e *	474.537885	ohm			
11	—C—	2.920906	ohm			
13	* u e *	513.688786	ohm			
15	—L—C—	167.139158	ohm	res. frequency	z1 = 172.206472	ohm
		5.067313	ohm	3.950936 KHz	z2 = 5.220944	ohm
17	* u e *	429.881922	ohm			
19	—C—	5.475990	ohm			
21	—R—	50.000000	ohm			

**** Incomplete circuit ****

06-08-2012 2:06 PM Filsyn for Windows V. 1.70

Please ignore the ‘Incomplete circuit’ comment. As long as there is an output termination, the circuit is complete.

This is of course, only one of a very large number of equivalent ladder circuits, but one that is found easy to implement as a microwave filter. For a bandpass case, there are an infinite number of equivalent forms and any one of them can be selected as the starting point for digital conversion. Using the above circuit, we obtain the ladder wave-digital form of:

```

Filsyn - PASSIVE LC/MICROWAVE MENU
File View Analyze Branch Modify Transform Macro Undo Help
Quasi-elliptic microwave lowpass 8-Jun-2012 14:07
*** Ladder wave-digital implementation *** decimal
    *t* indicates unit delay

3-port par. ad.
a= 1.951776D-01 *t*
b= 2.555390D-02
    *t*

3-port par. ad.      2-port adaptor
a= 6.332935D-01 *t* a= 9.527296D-01 *t*
b= 5.096557D-01
    *t*

3-port par. ad.
a= 1.216646D-02 *t*
b= 1.123919D-02
    *t*

3-port par. ad.      2-port adaptor
a= 3.862544D-01 *t* a= 9.411484D-01 *t*
b= 4.615559D-01
    *t*

3-port par. ad.
a= 2.270130D-02 *t*
b= 1.951776D-01
    
```

06-08-2012 2:08 PM Filsyn for Windows V. 1.70

As usual, the printout suggests the structure, which is shown below:

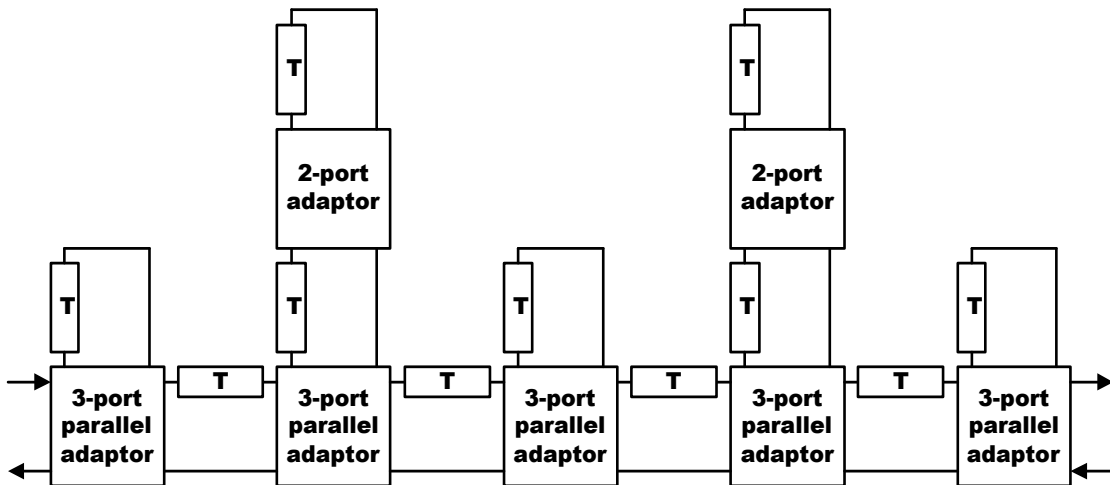


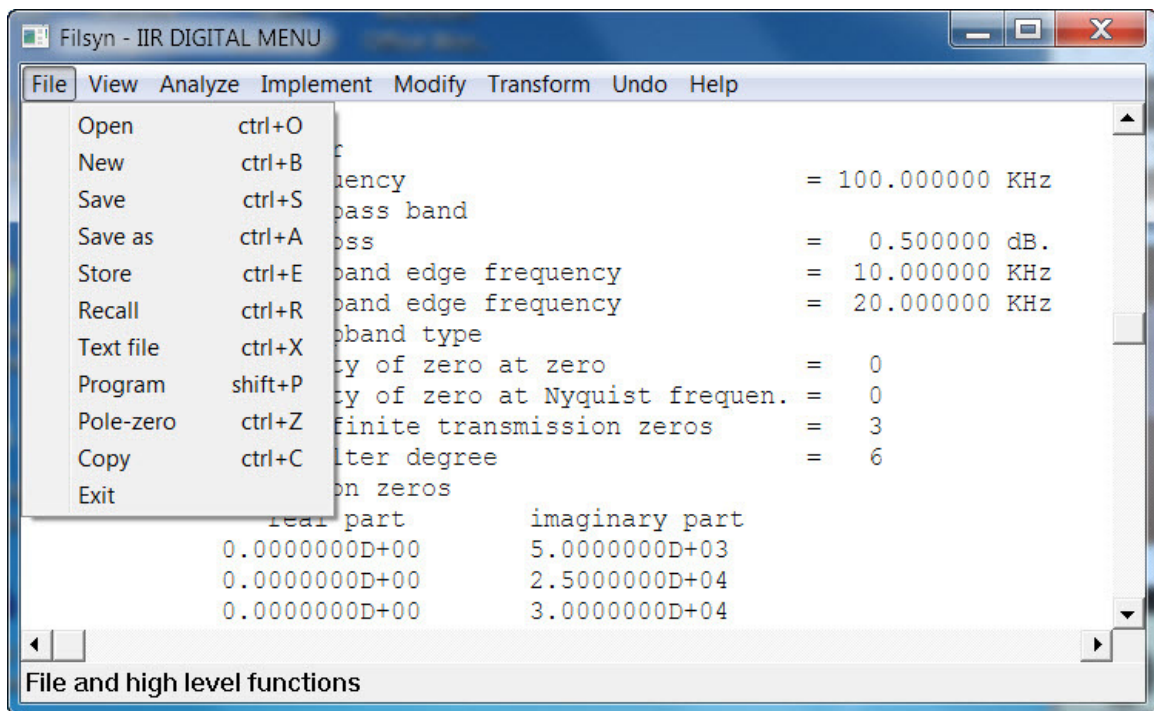
Fig. 18 Ladder wave-digital structure.

8. IIR DIGITAL ANALYSIS

In many respects, this chapter is similar to the passive analysis chapter. In the first place, it may be used to analyze and manipulate digital filters whether they were originally designed by **Filsyn** or obtained from other sources. In the second place, it uses many of the same menu items and some of the same features.

8.1 The 'File' menu option

Let us first look at the available menu options and their role in manipulating digital filters. Recalling the first example we used in the previous chapter, the **File** menu item has the following submenu options:



1. The **Open** option opens another saved data file, but the current set of data will be overwritten and before we can select the file to be opened, a warning to that effect will be shown.
2. The **New** option lets us restart the synthesis or start a completely new one without the need for stopping the program and restarting it.
3. **Save** and **Save as** will save the current set of data to the file it was read from or to a new one, as usual.
4. **Store** and **Recall** save the current set of data to a temporary file and recall it from there, respectively. This file is lost when we leave the program.

5. **Text file** writes the data to a text file in editable form for documentation purposes.
6. The **Program** option writes a Fortran program for the purpose of implementing the filter in a software format. The resulting program contains a subroutine called TIM and a function called TRUN to perform fixed point truncation or rounding of partial products and sums. The filter data is transferred to the program in 12 decimal place precision (about 40 bits binary), although they may have been truncated previously to less than that. Therefore truncation or rounding to more than 40 bits is ignored by the resulting program.

A simple calling sequence of this program might be:

```

...
IBT = 24                                ! Number of bits to be used
DO I = 1, LIMIT
  TIME = (I-1)/FS                        ! FS is the sampling frequency
  XIN = FUNCTION (TIME)                  ! This function generates the input
  CALL TIM (XIN, YOUT, I, IBT)           ! YOUT is the generated output
  WRITE (6, 100) TIME, XIN, YOUT
END DO
100  FORMAT (3x, 1P3E18.7)
...

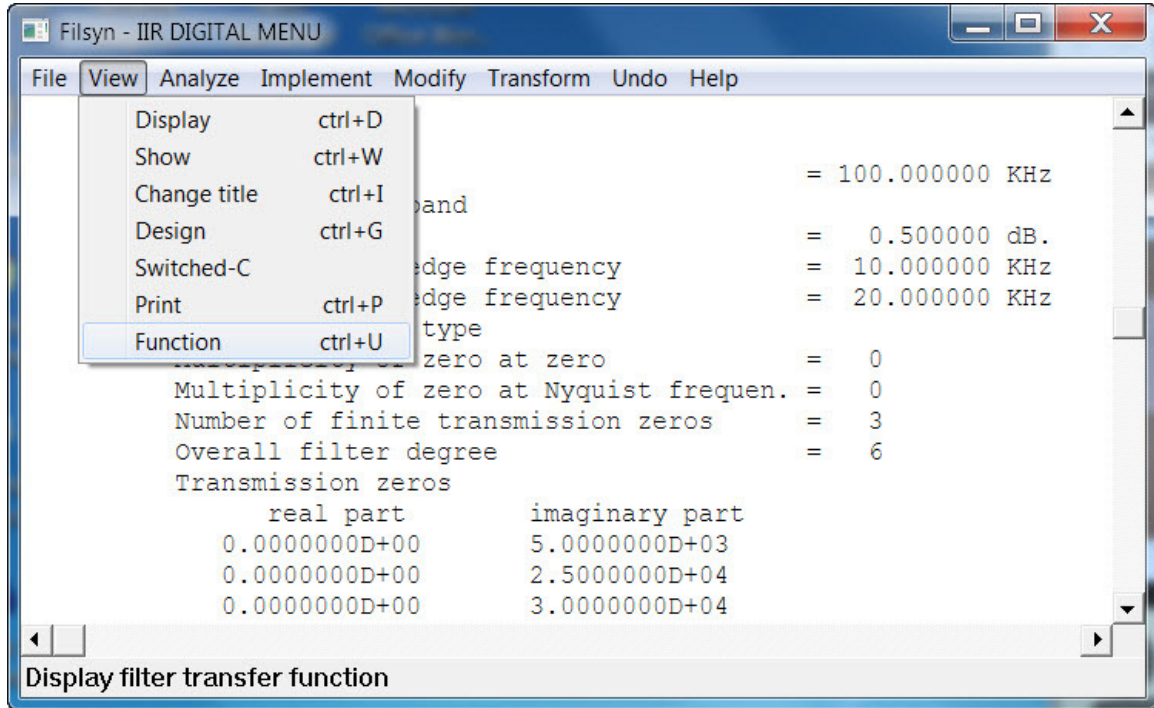
```

7. **Pole-zero** writes the pole-zero data to a text file. This may be used for documentation or for reading the data back to some other program that needs it. Again the file will have a '.pz' extension if the filter contains **no** delay equalizer sections and this file may be read back into **Filsyn** using the **Functional input** option. If however the filter *does* contain delay equalizers, then the file will have a '.dat' extension and may be read back into **Filsyn** using the **Analysis** option. The data format in these files will also depend on the application and is clearly indicated.
8. **Copy** is the usual copy option. We can select any part of the data on the screen with the help of the mouse and then copy it to the clipboard. Finally,
9. **Exit** terminates the program.

8.2 The 'View' menu option has the following submenu items:

1. **Display** prints the design data again, while.
2. **Show** prints the current filter implementation data.
3. **Change title** permits us to put our own title on the design.
4. **Design** offers us several hardware implementations of the current design,

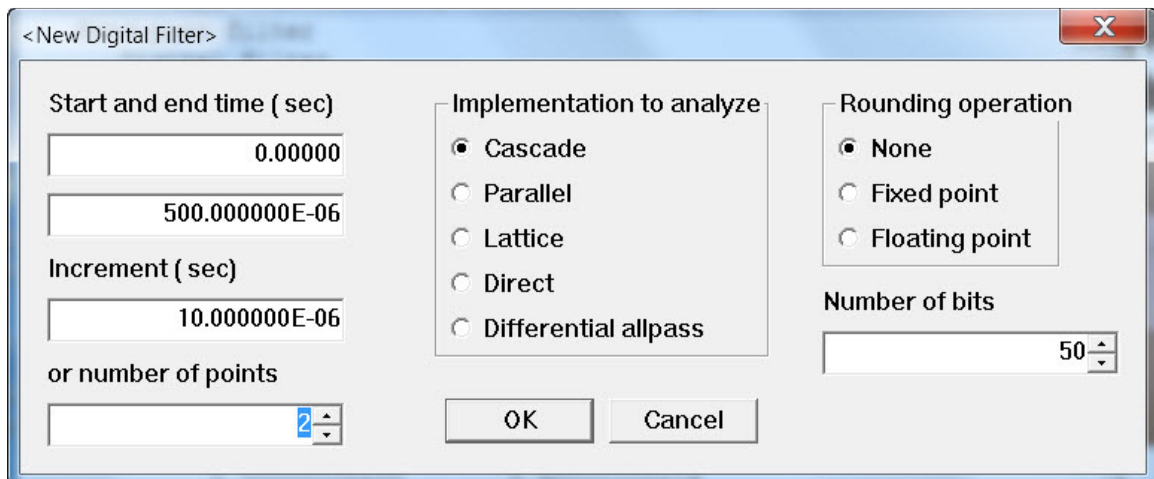
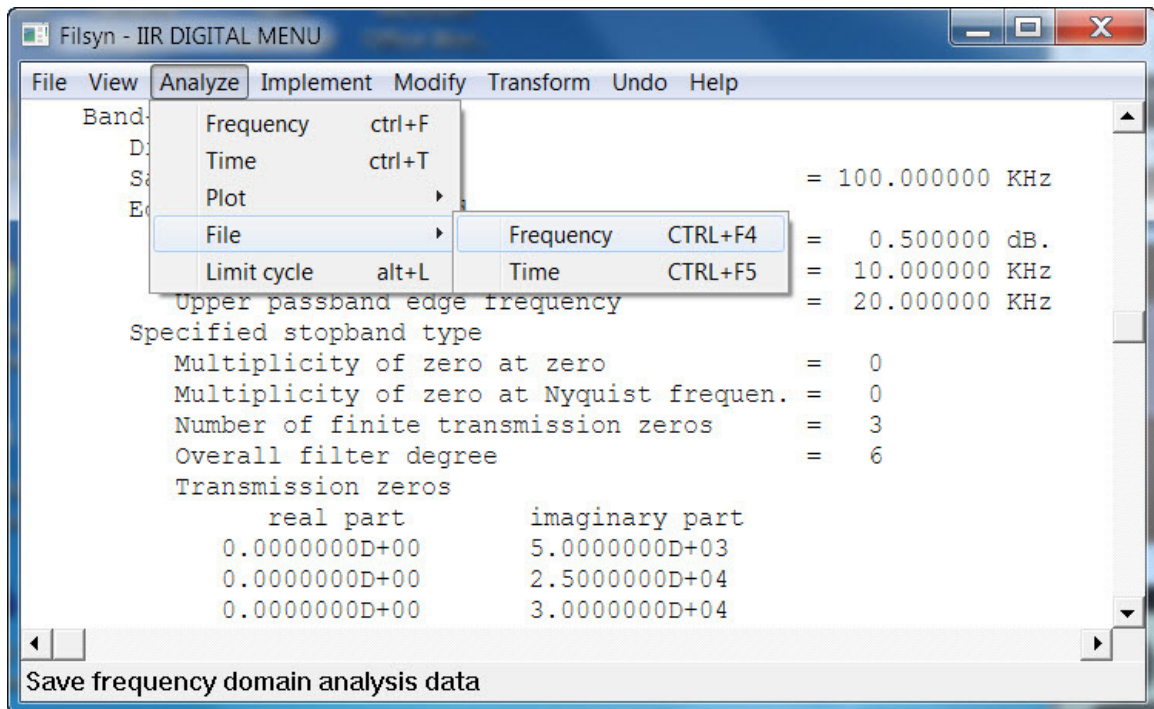
5. while **Switched C** writes the data to a special file if we wish to implement the filter in switched capacitor form. That file can then be read into the active RC segment for actual implementation.
6. **Print** is the usual switch to turn the print on or off. This is off by default; if it is on, everything that goes to the screen after turning the switch on, also goes to the printer or a text file.
7. **Function** prints the transfer function to the screen in factored form.



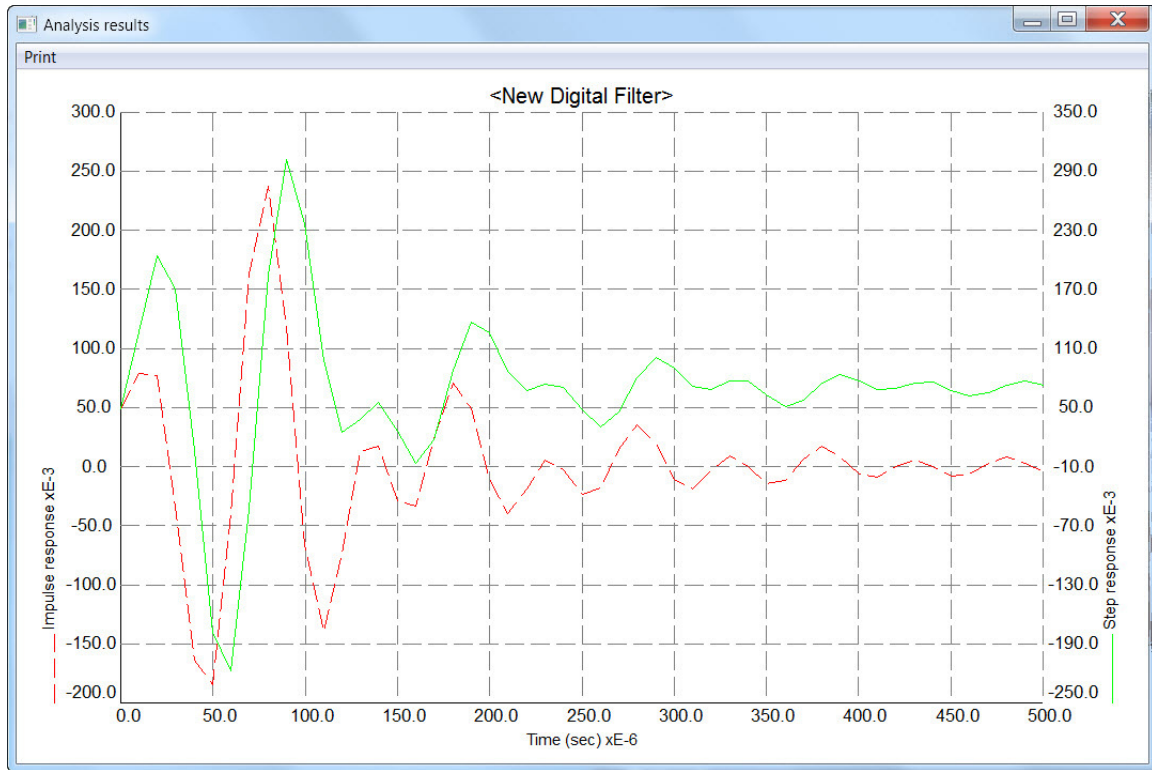
8.3 The ‘Analyze’ menu option contains the, by now usual, items:

1. **Frequency** and
2. **Time** are the usual analysis options, while
3. **Plot** offers us the capability to plot any of the computed responses.

Consider a simple example of the **Time** domain computation:



We compute the response from 0 to 0.5 msec in 10 μ sec steps (that is the sampling rate). Declining the option to print the tabulation, we plot the response instead:

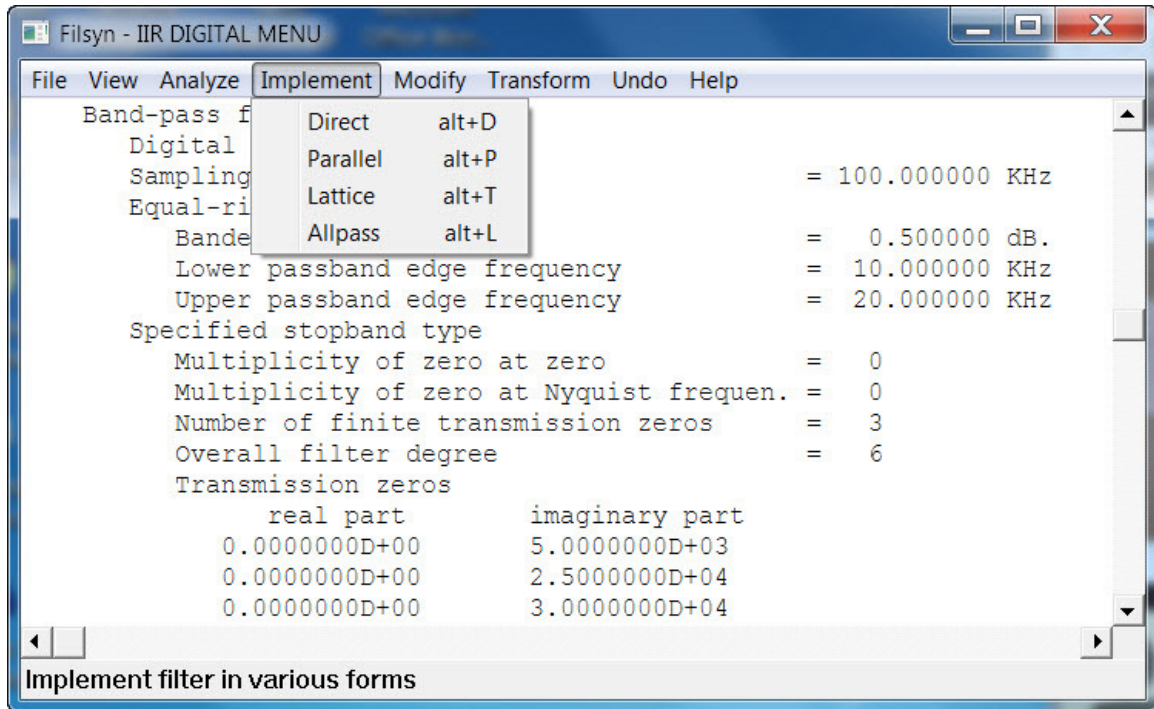


The **File** submenu option enables us to write the analysis data to a text file for documentation purposes, while **Limit cycle** provides tabulated results describing the limit cycle properties of the biquadratic sections of the cascade implementation. It assumes that we form these biquadratic sections from the numerator and denominator factors in their current implied sequence. It considers these biquadratic factors one at a time and independently of each other. It then tabulates the following data for each of the sections:

- The estimate of the limit cycle frequency
- The Long-Trick limit cycle amplitude bound
- The Kaiser-Sandberg (rms) limit cycle amplitude bound
- The limit cycle output of the biquadratic section, including the effects of its numerator

8.4 The 'Implement' menu option has these submenu items:

1. **Direct** provides us with the single section implementation of the filter, which is hardly ever used in the case of a high degree filter.
2. **Parallel** yields the implementation that has a number of second order sections all with a common input and whose outputs are summed up to form the final output. If we used the *impulse-invariant* Z-transform design method, this form is already available, since it was used to generate the Z-transform.
3. **Lattice** generates the Gray-Markel lattice implementation of the filter.



4. **Allpass** yields the differential allpass implementation (see Fig. 19). This realization is available if the filter characteristic function is a pure odd rational fraction (see *Appendix G* section 2 for the definition of the characteristic function). For those familiar with the theory of passive LC filters, this is equivalent to saying that the passive implementation is *symmetrical*. This requirement implies, that we use the bilinear Z-transform and that the multiplicities of the transmission zeros at zero and/or at the Nyquist frequencies are both odd, if not zero. The implementation can be obtained simply by using the **Implement** -> **Allpass** menu option. This option needs no further data and if it will not work, a warning is printed.

The form of the implementation is shown below in Fig. 1, where $A_0(z)$ and $A_1(z)$ are paths consisting of cascaded allpass sections and the two outputs in fact, generate *two* transfer functions, $H(z)$ and $G(z)$, that are complementary to each other, and one of them (usually the difference), is the function we need. For the definition of 'complementary' filters, please see *Application Note 3*, section 3. A frequency-domain analysis will always be necessary in order to find out, whether the sum or the difference path has the function you need.

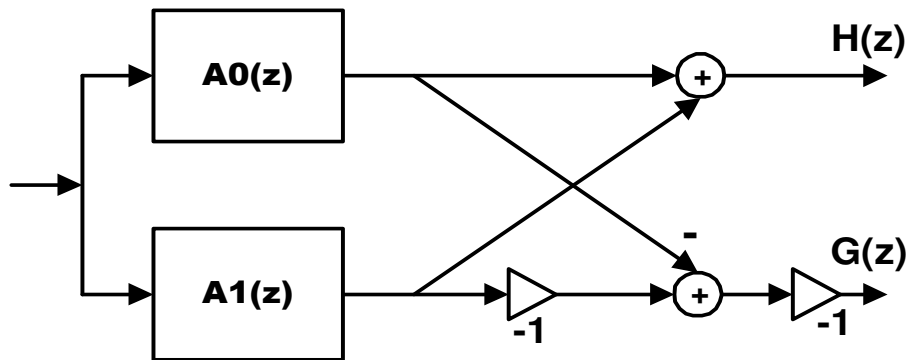
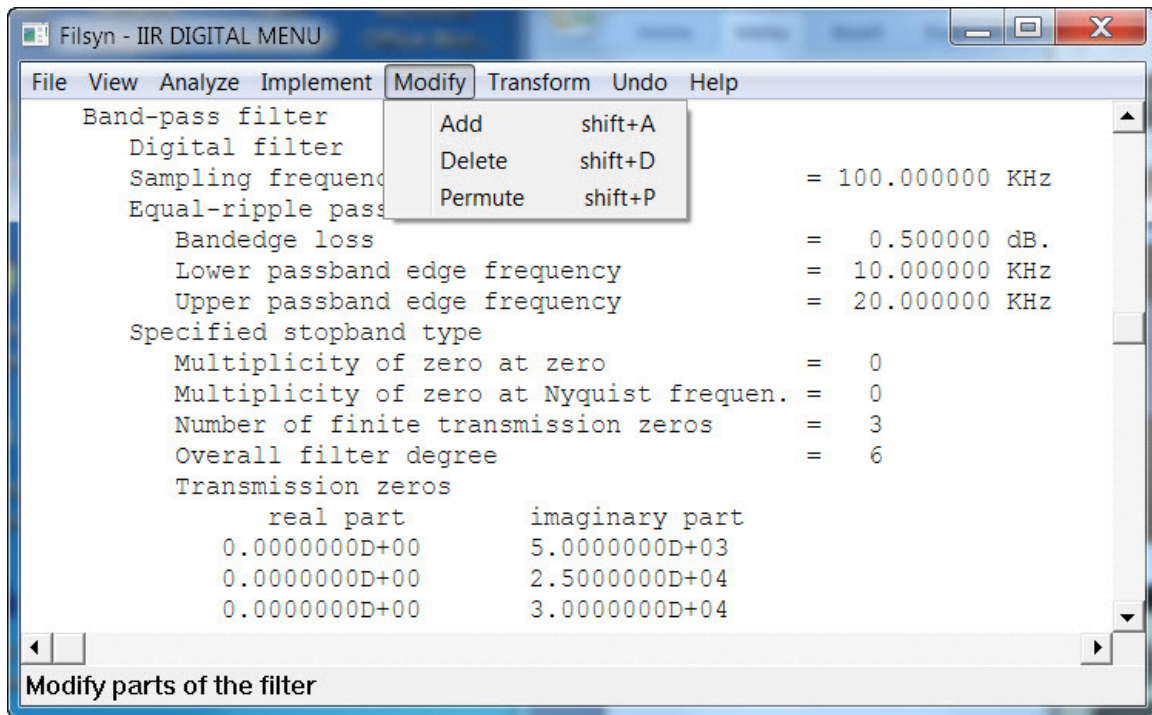


Fig. 19 Differential allpass implementation

The major advantage of this structure is that the allpass paths remain strictly allpass, even if the coefficients are truncated, hence it is *structurally passive* and is insensitive to coefficient precision. In addition, it can be implemented with fewer multipliers than any other realization.

8.5 The ‘Modify’ menu option has submenu items that change a part of the filter:

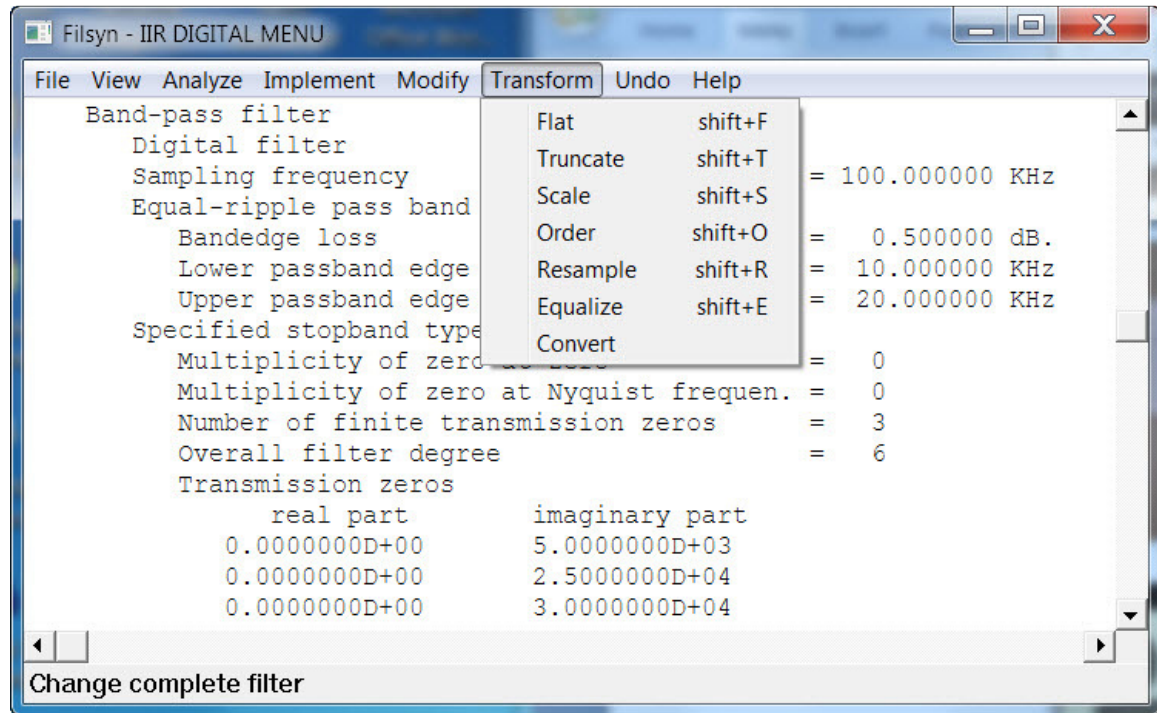


1. Add and

2. Delete offers us the capability of adding or deleting second order numerator or denominator factors to/from the cascade implementation. Since this form is the basic form all others are derived from, when we perform any of these operations, all other implementations are deleted and must be regenerated, if needed.

3. Permute lets us rearrange the sequence of numerators and denominators of the cascade implementation again. This will be used later on for optimizing the dynamic behavior of the cascade implementation.

8.6 The ‘Transform’ menu option has submenu items that affect the whole filter:



1. **Flat** modifies the gain of the filter by adding a (positive or negative) flat loss to it.

2. **Truncate** truncates or rounds the filter coefficients to a specified number of (fixed- or floating point binary digits).

3. **Scale** is an operation, that changes the gains of each of the cascaded second order sections in order to make sure their outputs are not overloaded. Two algorithms are available:

- L_2 scaling
- L_∞ scaling

The L_2 algorithm calculates the scale factors in such a manner that the sum of the squares of the unit pulse response of every section is never greater than unity. The calculation is always done from the common input to the output of the particular section for which we are calculating the scale factor.

For L_2 scaling, the sections are formed as follows:

- a) The first denominator quadratic factor is the first section with a unity numerator.

- b) The second section is formed by the first numerator factor paired with the second denominator factor.
- c) The next section is formed by the second numerator factor paired with the third denominator factor, etc. This method is employed because the most often used implementation uses this particular arrangement and it is also the method used by the **Order** menu item.

In the case of the L_∞ scaling, the scale factors are calculated in such a manner that the frequency response of every section is not greater than unity at any frequency. The frequency response may be defined to be either that of individual 2nd order sections (pessimistic case) or that of the (common) input to the output of the section under investigation (more common case). For L_∞ scaling, one may request either the section arrangement used for the L_2 scaling, or the natural arrangement, where the k^{th} numerator is paired with the k^{th} denominator to form the k^{th} section.

In either case, after the new scale factors are printed, we have the option of modifying them, possibly to change them into simple binary shifts in order to simplify their implementation. Since the resulting noise gain is also printed, we can observe the effects of both the scaling policies as well as of changing scale factors.

4. Order is another complex operation, that invokes the Peled-Liu heuristic pairing and ordering algorithm as described in their book (see ref. [37]). The menu item needs no data and the sequence of numerators and denominators remain unchanged. If we wish to make use of the results, we must perform the permutations subsequently ourselves.

The operation provides 15 sub-optimal pairing and ordering sequences with their corresponding noise gain values. From these, we must select the best (lowest noise gain), then use the **Permute** operation to rearrange the sequence of numerator and denominator factors, and finally employ the **Scaling** menu item to calculate the optimal scale factors. The scaling method applied in this pairing and ordering algorithm is the L_2 scaling, with the denominator factors implemented first in the physical realization of the filter. Consequently, if we use the same scaling policy, the noise gain obtained from the scaling algorithm will be the same as that of the optimal pairing and ordering.

The starting permutations in the optimization are selected by a pseudo-random process with a system-dependent initialization. On systems where the initialization is based on the system clock, two consecutive invocations of the **Order** menu item may yield different results. On other systems the initialization is based on built-in constants and therefore it will always yield the same results.

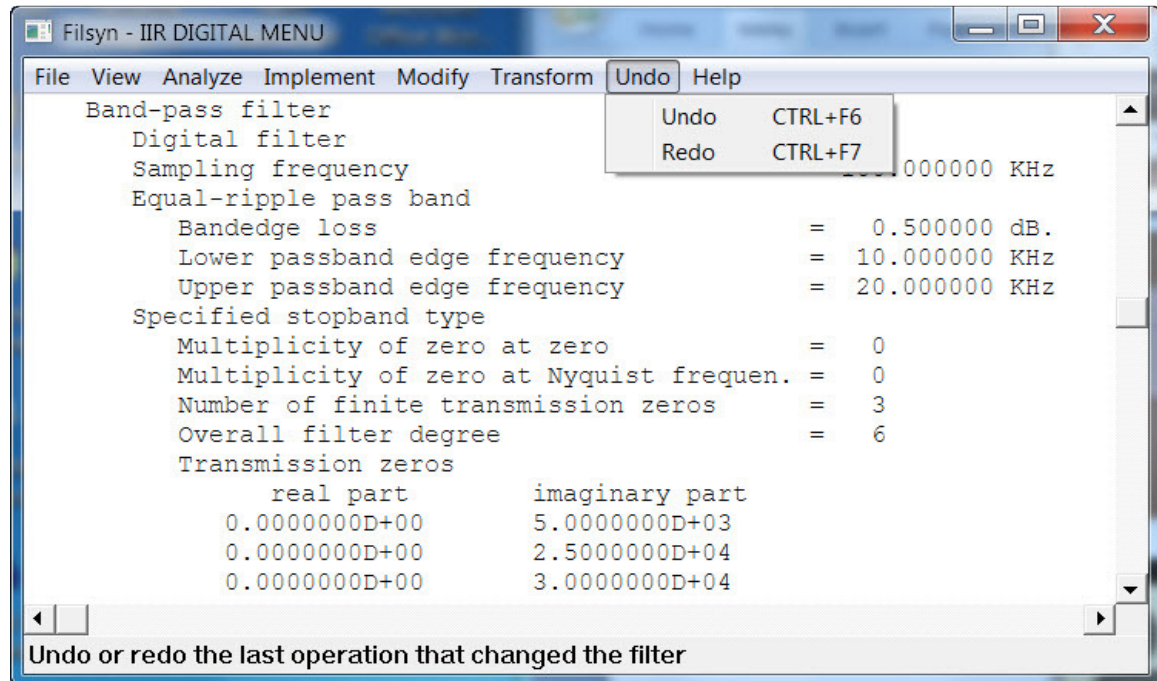
5. Resample simply changes the sampling frequency.

6. Equalize invokes our usual delay equalization operation.

7. Convert performs a conversion of an IIR lowpass filter to a highpass or vice versa by effectively inverting the frequency scale. The operation is used in both *Application notes*

6 and 15. Applying it twice the original set of data is returned.

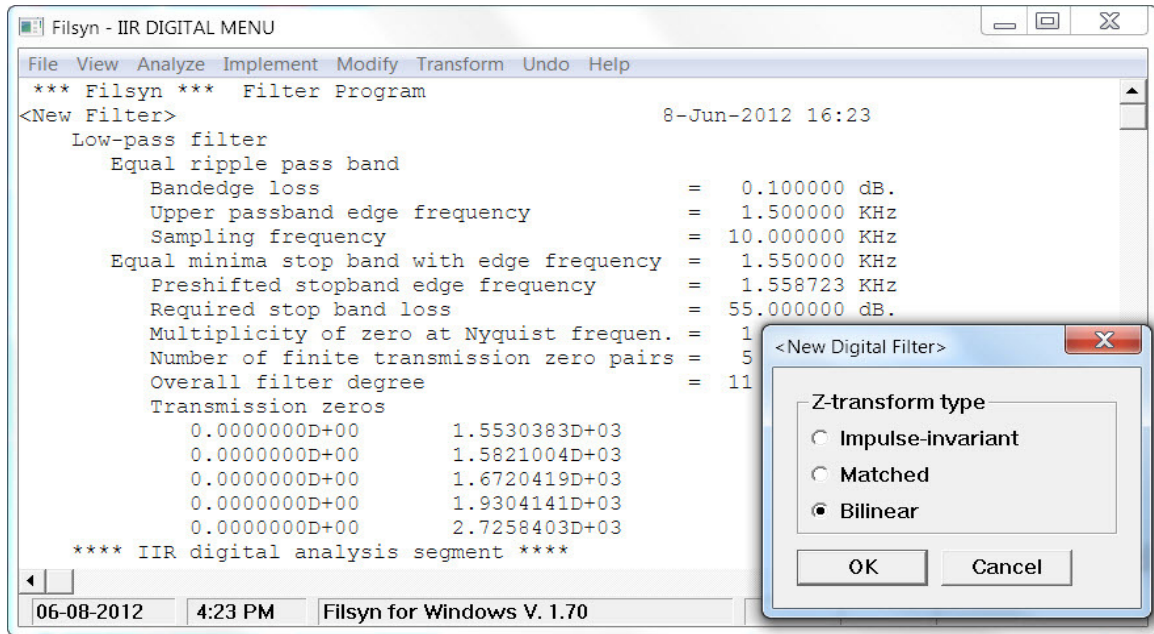
8.7 The ‘Undo’ menu option has the usual ‘Undo’ and ‘Redo’ operations available. **Undo** can undo up to ten of the last operations that modified the filter, in reverse sequence. **Redo** can redo them, except the first undo.



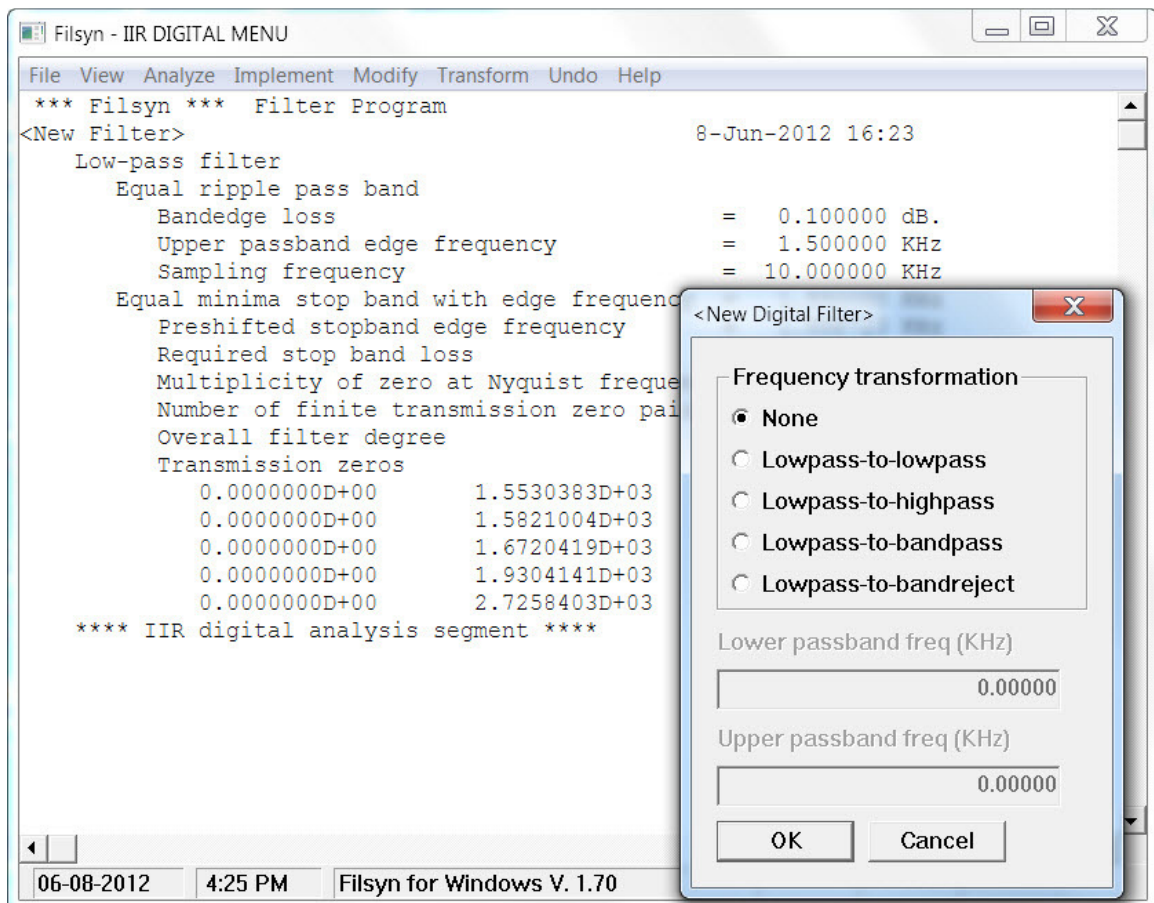
8.8 The ‘Help’ menu option contains the usual items.

8.9 Example 1: Elliptic Lowpass Filter

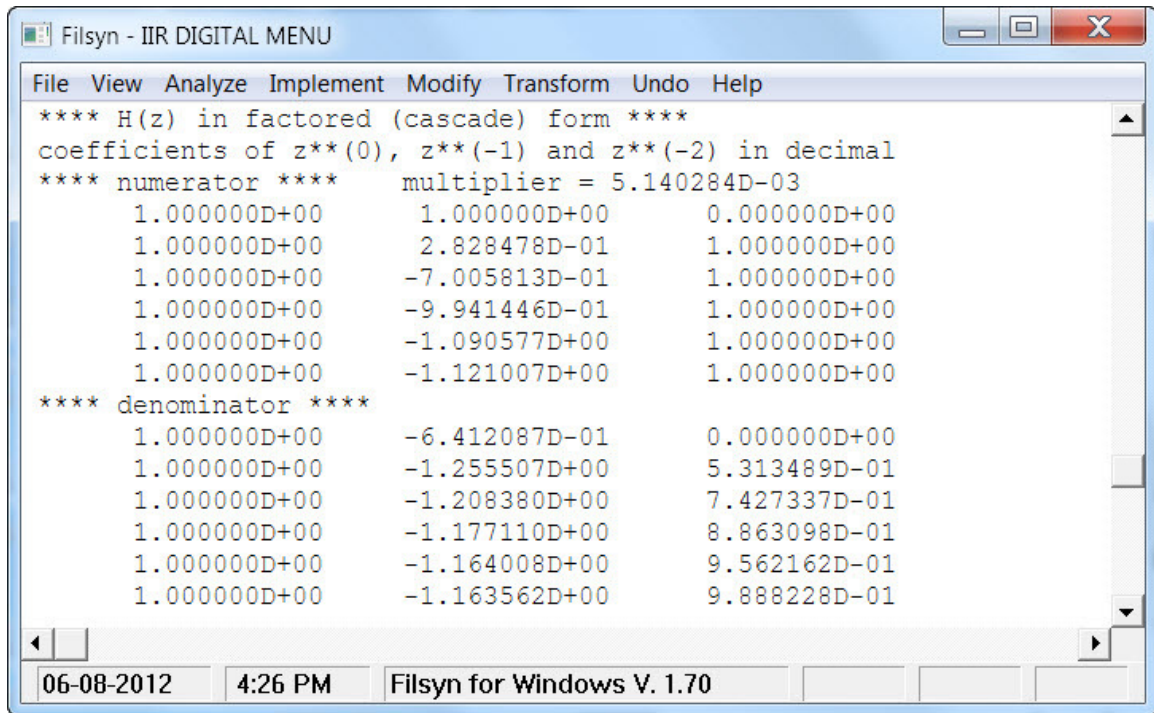
Our first example is an 11th order elliptic lowpass with sampling frequency of 10 KHz, passband up to 1500 Hz and 0.1 dB ripple, stopband beginning at 1550 Hz and 55 dB minimum loss. Selecting the bilinear Z-transform:



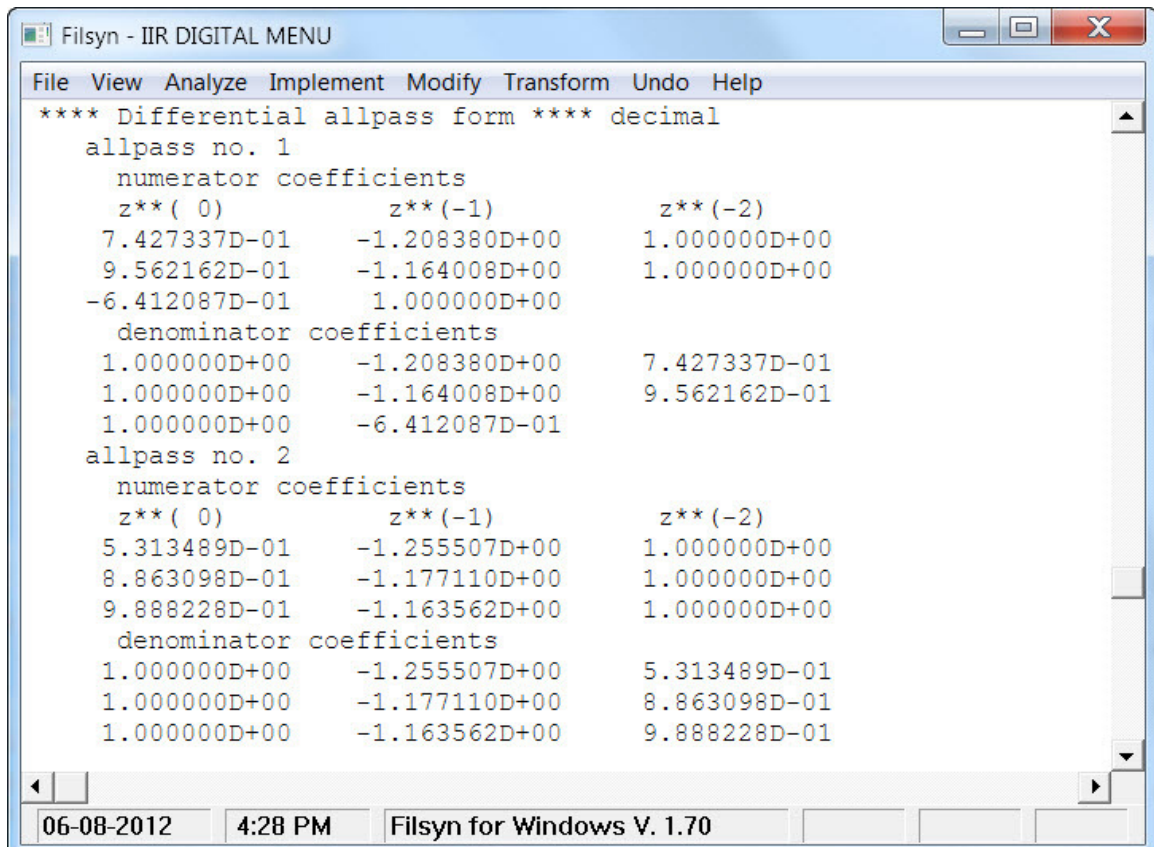
Next we are offered a frequency transformation, which we decline (the default):



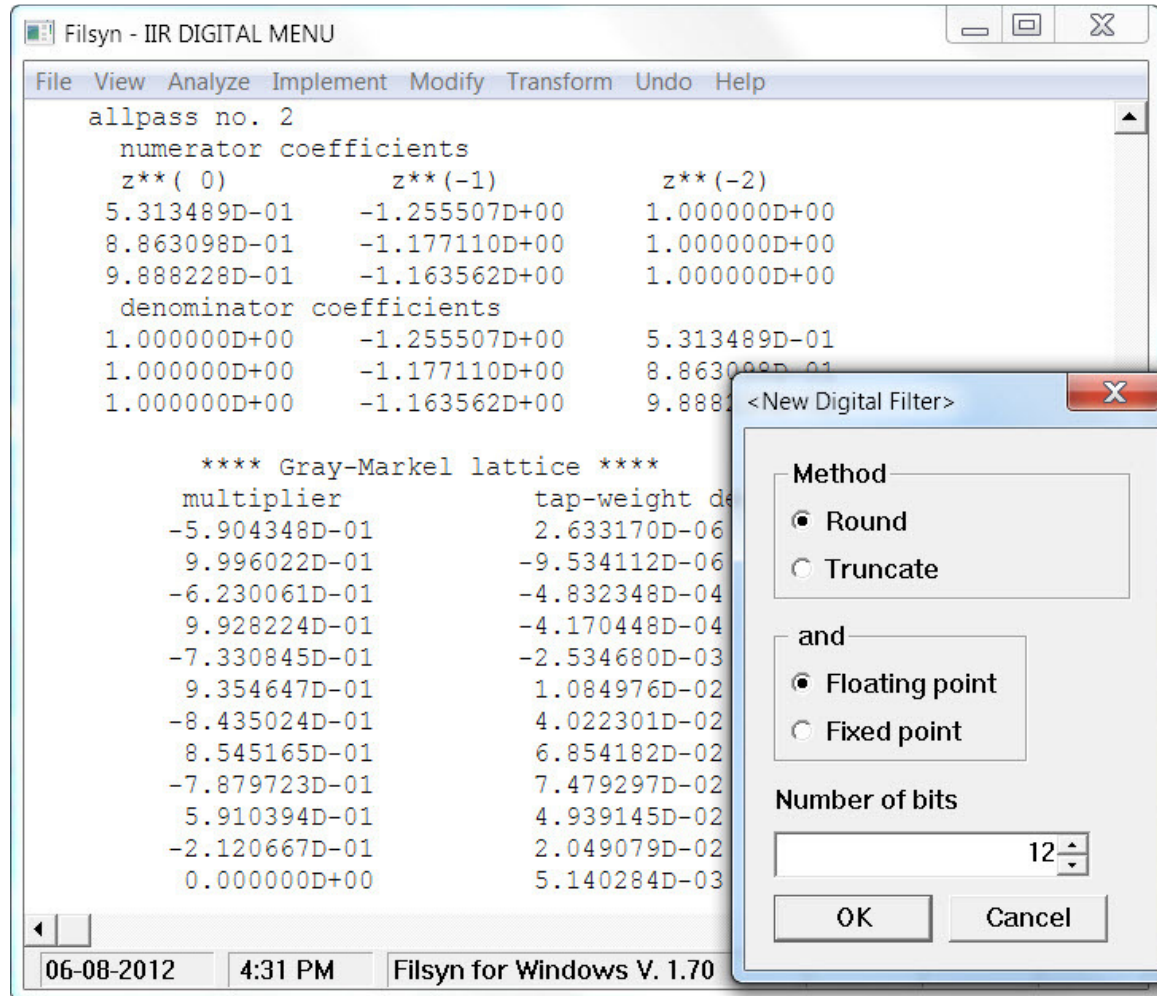
The display now shows the cascaded second order section implementation:



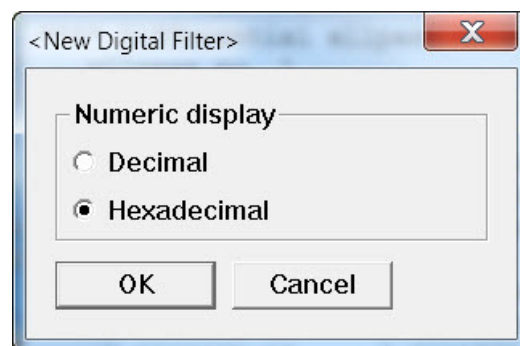
Next we ask for alternative implementations and all of them are available, including the **Differential allpass** form (see Fig. 19 above), which is unavailable more often than not:



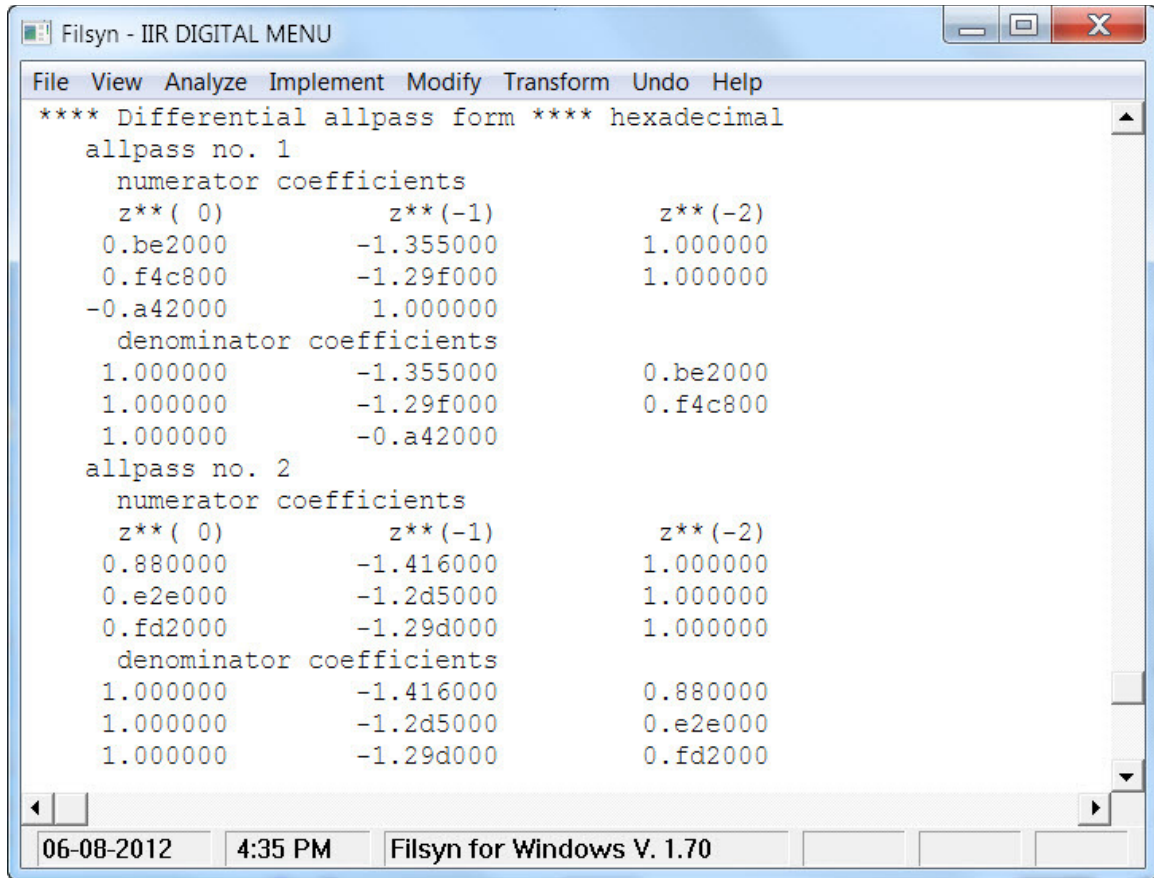
Our next step is to truncate the coefficients of all implementations:



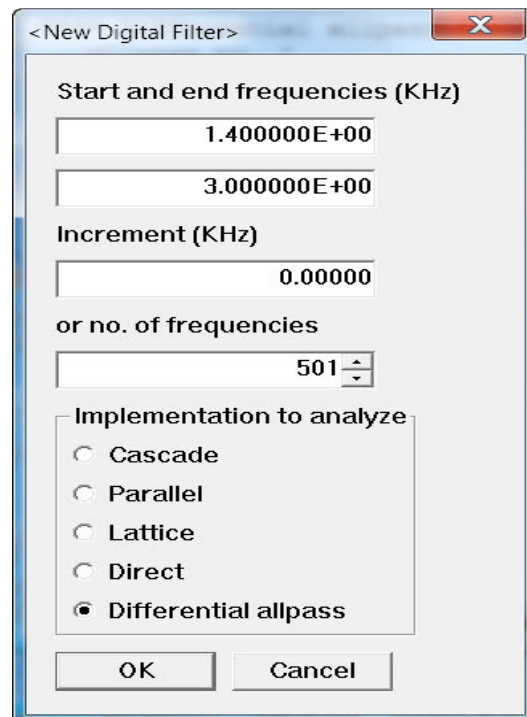
Here we see the differential allpass and Gray-Markel lattice implementations in the background. After the truncation operation we wish to see the coefficients in hexadecimal form, which we can do by using the **View->Show** menu option and selecting the hexadecimal format:



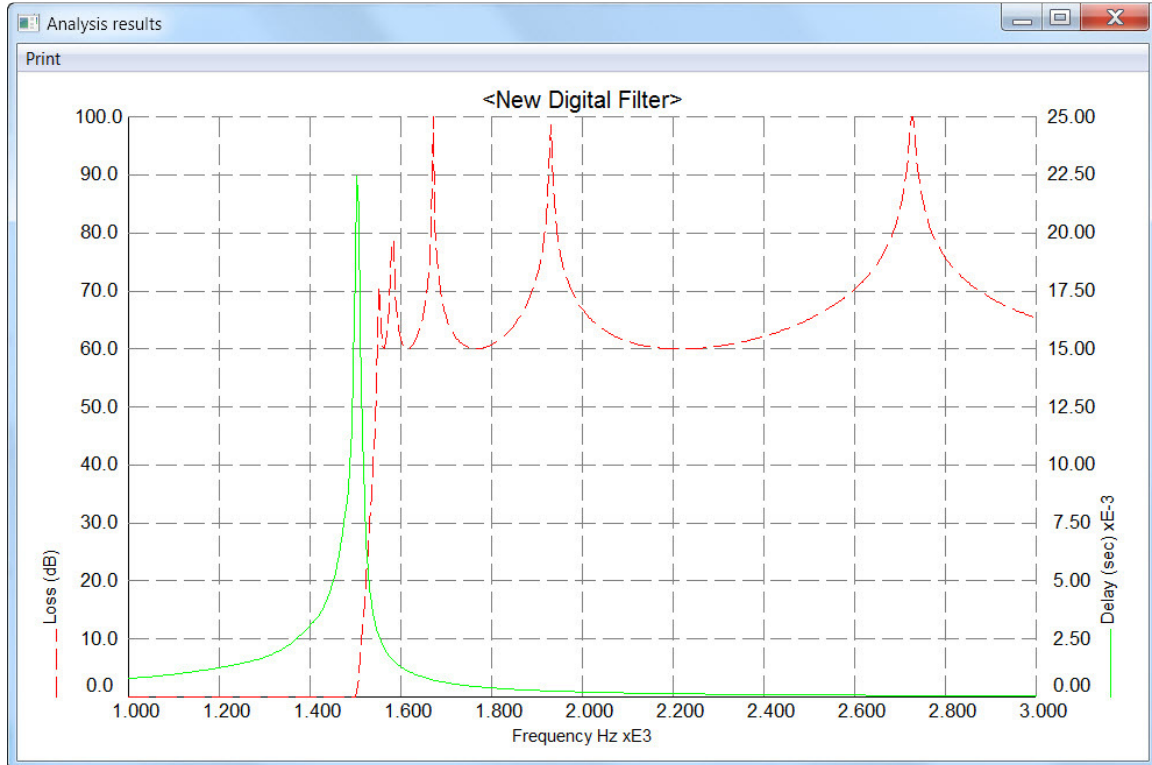
The differential allpass results:



Analyzing the truncated forms in the frequency domain and just in the transition region, we get for instance, for the differential allpass implementation:

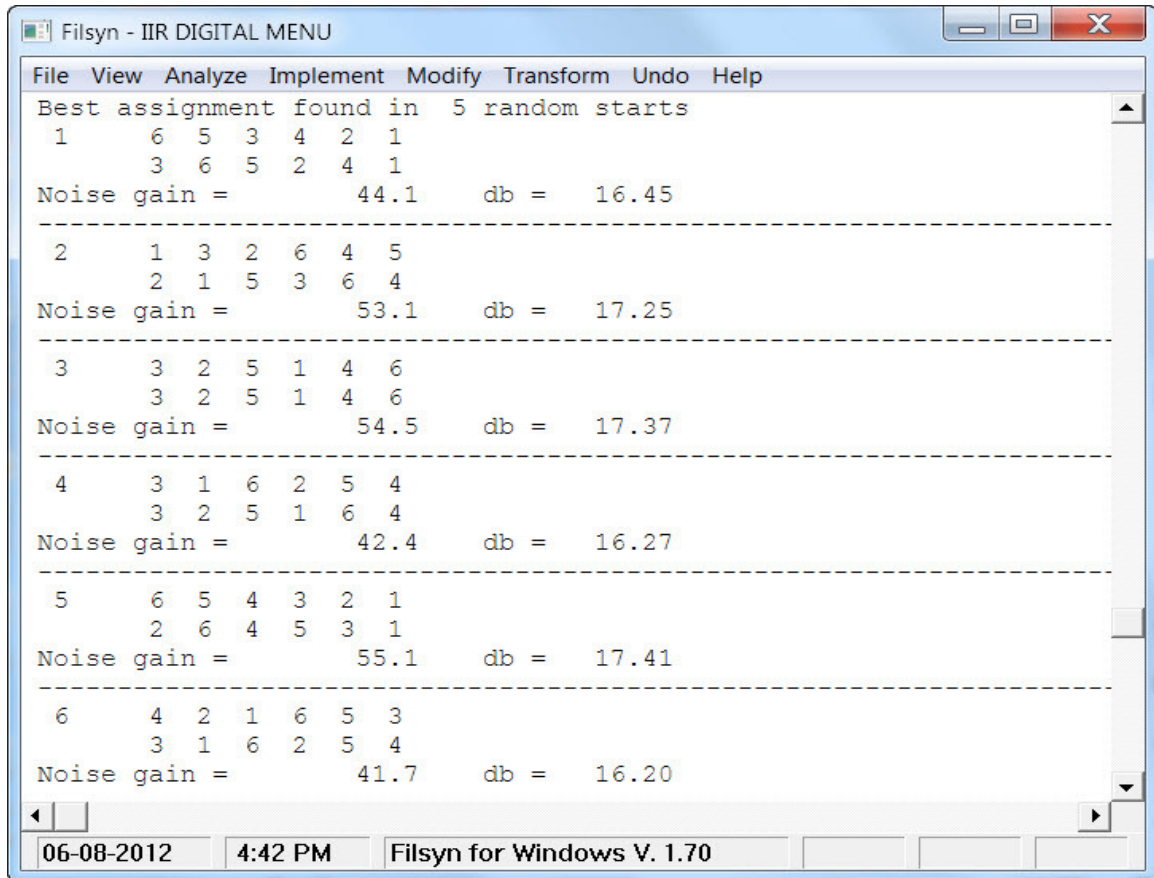


The response is shown next:

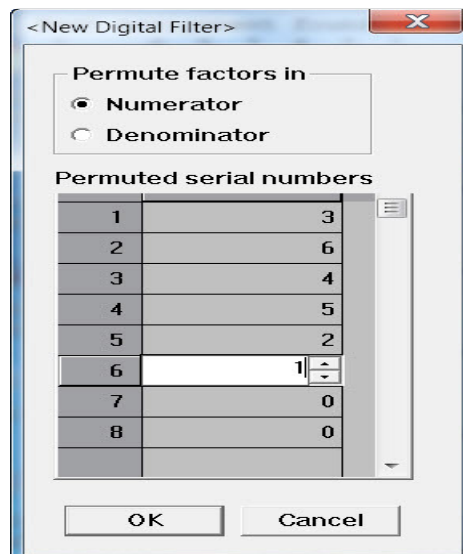


The response, the sum function this time, seems to be quite close to the ideal one without truncation and in fact, the differential allpass seems to be the most insensitive to truncation. The cascade one is close, because both have the property that while the transmission zeros will move due to the truncation, they remain on the unit circle (real frequencies). The parallel and the Gray-Markel lattice implementations are substantially worse and quite similar to each other. Finally the direct implementation is useless at this level of truncation.

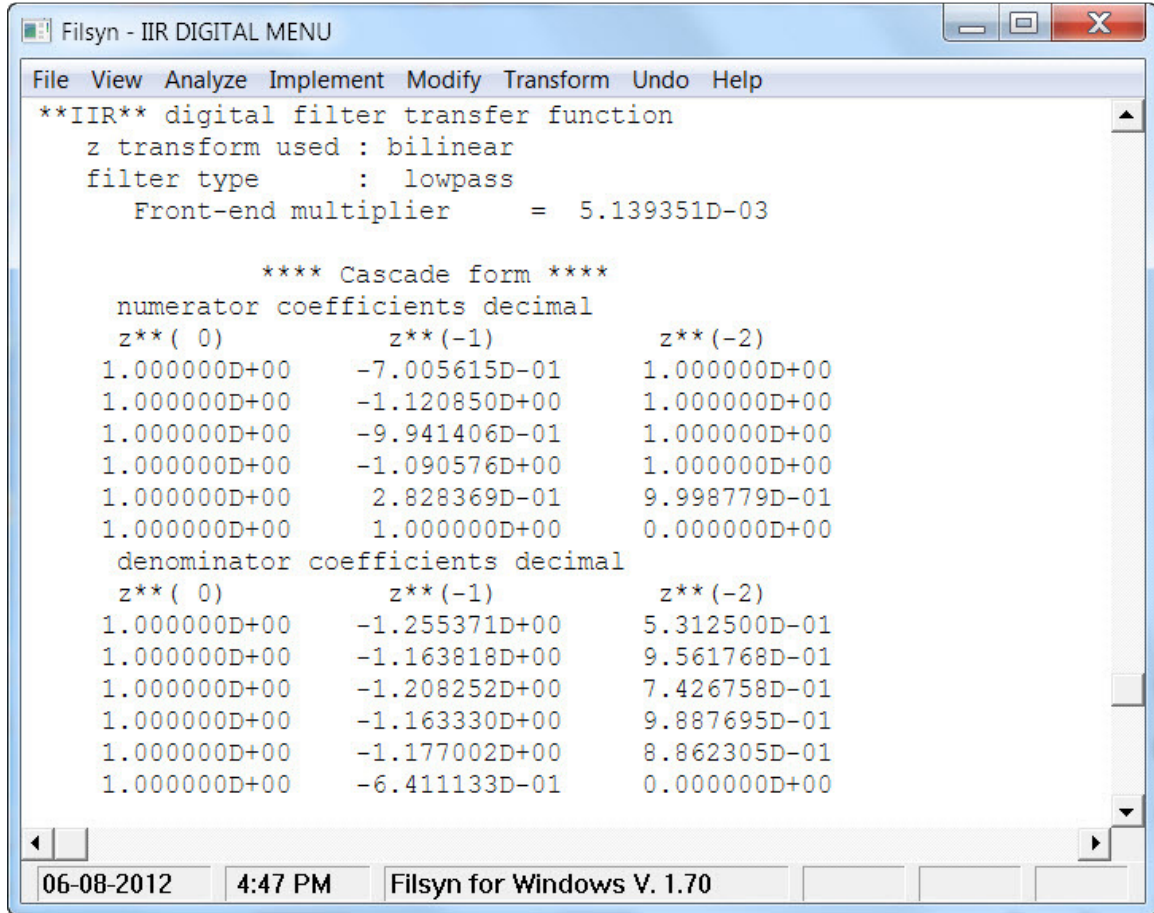
We can return to the non-truncated coefficients using the **Undo** menu option and next use the **Transform->Order** menu item to find the optimal arrangements of the numerator and denominator factors in the cascade implementation. We show here the first six rows:



The best result (least noise gain) is 15.86 dB (not shown), but because this is an odd degree filter, the linear factors in the numerator and denominator must be kept together. Luckily, it turns out that the best result found is also one that keeps the linear factors numbered 1 together. The permutations needed are 3,6,4,5,2,1 and 2,5,3,6,4,1 in the numerator and denominator respectively. Performing the permutation is by calling the **Modify->Permute** option:



The rearranged transfer function, after permuting the denominators as well, is now:



The screenshot shows a software window titled "Filsyn - IIR DIGITAL MENU". The window contains a text area with the following text:

```

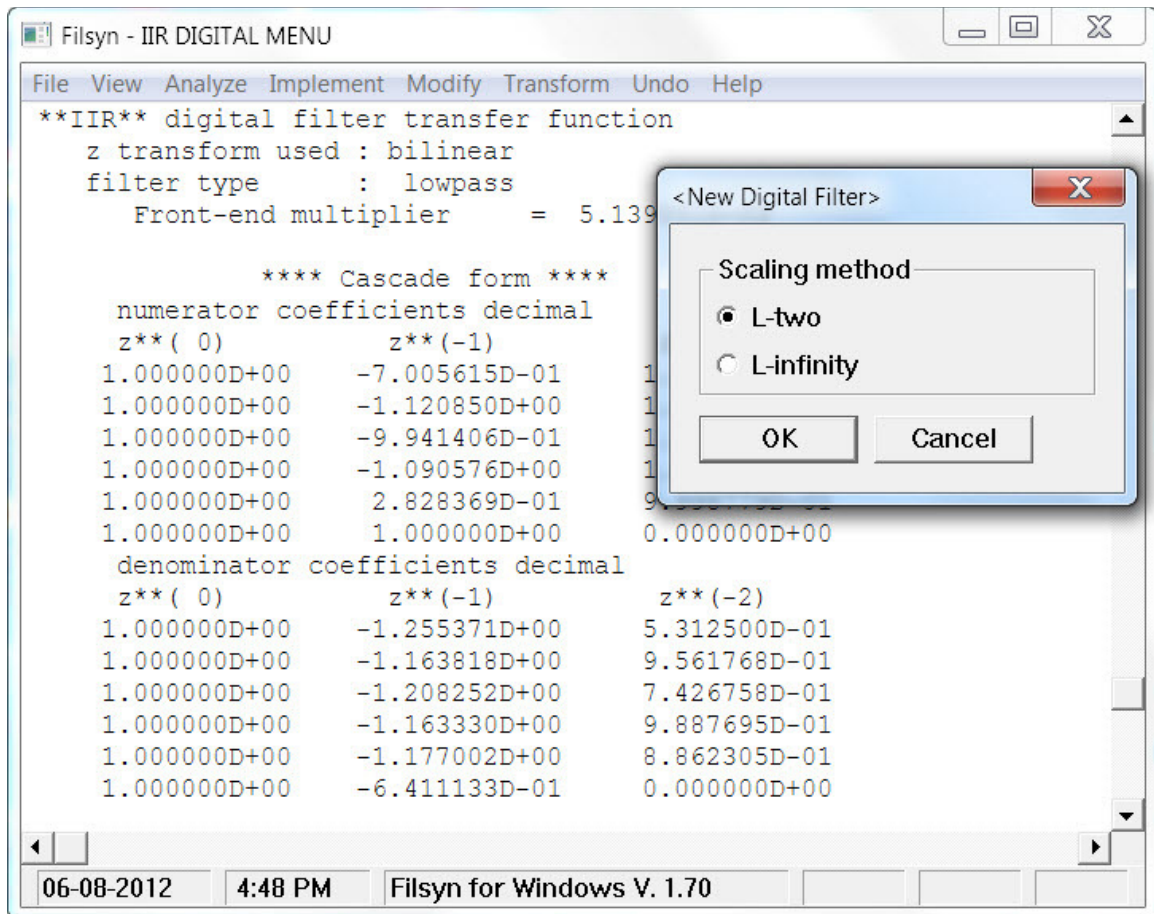
**IIR** digital filter transfer function
z transform used : bilinear
filter type      : lowpass
Front-end multiplier = 5.139351D-03

**** Cascade form ****
numerator coefficients decimal
z**( 0)          z**(-1)          z**(-2)
1.000000D+00     -7.005615D-01     1.000000D+00
1.000000D+00     -1.120850D+00     1.000000D+00
1.000000D+00     -9.941406D-01     1.000000D+00
1.000000D+00     -1.090576D+00     1.000000D+00
1.000000D+00     2.828369D-01      9.998779D-01
1.000000D+00     1.000000D+00      0.000000D+00
denominator coefficients decimal
z**( 0)          z**(-1)          z**(-2)
1.000000D+00     -1.255371D+00     5.312500D-01
1.000000D+00     -1.163818D+00     9.561768D-01
1.000000D+00     -1.208252D+00     7.426758D-01
1.000000D+00     -1.163330D+00     9.887695D-01
1.000000D+00     -1.177002D+00     8.862305D-01
1.000000D+00     -6.411133D-01     0.000000D+00

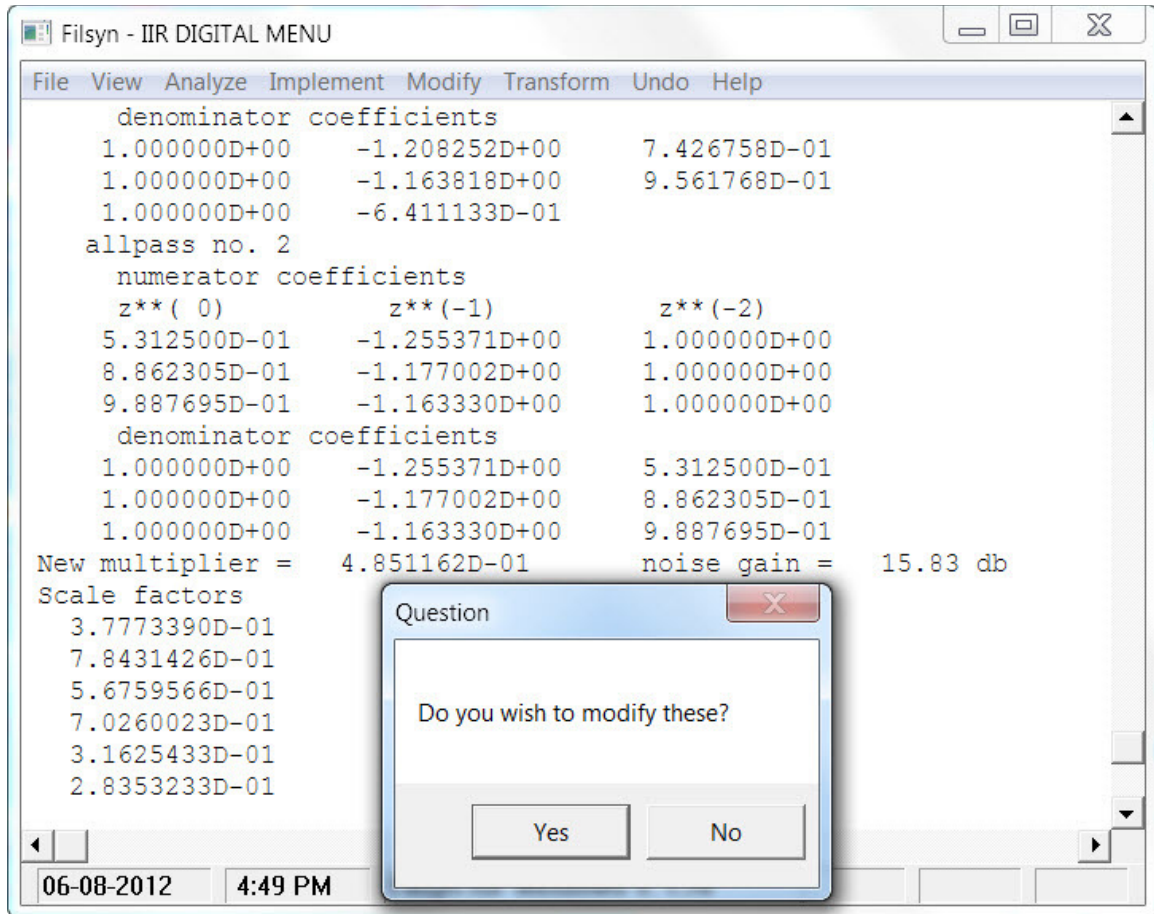
```

The status bar at the bottom of the window displays the date "06-08-2012", the time "4:47 PM", and the version "Filsyn for Windows V. 1.70".

Next we invoke the scaling function using the **Transform->Scale** option and use the L_2 scaling (the one used for the pairing and ordering operation):

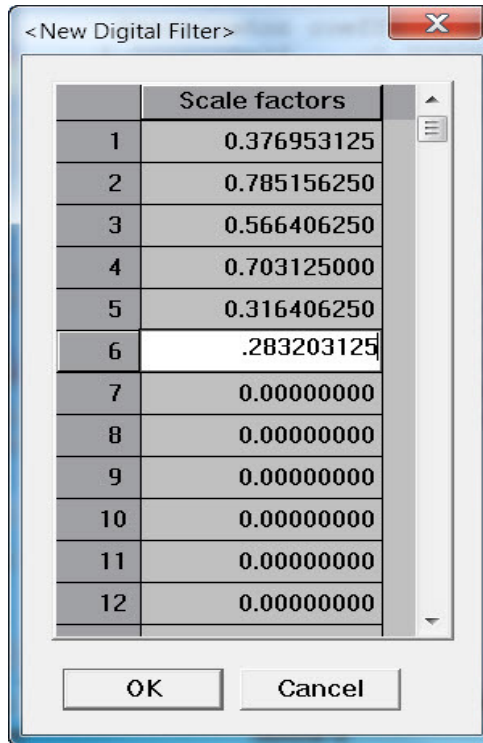


This operation displays the following results:

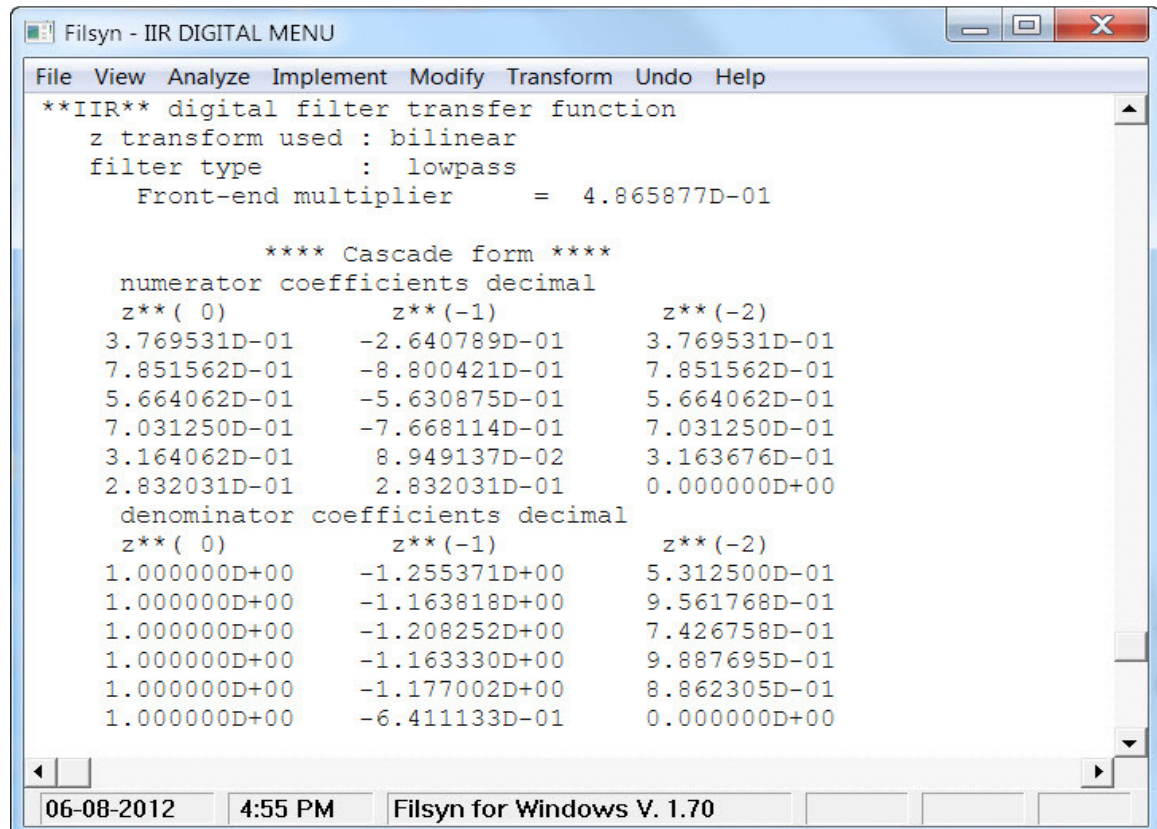


Note that the noise factor is very close to what the ordering algorithm computed.

We do indeed wish to modify these in a manner that the numbers should be unchanged by a truncation operation. If we wish to use 12 bits for truncation, the simplest way to find what we should use is to multiply these numbers by 4096 (2^{12}), truncate the results to an integer and divide the results by 4096. We used here 512 (2^9 , 9 bits) instead, leading to the scale factors:



The final results are shown next. We only show the cascade implementation here, since the others are not affected by these operations.



Truncating the coefficients to 12 bits again give the new values (note that the numerator quadratic and constant coefficients are unchanged):

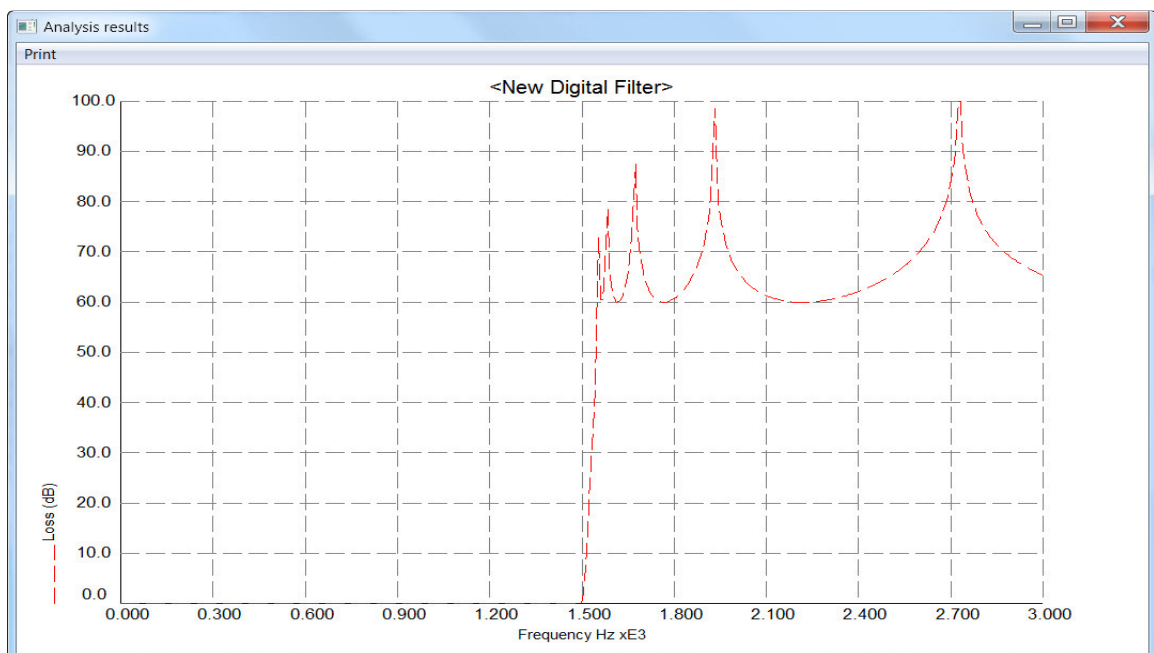
```

Filsyn - IIR DIGITAL MENU
File View Analyze Implement Modify Transform Undo Help
**IIR** digital filter transfer function
z transform used : bilinear
filter type      : lowpass
Front-end multiplier = 4.865723D-01

**** Cascade form ****
numerator coefficients hexadecimal
z**( 0)      z**(-1)      z**(-2)
0.608000    -0.439800    0.608000
0.c90000    -0.e14800    0.c90000
0.910000    -0.902000    0.910000
0.b40000    -0.c44800    0.b40000
0.510000    0.16e800    0.50fc00
0.488000    0.488000    0.000000
denominator coefficients hexadecimal
z**( 0)      z**(-1)      z**(-2)
1.000000    -1.416000    0.880000
1.000000    -1.29f000    0.f4c800
1.000000    -1.355000    0.be2000
1.000000    -1.29d000    0.fd2000
1.000000    -1.2d5000    0.e2e000
1.000000    -0.a42000    0.000000
06-08-2012  4:57 PM  Filsyn for Windows V. 1.70

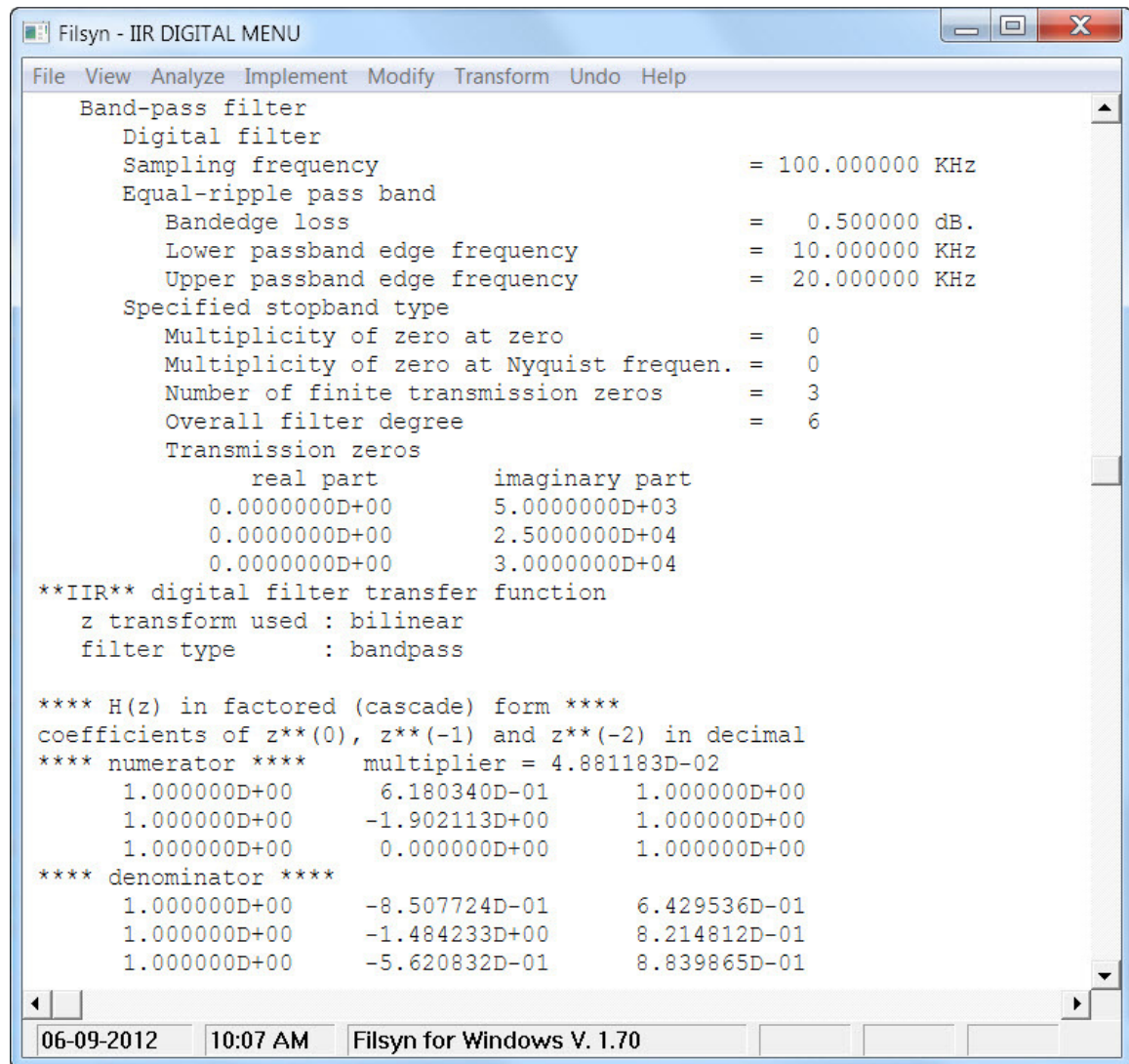
```

Performing another frequency domain analysis indicates excellent performance, the results differ from the non-truncated one by only a minute amount.



8.10 Example 2: Digital Bandpass Filter

This is a simple 6th order bandpass, with passband from 10 KHz to 20 KHz:

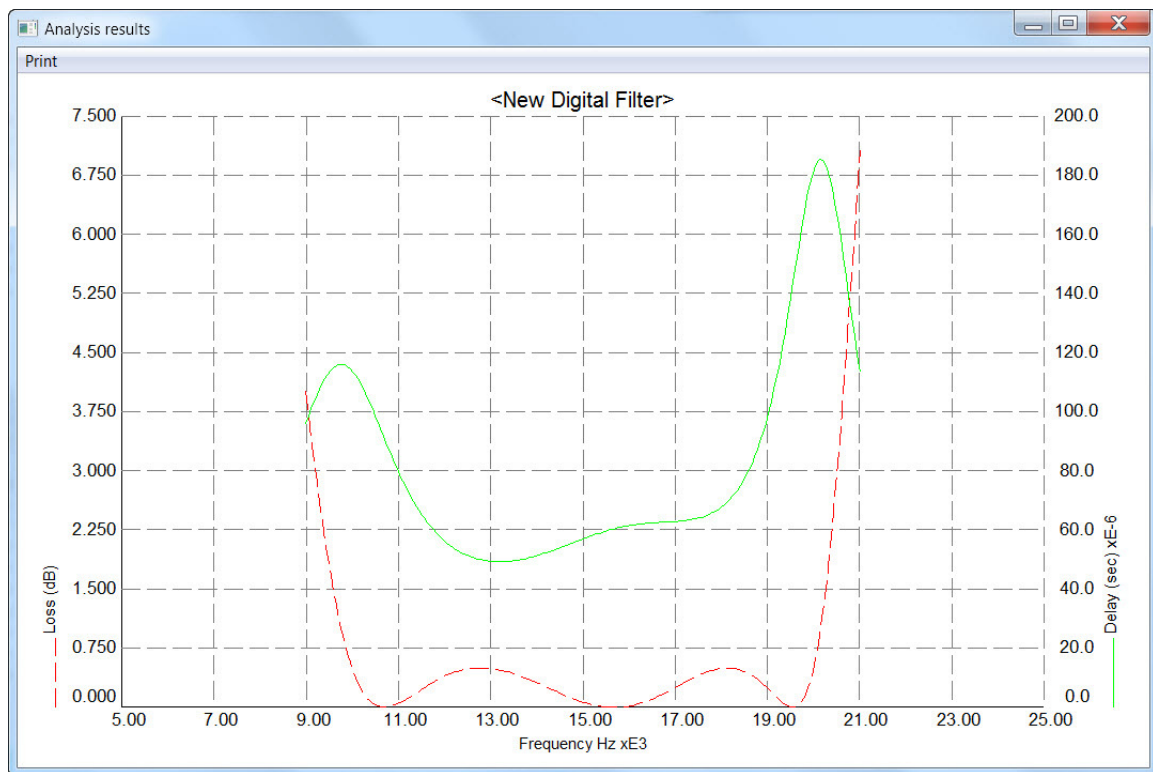


```

Filsyn - IIR DIGITAL MENU
File View Analyze Implement Modify Transform Undo Help
Band-pass filter
Digital filter
Sampling frequency                      = 100.000000 KHz
Equal-ripple pass band
  Bandedge loss                        = 0.500000 dB.
  Lower passband edge frequency        = 10.000000 KHz
  Upper passband edge frequency        = 20.000000 KHz
Specified stopband type
  Multiplicity of zero at zero          = 0
  Multiplicity of zero at Nyquist frequen. = 0
  Number of finite transmission zeros   = 3
  Overall filter degree                = 6
  Transmission zeros
    real part      imaginary part
    0.000000D+00   5.000000D+03
    0.000000D+00   2.500000D+04
    0.000000D+00   3.000000D+04
**IIR** digital filter transfer function
  z transform used : bilinear
  filter type      : bandpass

**** H(z) in factored (cascade) form ****
coefficients of z**(0), z**(-1) and z**(-2) in decimal
**** numerator ****      multiplier = 4.881183D-02
    1.000000D+00      6.180340D-01      1.000000D+00
    1.000000D+00     -1.902113D+00      1.000000D+00
    1.000000D+00      0.000000D+00      1.000000D+00
**** denominator ****
    1.000000D+00     -8.507724D-01      6.429536D-01
    1.000000D+00     -1.484233D+00      8.214812D-01
    1.000000D+00     -5.620832D-01      8.839865D-01
06-09-2012 10:07 AM Filsyn for Windows V. 1.70
  
```

Analyzing it in the passband we have a performance shown:



Next we invoke the delay equalizer by the **Transform->Equalize** menu option and use two second order sections and equalize the delay from 10.5 KHz to 19.5 KHz:

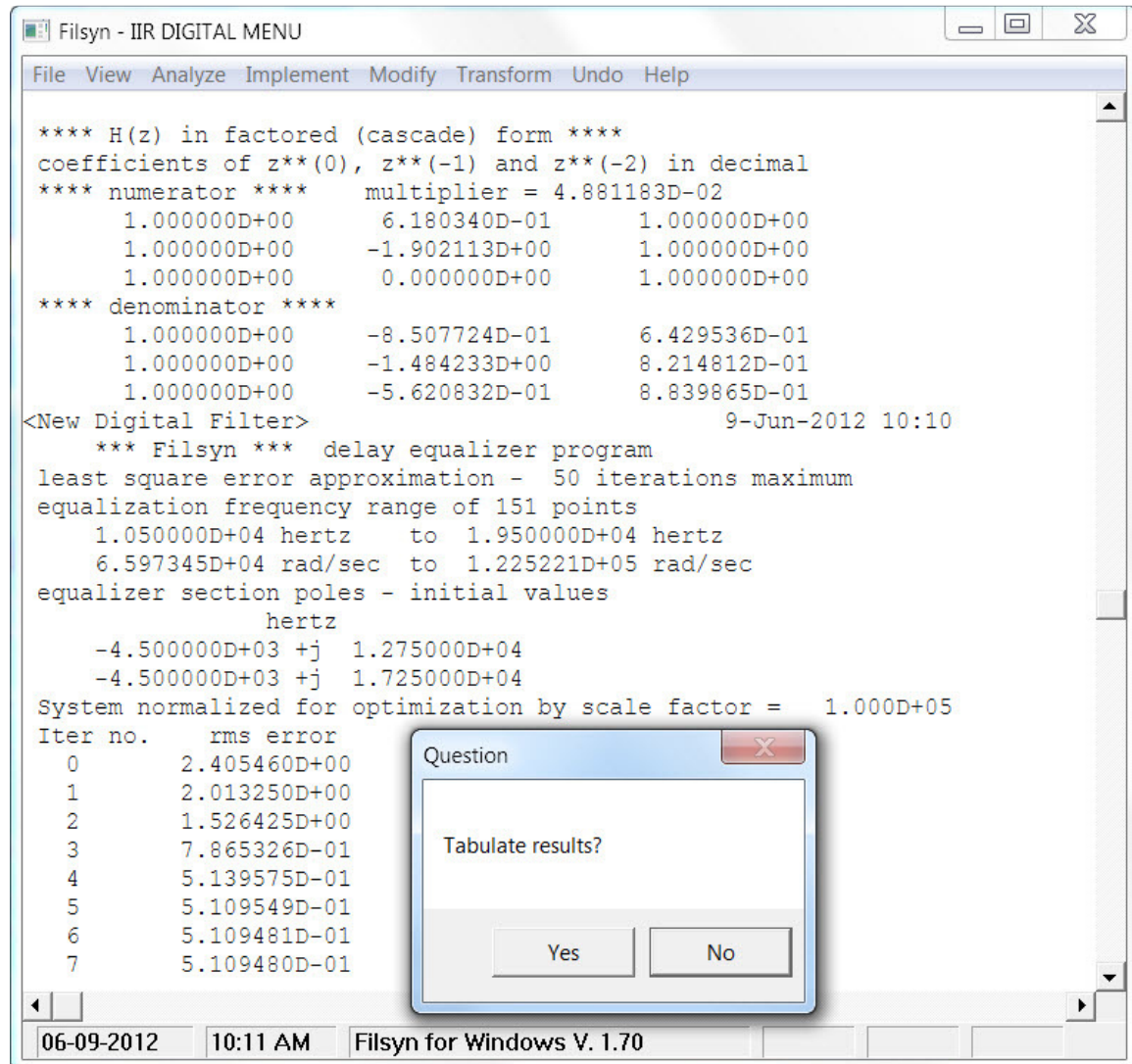
The figure shows a dialog box titled "<New Digital Filter>". It contains the following settings:

- Number of 1st order sections: 0
- Number of 2nd order sections: 2
- Lower approx. band edge (KHz): 10.500000E+00
- Upper approx. band edge (KHz): 19.5
- Stepsize: 0.00000

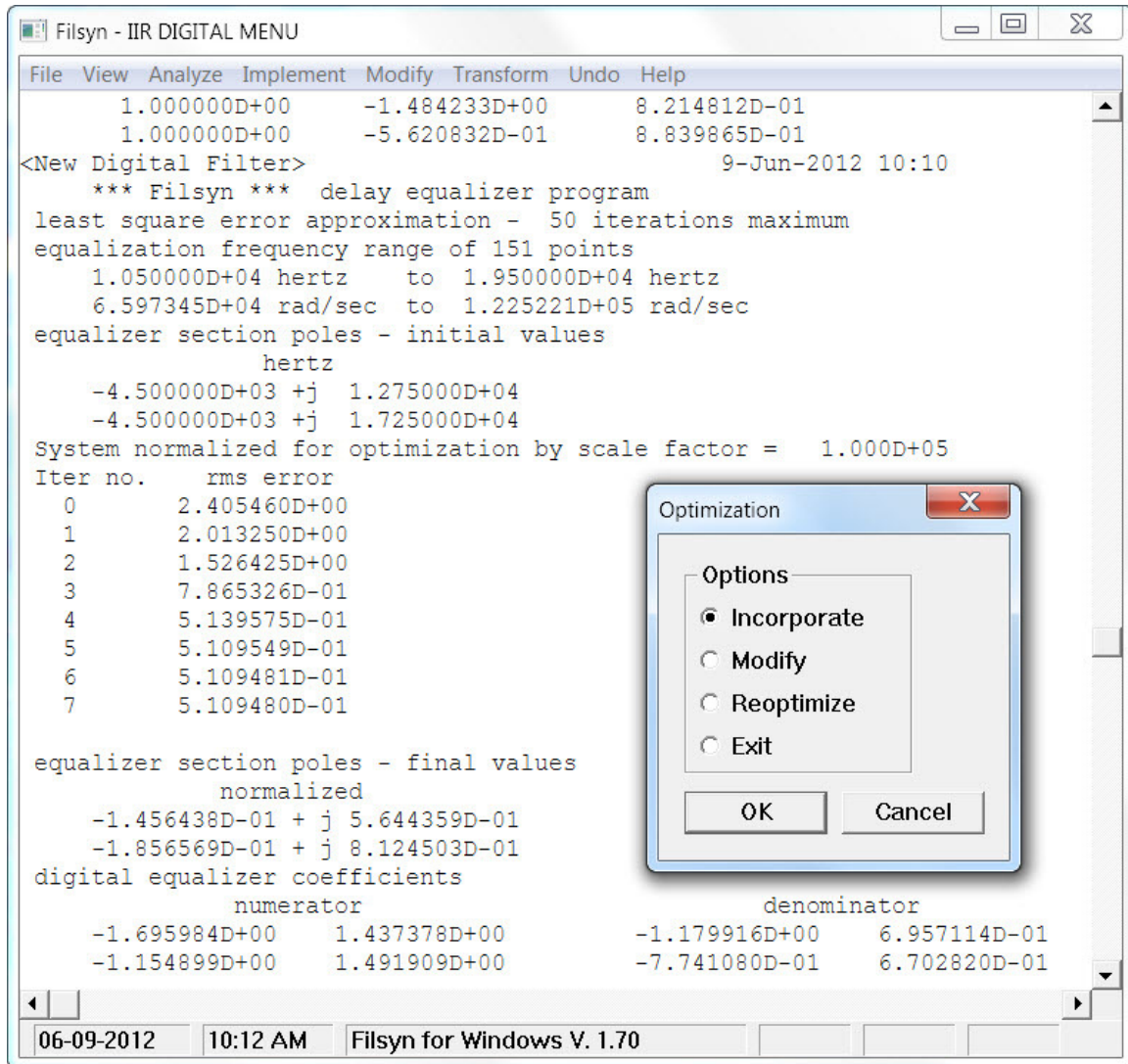
At the bottom are "OK" and "Cancel" buttons.

Note that in order to use the equalization procedure, the filter has to be analyzed in the frequency domain prior to invoking the option.

The optimization needs only a few iterations and follows up with the offer to tabulate the results, which we do not need:



The next prompt offers several options, including accepting the results and incorporating the equalizer into the filter:



The **Modify** option permits us to change some of the parameters, most significantly the number of sections used. In this case, we have accepted the design using the **Incorporate** option and displayed the new function:

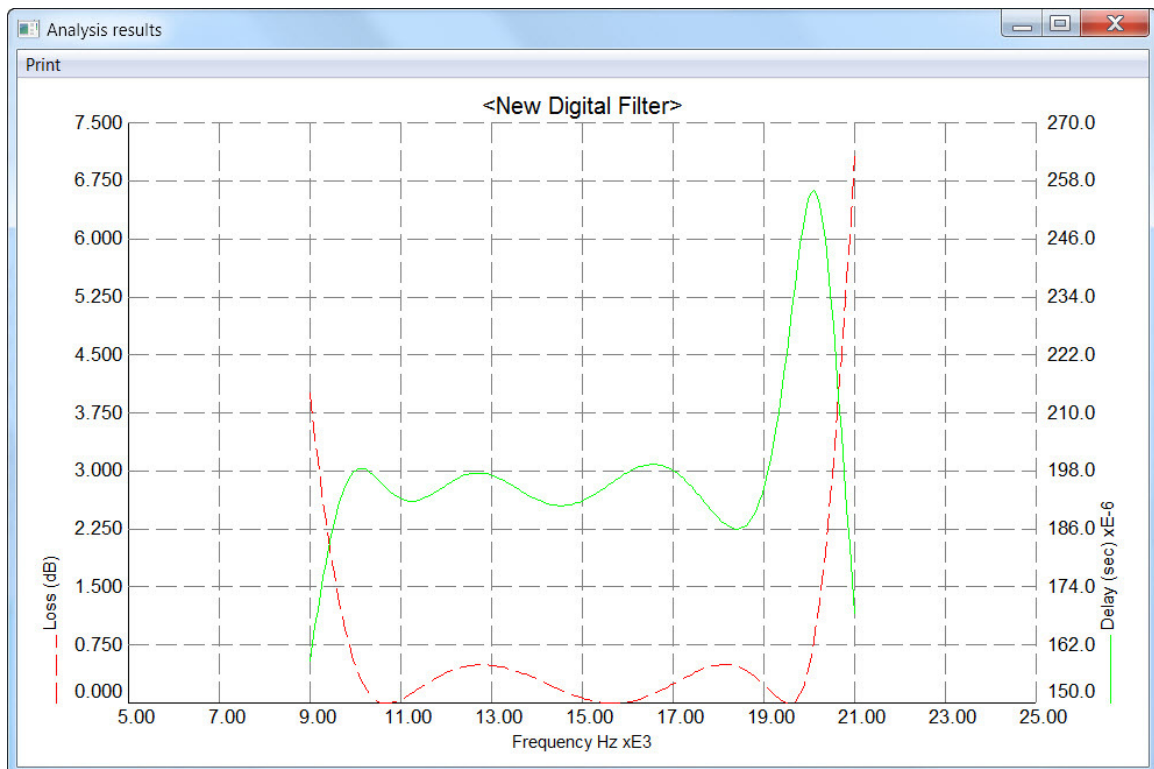

```

Filsyn - IIR DIGITAL MENU
File View Analyze Implement Modify Transform Undo Help
**IIR** digital filter transfer function
z transform used : bilinear
filter type      : bandpass
Front-end multiplier = 2.276207D-02

**** Cascade form ****
numerator coefficients decimal
z**( 0)      z**(-1)      z**(-2)
1.000000D+00  6.180340D-01    1.000000D+00
1.000000D+00 -1.902113D+00    1.000000D+00
1.000000D+00  0.000000D+00    1.000000D+00
1.000000D+00 -1.695984D+00    1.437378D+00
1.000000D+00 -1.154899D+00    1.491909D+00
denominator coefficients decimal
z**( 0)      z**(-1)      z**(-2)
1.000000D+00 -8.507724D-01    6.429536D-01
1.000000D+00 -1.484233D+00    8.214812D-01
1.000000D+00 -5.620832D-01    8.839865D-01
1.000000D+00 -1.179916D+00    6.957115D-01
1.000000D+00 -7.741080D-01    6.702820D-01
06-09-2012 10:14 AM Filsyn for Windows V. 1.70

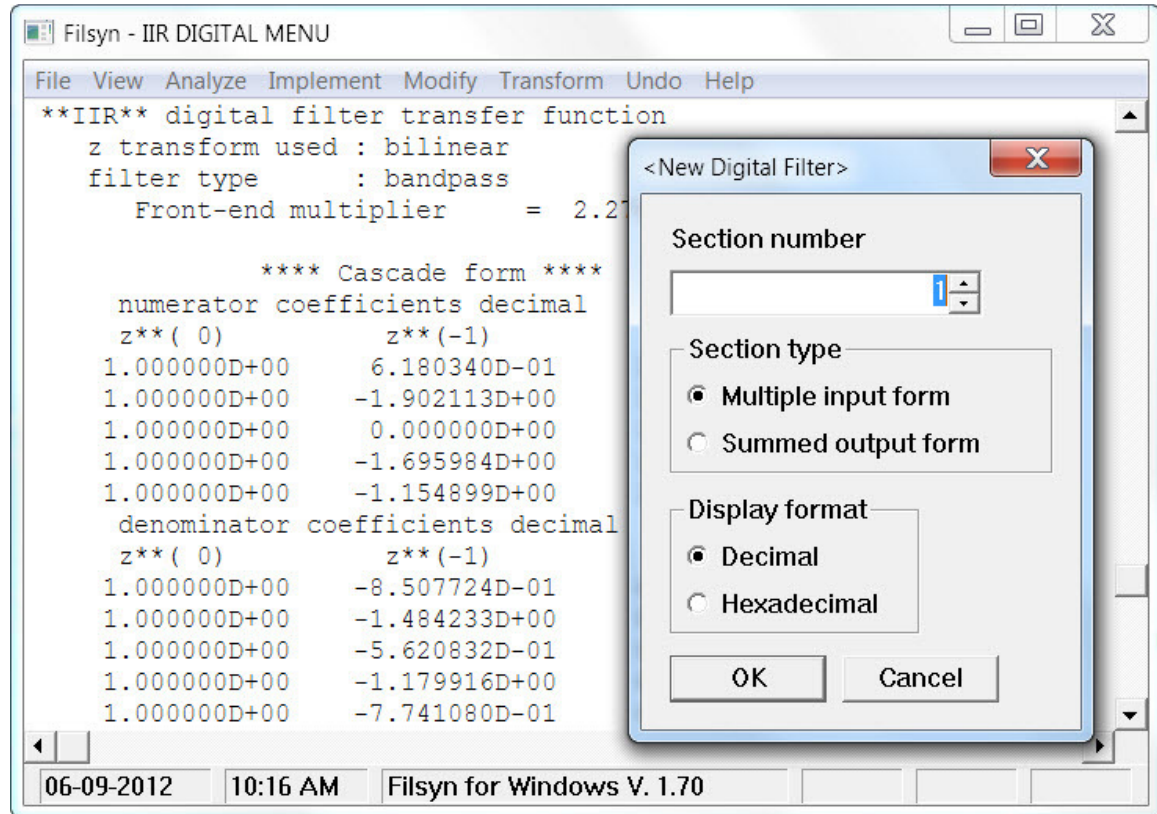
```

A new frequency domain analysis shows the performance of the equalizer:

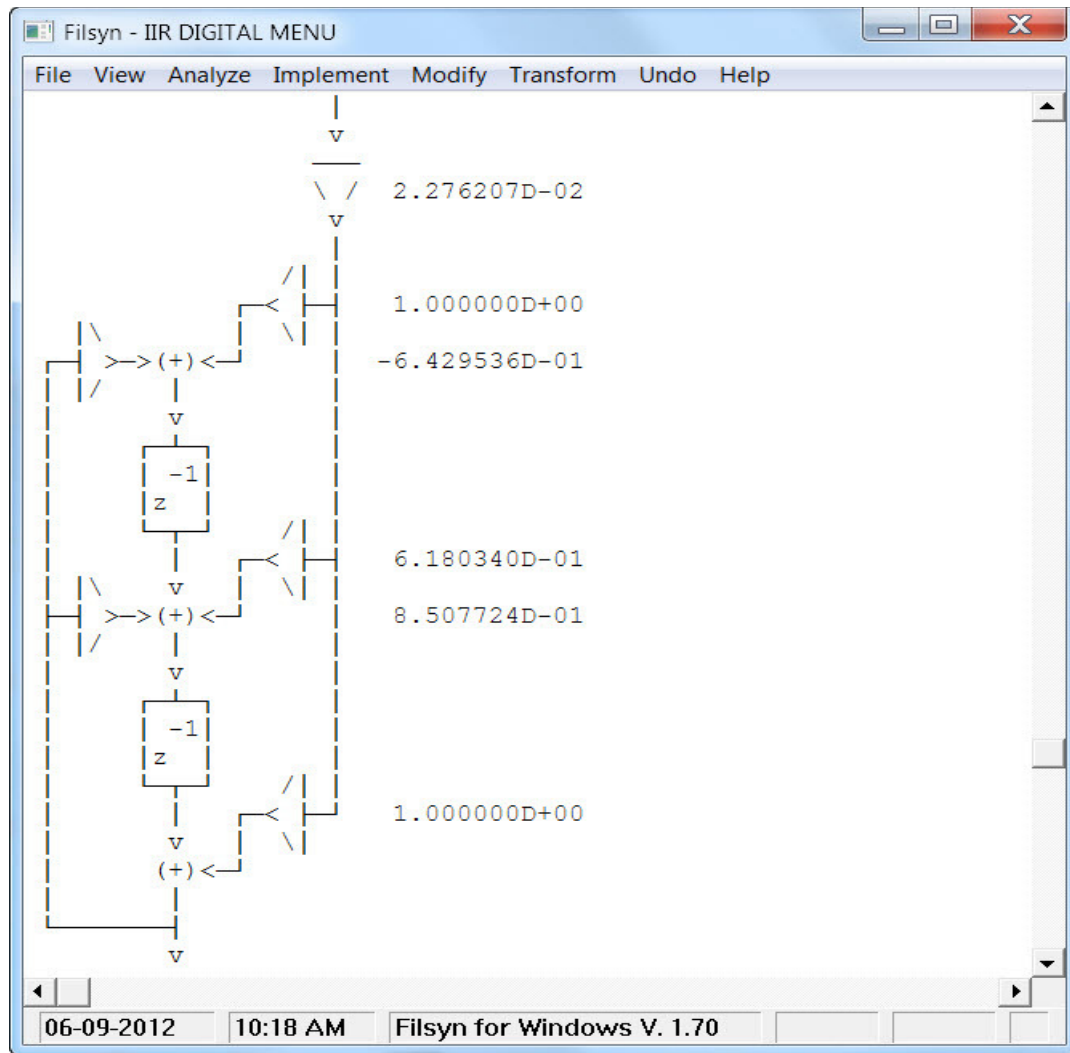


The optimization minimizes the sum of squared errors and as such, does not give us the equal ripple type results, but the results are still quite acceptable.

Next we demonstrate the **View->Design** menu option that generates the hardware implementation of the second order sections of either the cascade form or the parallel one (if generated). The option needs additional information:

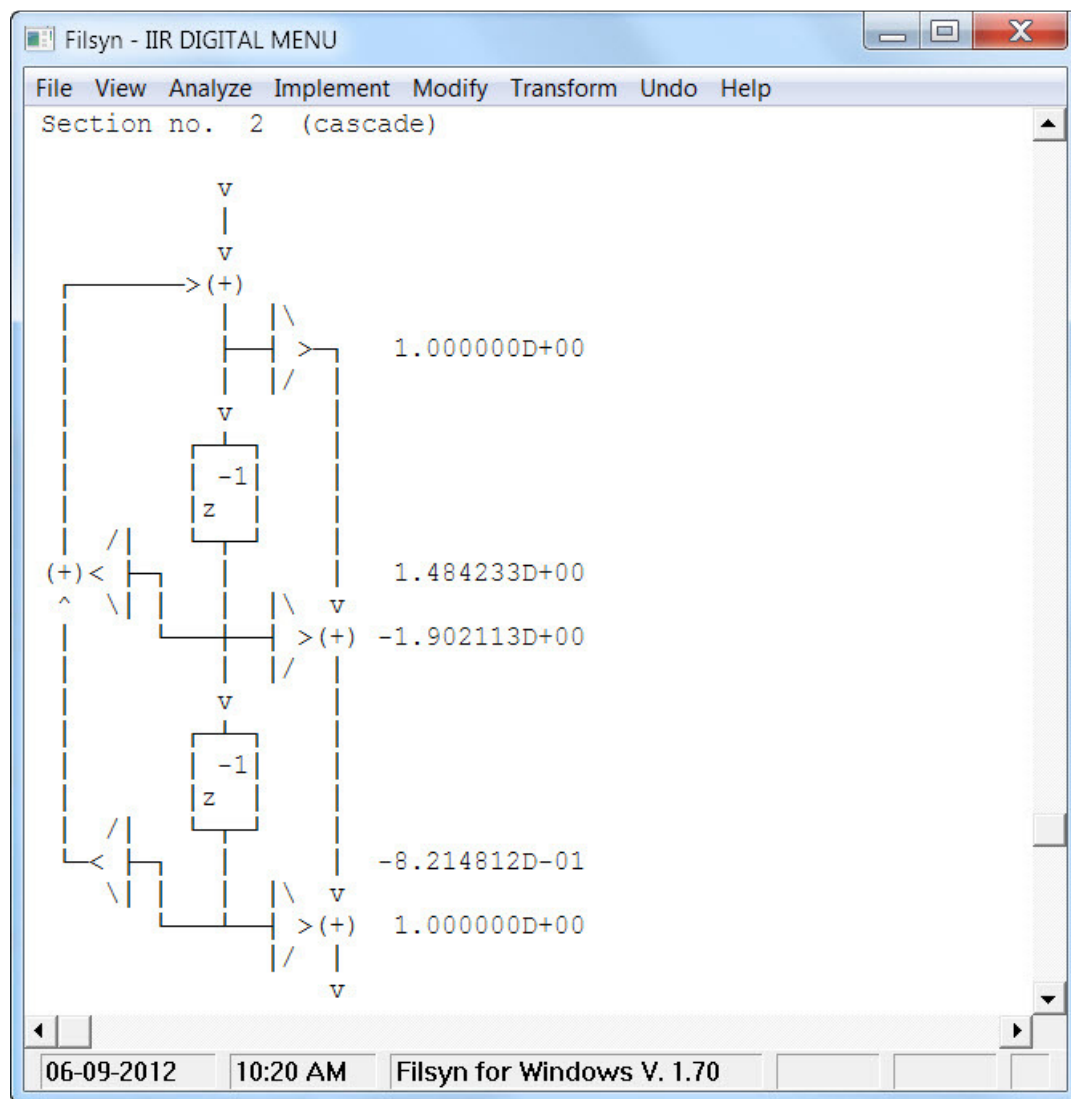


As we can see, there are two formats available and we selected the **Multiple input** form here. The coefficients may be displayed either in decimal or hexadecimal forms. The next prompt asks us if we wish to use the parallel implementation, since we have that form implemented here as well. Declining that option, we get the hardware implementation of the first section:

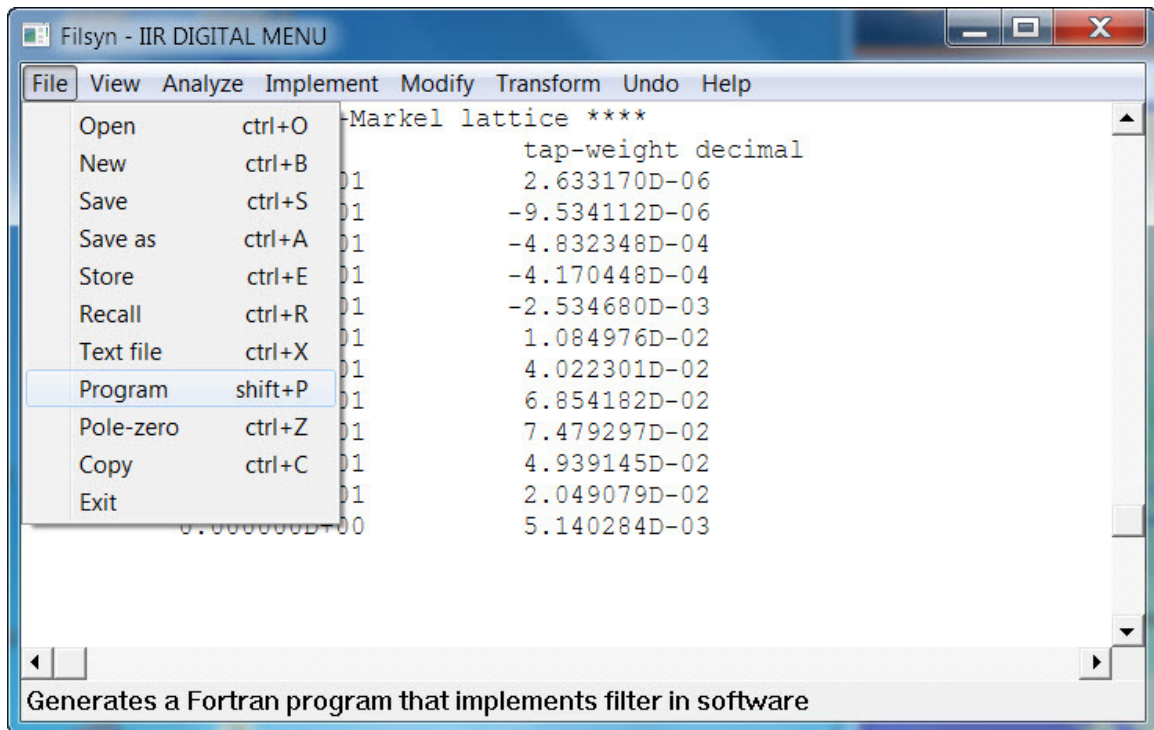


The other, **Summed output** form of implementation is demonstrated next using section 2 of the cascade form:

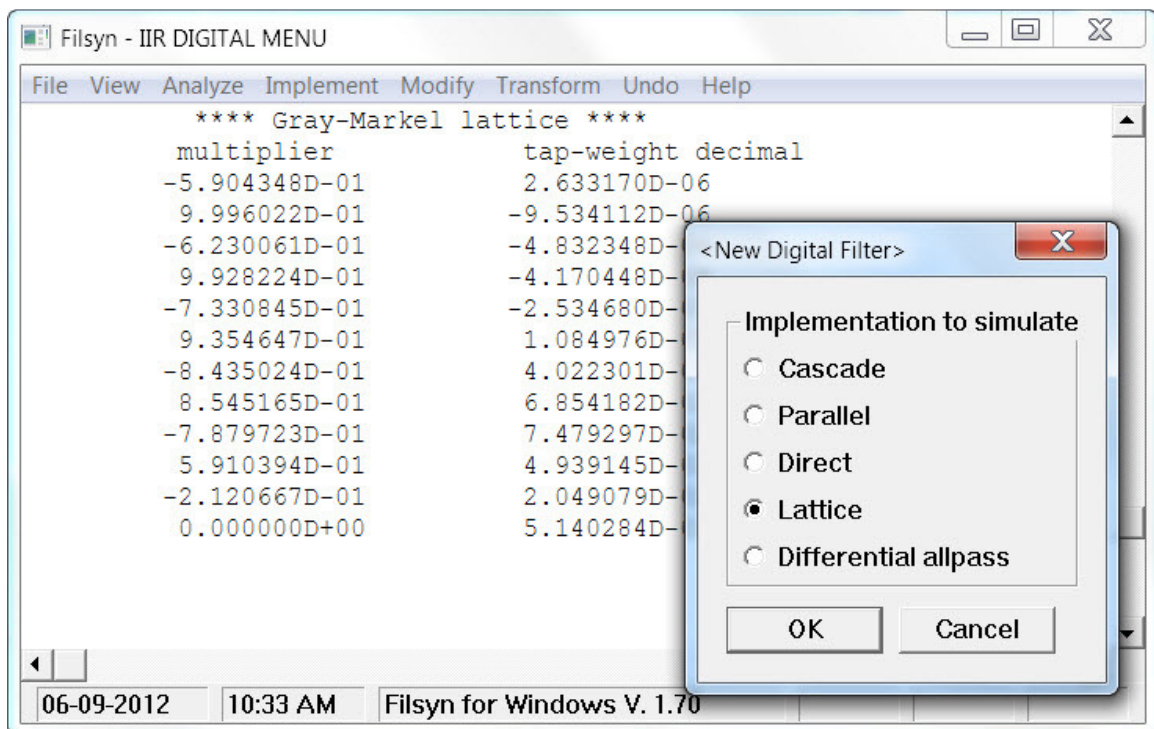
which yields:



We can also generate a software implementation of the filter in the form of a Fortran subroutine. This is available through the **File->Program** menu option:



which leads to the further menu options:



We selected the **Lattice** implementation, since we have generated that one also. If the selected implementation is not generated, the program displays an error message. The routine was saved to a text file called "ex02.f", which is shown on the next page:

```

subroutine tim(x, y, n, ibt)
real(len=8) gmul, gtap, x, y, z, z1, sum, zero, one, trun
dimension gmul(51), gtap(51), e(51), z(51), z1(51)
save
data gmul/ &
-5.90434763229D-01, 9.99602234814D-01, -6.23006122682D-01, &
9.92822422657D-01, -7.33084508594D-01, 9.35464730738D-01, &
-8.43502383435D-01, 8.54516506299D-01, -7.87972257698D-01, &
5.91039353759D-01, -2.12066707806D-01, 0.0000000000D+00, &
39*0.d0/
data gtap/ &
2.63317036143D-06, -9.53411178122D-06, -4.83234808731D-04, &
-4.17044797577D-04, -2.53468038526D-03, 1.08497639899D-02, &
4.02230095013D-02, 6.85418206065D-02, 7.47929658947D-02, &
4.93914540710D-02, 2.04907868038D-02, 5.14028417622D-03, &
39*0.d0/
data jm, jt, zero, one/11,12, 0.d0, 1.d0/
if (n == 1) then
do i = 1, 51
if (i == 1) then
z(i) = one
else
z(i) = gmul(i-1)
endif
z1(i) = zero
e(i) = one
end do
else
sum = zero
do i = 1, jt
sum = sum + trun(gtap(i)*z(i),ibt)
end do
y = trun(sum,ibt)
do i = 1, jt
z1(i) = z(i)
end do
e(1) = x
do i = 2, jt
e(i) = e(i-1) - trun(gmul(jt+1-i)*z1(jt+1-i),ibt)
end do
z(1) = e(jt)
do i = 2, jt
z(i) = z1(i-1) + trun(gmul(i-1)*e(jt+2-i),ibt)
end do
end if
return
end
function trun(x,i)
real(len=8) trun, x, a, b
if (iabs(i) >= 40) then
trun = x
else
a = dabs(x)
d = 0.5
if (i < 0) d = 0.0
b = 2.0**iabs(i)

```

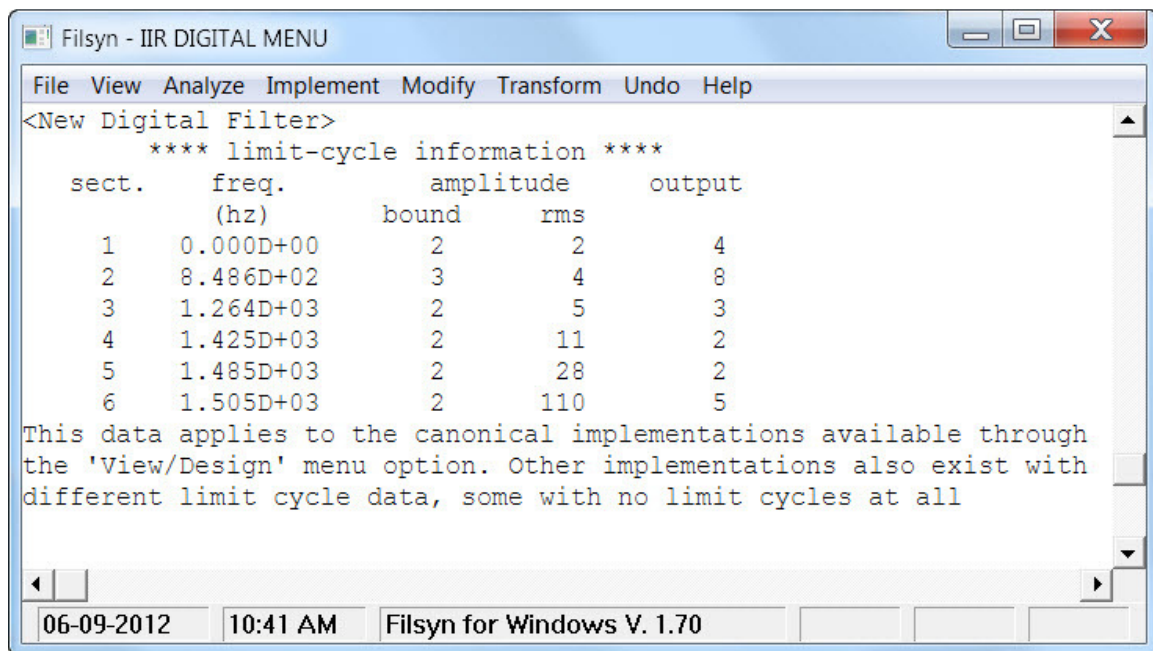
```

n = a*b + d
trun = float(n)/b
if (x < 0.d0) trun = -trun
endif
return
end

```

This is in Fortran90 format and of course, can be easily converted to many other languages.

Finally, let us see, what the **Analysis->Limit cycle** menu option yields:



9. FIR DIGITAL FILTERS

The design of FIR (Finite Impulse Response) digital filters is completely different from those of all other filters. This can be seen from the entirely different starting data entry screen, which we get to by selecting **Design->FIR filter->New** menu option:

There are two design methods available for us here, the first one **Equal ripple**, is the well known McClellan-Parks-Rabiner algorithm, which has been found unsurpassed in efficiency. The other, **Windowed**, is also well known and detailed descriptions of both methods can be found in any textbook on digital signal processing.

9.1 Linear Phase Equal Ripple Design

The theory of the linear phase FIR filter design method has been described in the literature in detail. The process is an iterative one and converges to an equal-ripple optimal design which is shown schematically in Fig.1, for a bandpass filter. Up to 10 frequency bands (passbands and stopbands) may be specified for filters. For differentiators and Hilbert transformers, a single band is automatically assumed. The bands are specified by their edge frequencies, the desired function value in the band, and a weight factor. If the desired function value is 1 (passband) or 0 (stopband), we may replace the weight factors by equivalent loss (dB) values.

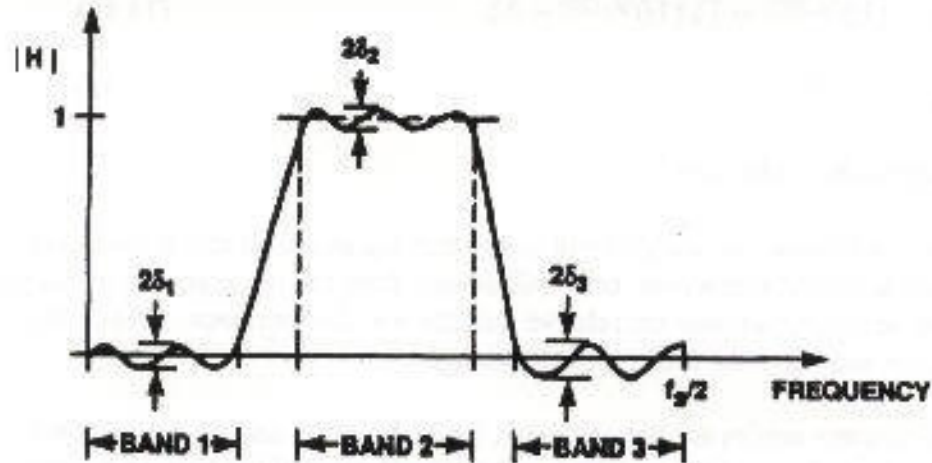


Fig.20 Optimal bandpass design

To understand the relationship between weights and loss values, consider the δ_i ripple values of Fig.1 and the passband loss ripple a_p or stopband minimum loss a_s . These are given in dB and they are related by the expressions:

Passband ripple:

$$a_p = 20 \log_{10} (1 + \delta_i)/(1 - \delta_i) \quad (1)$$

Stopband suppression:

$$a_s = -20 \log_{10} \delta_i \quad (2)$$

The filter length (number of taps N) is the most important design parameter and in most cases can be determined only by trial and error. For the special case of lowpass filters, one can estimate the length by an empirical expression, which is built into the program. It may be used by entering zero for the required filter length.

The maximum filter length in this program segment is 512 taps. Frequencies may be entered either in normalized form, in which case the highest frequency must be less than or equal to 0.5, or alternatively in Hz, in which case the last value entered will be considered to be half the sampling rate (the so-called *Nyquist* frequency).

If we request a filter, the **Details** button is active and we can enter the specification by clicking on it. The bands are specified by their band-edge frequencies, their function value (usually 1.0 for passbands and 0.0 for stopbands, but they can have any value) and ripple or weight value. Frequencies can be entered either all in Hz or all in normalized frequencies. For differentiators and Hilbert transformers, the number of bands is automatically set to one and the **Details** button becomes inactive. For Hilbert transformers the band may not start at 0.0 and for differentiators it may not extend to 0.5 (or the *Nyquist* frequency).

The ripple values are the passband ripple in the passbands or the minimum stopband loss in the stopbands. The relationship between these weights and loss values is given by the equations:

$$W_p = (10^{(a_p/20)} + 1)/(10^{(a_p/20)} - 1) \quad (3a)$$

and

$$W_s = 10^{(a_s/20)} \quad (3b)$$

in the passbands and stopbands, respectively, where a_p is the passband loss ripple and a_s is the stopband minimum loss.

If the length of the filter is adequate, the weights will insure that the passband and stopband requirements are met simultaneously. Otherwise, both will deviate from the requirements by varying amounts. Also, it should be noted that only the relative weights are of importance. Multiplying all of them by the same factor will leave the final results unchanged.

The **Shaped passband** button permits us to design a filter with a passband that is not flat, but has a prescribed shape, described by a tabulated set of loss values versus frequency. An example will illustrate this later on.

9.2 Equal ripple design output

Most of the output of this segment is self-explanatory. Heading and labeling are followed by a couple of explanatory items and then by the listing of coefficients (impulse response data) in both decimal and hexadecimal formats. Decimals are printed to eight places, hexadecimal to 10. If we so desire, we may also obtain a plot of the impulse and step responses.

Next, we are given a band-by-band performance summary, containing the normalized band edge frequencies, desired function value and weight, maximum deviation from desired value (δ_i values in Fig. 20) and the corresponding passband ripple or stopband loss values, all in dB. At the conclusion of this tabulation we automatically enter the analysis and manipulation segment of the program. Before considering that segment, we will illustrate the design of equal ripple filters by the use of examples.

9.3 Example 1: Lowpass Filter

We will design a lowpass filter with passband up to 1500 Hz and passband ripple of 1.3 dB. The stopband starts at 2000 Hz and goes all the way to 5000 Hz, which is the *Nyquist* frequency, that is to say, the sampling frequency is 10 kHz. We need 62 dB minimum stopband loss at the *Nyquist* frequency. If we specify a filter length of zero (which we may only do for equal-ripple lowpass filters), that indicates to the program that it needs to estimate the necessary length. The starting input screen input is exactly like the first shot shown at the beginning of this chapter except the length while the **Detail** input screen has the data:

	Lower band edge (KHz)	Upper band edge (KHz)	Function value	Ripple or weight
1	0.00000	1.500000E+00	1.00000000	1.30000000
2	2.000000E+00	5.000000E+00	0.00000000	62
3	0.00000	0.00000	0.00000000	0.00000000
4	0.00000	0.00000	0.00000000	0.00000000
5	0.00000	0.00000	0.00000000	0.00000000
6	0.00000	0.00000	0.00000000	0.00000000
7	0.00000	0.00000	0.00000000	0.00000000
8	0.00000	0.00000	0.00000000	0.00000000
9	0.00000	0.00000	0.00000000	0.00000000
10	0.00000	0.00000	0.00000000	0.00000000

OK Read from file Cancel

The data input can be entered manually, or read in from a text file with a .dat extension. The results indicate that the program estimated the need for a 39 tap filter:

```

Filsyn - FIR DIGITAL MENU
File View Analyze Modify Transform Undo Help
    Finite impulse response (FIR)
    Linear phase digital filter design
<New FIR Filter>                                     13-Jun-2012 10:12
    Remez exchange algorithm
    Lowpass filter
    Filter length = 39
    Filter length determined by approximation
    ***** Impulse response *****
        Function no. 0
    Sampling frequency = 1.0000000D+04

        decimal                                hexadecimal
h( 1) = -3.8899764D-03 = h( 39) = -0.00feef96
h( 2) = -1.0016633D-02 = h( 38) = -0.029073362
h( 3) = -1.4786241D-02 = h( 37) = -0.03c907f40
h( 4) = -1.2089422D-02 = h( 36) = -0.03184ad82
h( 5) = -4.9069988D-04 = h( 35) = -0.00202893f
h( 6) = 1.2491771D-02 = h( 34) = 0.0332a923f
h( 7) = 1.4679576D-02 = h( 33) = 0.03c20a6ba
h( 8) = 1.1200594D-03 = h( 32) = 0.0049677ab
h( 9) = -1.7484462D-02 = h( 31) = -0.0479dc97a
h(10) = -2.1264123D-02 = h( 30) = -0.057190c9b
h(11) = -6.7335520D-04 = h( 29) = -0.002c21069
h(12) = 2.8865289D-02 = h( 28) = 0.0763b72ff
h(13) = 3.5628575D-02 = h( 27) = 0.091ef44b4
h(14) = 1.8473735D-03 = h( 26) = 0.007911c8f
h(15) = -5.0423330D-02 = h( 25) = -0.0ce88b17a
h(16) = -6.6522875D-02 = h( 24) = -0.1107a4a4c
h(17) = -1.4000890D-03 = h( 23) = -0.005bc1989
h(18) = 1.3583579D-01 = h( 22) = 0.22c6226f5
h(19) = 2.7624825D-01 = h( 21) = 0.46b834914
h(20) = 3.3559677D-01 = h( 20) = 0.55e9ab778

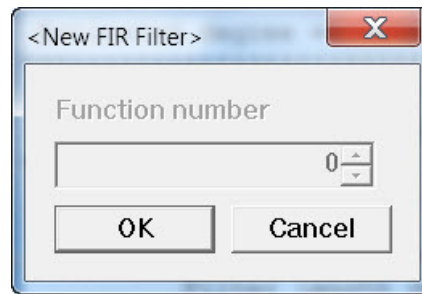
        band 1                                band 2
Lower band edge    0.0000000000    0.2000000000
Upper band edge    0.1500000000    0.5000000000
Desired value      1.0000000000    0.0000000000
Weighting          1.3000000000    62.0000000000
Deviation          0.062702782    0.001314736
Deviation in db    1.090690017    -57.623229980

06-13-2012 10:13 AM Filsyn for Windows V. 1.70

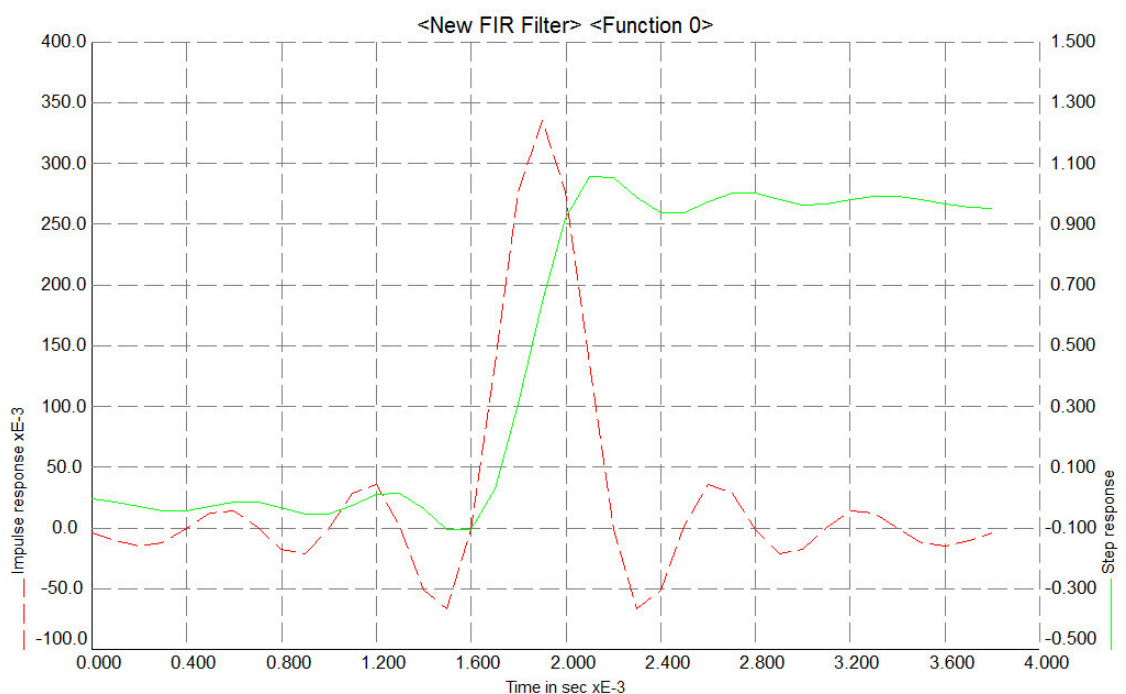
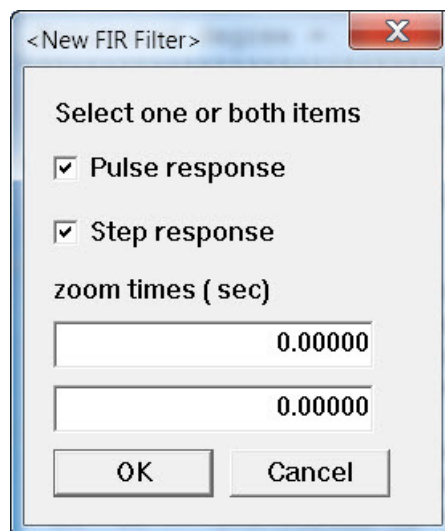
```

The *function no. 0* legend, shown immediately above the listing, is printed to distinguish it from other functions which may be computed later. The performance summary at the bottom indicates that the estimated filter length does not quite make it. We would probably need a 40 - 41 tap design to meet the requirements.

We now can have this impulse response as well as the corresponding step response plotted. Even though the impulse response is known, the step response is not hence we must invoke the **Analyze->Time** menu option, which leads to the prompt:

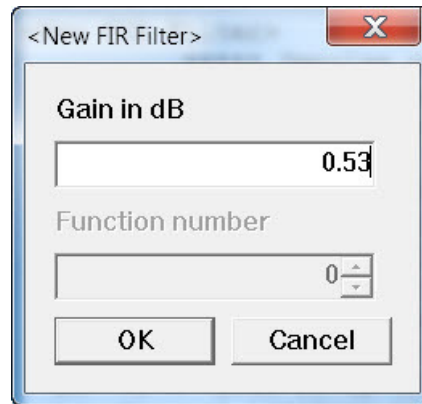


Since there is only one function available, we can only accept or cancel the operation. No time specification is needed here either. Declining the offer to tabulate the results, the plot is then invoked by the **Analyze->Plot->Time** menu option and can also be printed as usual.

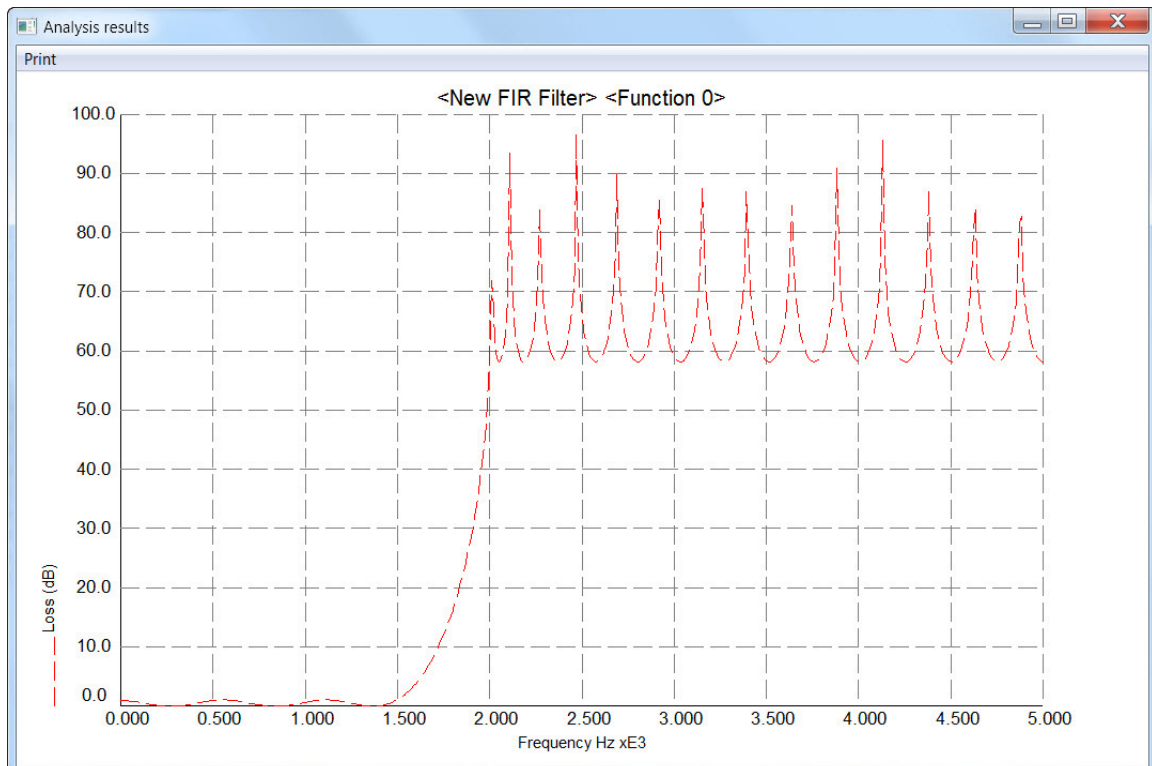


We are now in the analysis segment with its own set of menu options. However, we will postpone any discussion of this second segment and its functions until later in this manual. What we want to do now is to save the data on a permanent file for later use. We use the **File->Save** menu option to save the data in a file called “*example1*”. The extension is automatically selected as always.

Next we do a frequency domain analysis over the full band from zero to 5 KHz, but when we plot the loss, we find that the display has an awkward loss scale. This is due to the fact that in the passband the loss has a small negative value of -0.53 dB. We can easily correct for that using the **Modify->Flat gain** menu option:



and repeat the analysis. This time the plot is more convenient and shows the correct behavior:



9.4 Example 2: Band-Elimination Filter

Instead of repeating the previous design with 41 taps, we will design next a 25 tap band-elimination filter. We opt for weights instead of loss and arbitrarily select the stopband weight of 20 with reference to a passband weight of unity. The stopband is from 2 KHz to 3 KHz with a 10 KHz sampling rate:

	Lower band-edge (KHz)	Upper band-edge (KHz)	Function value	Ripple or weight
1	0.00000	1.500000E+00	1.00000000	1.00000000
2	2.000000E+00	3.000000E+00	0.00000000	20.00000000
3	3.500000E+00	5.000000E+00	1.00000000	1
4	0.00000	0.00000	0.00000000	0.00000000
5	0.00000	0.00000	0.00000000	0.00000000
6	0.00000	0.00000	0.00000000	0.00000000
7	0.00000	0.00000	0.00000000	0.00000000
8	0.00000	0.00000	0.00000000	0.00000000
9	0.00000	0.00000	0.00000000	0.00000000
10	0.00000	0.00000	0.00000000	0.00000000

OK Read from file Cancel

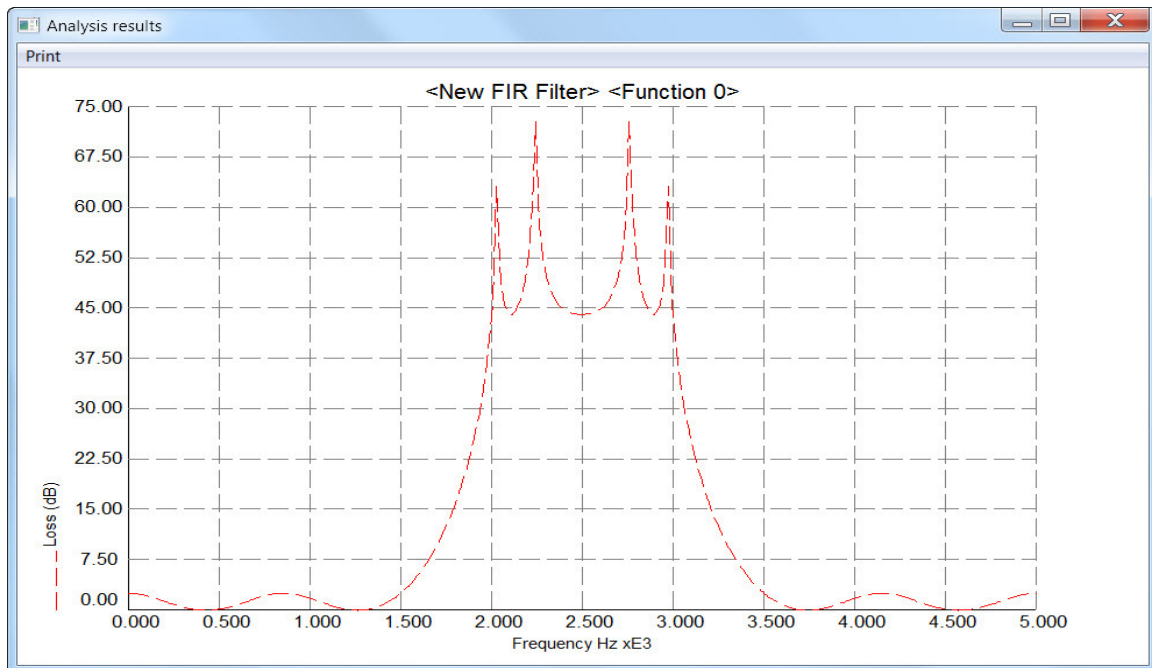
The results are shown next:

```

Filsyn - FIR DIGITAL MENU
File View Analyze Modify Transform Undo Help
Finite impulse response (FIR)
Linear phase digital filter design
<New FIR Filter> 10-Jun-2012 8:48
Remez exchange algorithm
Bandpass filter
Filter length = 25
***** Impulse response *****
Function no. 0
Sampling frequency = 1.0000000D+04
decimal hexadecimal
h( 1) = -4.4900649D-02 = h( 25) = -0.0b7e9be2d
h( 2) = -6.5037493D-06 = h( 24) = -0.00006d1d6
h( 3) = -6.1626039D-02 = h( 23) = -0.0fc6b9604
h( 4) = -1.3841786D-05 = h( 22) = -0.0000e83a0
h( 5) = 5.8553686D-02 = h( 21) = 0.0efd5fd5c
h( 6) = -1.6099586D-05 = h( 20) = -0.00010e1b3
h( 7) = 3.2906170D-03 = h( 19) = 0.00d7a7649
h( 8) = -1.0120123D-05 = h( 18) = -0.0000a9c9a
h( 9) = -1.3312230D-01 = h( 17) = -0.22144da6d
h(10) = -5.2415375D-06 = h( 16) = -0.000057f04
h(11) = 2.7048137D-01 = h( 15) = 0.453e444d5
h(12) = -2.7768446D-06 = h( 14) = -0.00002e967
h(13) = 6.7044703D-01 = h( 13) = 0.aba26ab81
band 1 band 2 band 3
Lower band edge 0.000000000 0.200000000 0.350000000
Upper band edge 0.150000000 0.300000000 0.500000000
Desired value 1.000000000 0.000000000 1.000000000
Weighting 1.000000000 20.000000000 1.000000000
Deviation 0.144308776 0.007215439 0.144308776
Deviation in db 2.524523258 -42.834743500 2.524523258
06-10-2012 8:53 AM Filsyn for Windows V. 1.70

```

At the end of this printout we again find ourselves in the analysis segment. We will therefore analyze the filter in the frequency domain. Before we do that, we use the **Transform->Scale** menu item to magnitude scale the function so that the minimum loss is reset to zero. This is an alternative to the **Modify->Flat gain** option we used before, which is somewhat more general. After that we do a frequency domain analysis and display the result graphically:



These results indicate an acceptable filter with linear phase and constant delay. The delay is not shown, since it is a constant. Incidentally, the delay calculation is exact, although an occasional 0./0. computation may cause some glitches. Furthermore, if the delay is, in fact, constant, it may not be plotted and that fact will be indicated if we attempt it.

9.5 Example 3: Hilbert Transformer

This example is a 30 tap Hilbert transformer. The input is even briefer than usual since very little is needed:

<New FIR Filter>

Filter length
30

Design method
☒ Equal ripple
☐ Windowed

Type
☐ Filter
☐ Differentiator
☒ Hilbert transformer

Specification type
☒ Ripple in dB
☐ Weights

Filter type
☒ Lowpass
☐ Highpass
☐ Bandpass
☐ Bandreject

Lower passband freq (KHz)
500.000000E-03

Upper passband freq (KHz)
0.000000

Sampling frequency (KHz)
10

Slope (dB/oct)
0.00000000

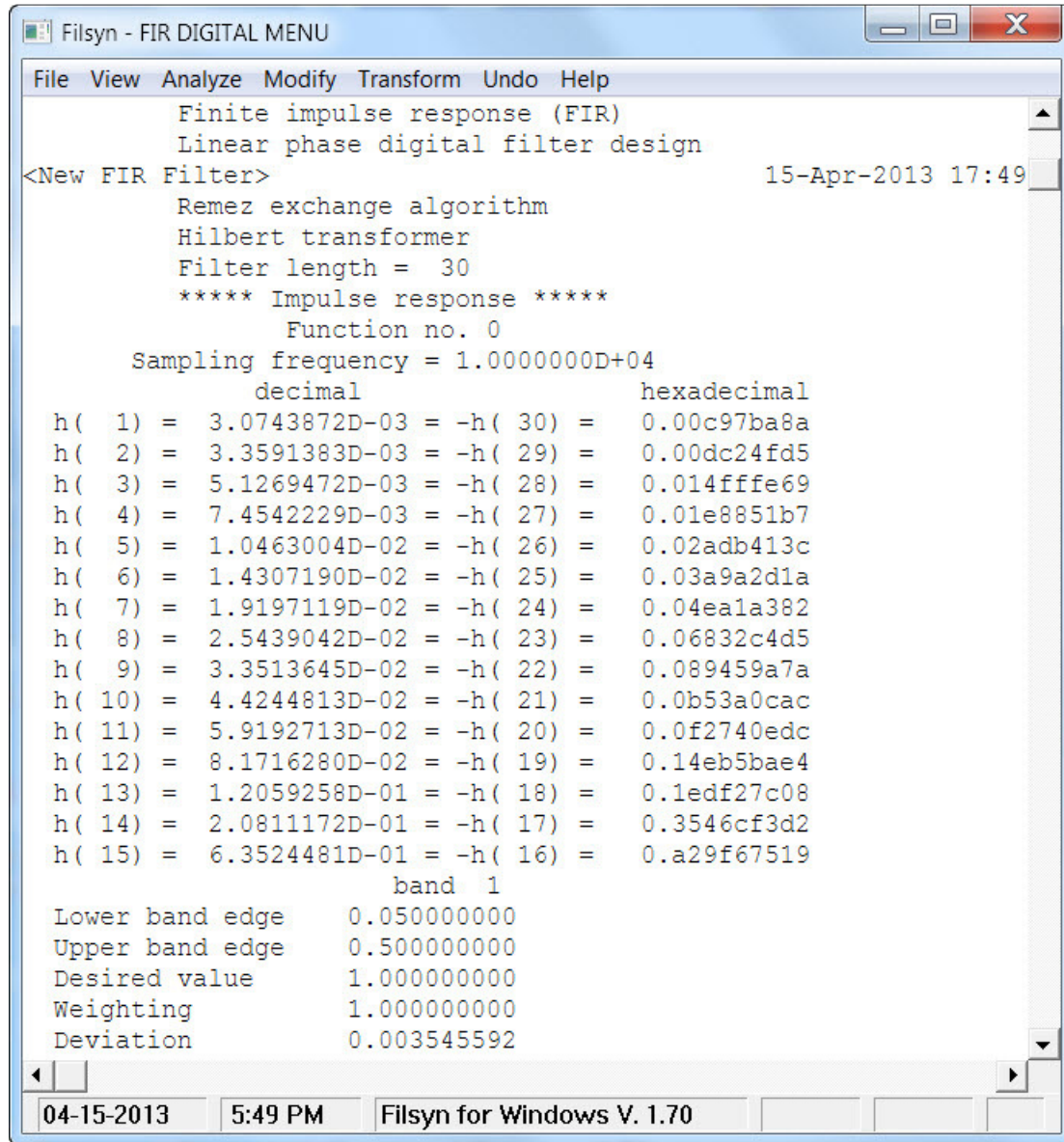
Window type
☐ Rectangular
☒ Triangular
☐ Hamming
☐ Hann
☐ Cosine
☐ Cos**3
☐ Cos**4
☐ Blackman
☐ 'Exact' Blackman
☐ 3-term B-H
☐ 3-term 18dB/oct
☐ Opt. 3-term
☐ 4-term B-H
☐ 4-term 40dB/oct
☐ 4-term 18dB/oct
☐ Opt. 4-term
☐ Kaiser
☐ Chebyshev
☐ Gaussian
☐ Taylor
☐ Sampled Kaiser

Enter loss value or parameter
0.500000000

☐ Save design data

OK **Cancel**

The results are:

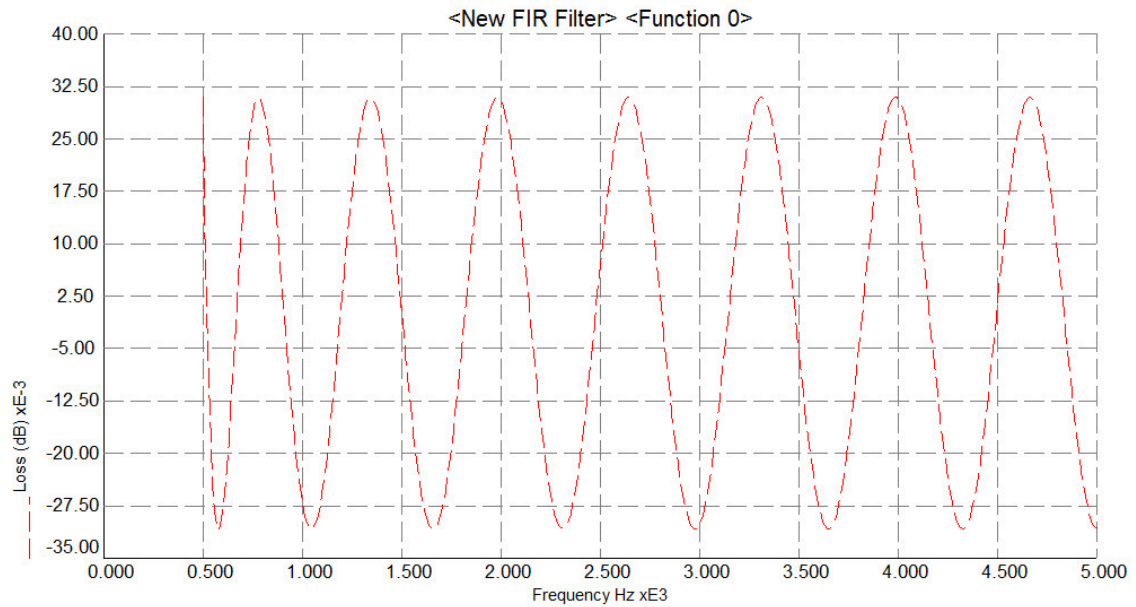


```

Filsyn - FIR DIGITAL MENU
File View Analyze Modify Transform Undo Help
Finite impulse response (FIR)
Linear phase digital filter design
<New FIR Filter> 15-Apr-2013 17:49
Remez exchange algorithm
Hilbert transformer
Filter length = 30
***** Impulse response *****
Function no. 0
Sampling frequency = 1.0000000D+04
decimal hexadecimal
h( 1) = 3.0743872D-03 = -h( 30) = 0.00c97ba8a
h( 2) = 3.3591383D-03 = -h( 29) = 0.00dc24fd5
h( 3) = 5.1269472D-03 = -h( 28) = 0.014fffe69
h( 4) = 7.4542229D-03 = -h( 27) = 0.01e8851b7
h( 5) = 1.0463004D-02 = -h( 26) = 0.02adb413c
h( 6) = 1.4307190D-02 = -h( 25) = 0.03a9a2d1a
h( 7) = 1.9197119D-02 = -h( 24) = 0.04ea1a382
h( 8) = 2.5439042D-02 = -h( 23) = 0.06832c4d5
h( 9) = 3.3513645D-02 = -h( 22) = 0.089459a7a
h(10) = 4.4244813D-02 = -h( 21) = 0.0b53a0cac
h(11) = 5.9192713D-02 = -h( 20) = 0.0f2740edc
h(12) = 8.1716280D-02 = -h( 19) = 0.14eb5bae4
h(13) = 1.2059258D-01 = -h( 18) = 0.1edf27c08
h(14) = 2.0811172D-01 = -h( 17) = 0.3546cf3d2
h(15) = 6.3524481D-01 = -h( 16) = 0.a29f67519
band 1
Lower band edge 0.050000000
Upper band edge 0.500000000
Desired value 1.000000000
Weighting 1.000000000
Deviation 0.003545592
04-15-2013 5:49 PM Filsyn for Windows V. 1.70

```

The analysis of this circuit does not show much, i.e., the magnitude is flat above 500 Hz with about 0.06 dB peak-to-peak variation. We plot the loss over the passband only, otherwise we could not see the variation.



9.6 Shaped passband

The passband of a lowpass filter is required to be a constant 0 dB from zero to 2 KHz and from 2 KHz to 3 KHz it should increase linearly from zero to 3 dB. This can be summarized in the table:

```
!Testing shaped filters
! freq      loss in dB
0.0 0.0
2000.0 0.0
2100.0 0.3
2200.0 0.6
2300.0 0.9
2400.0 1.2
2500.0 1.5
2600.0 1.8
2700.0 2.1
2800.0 2.4
2900.0 2.7
3000.0 3.0
```

that is in a text file called “test.dat”. Note that for a range of a constant value, we only need to specify the end points. Assuming a length of 32 taps and loss ripple in dB, we specify the pass and stop bands using the **Details** button as follows:

Band specification

	Lower band-edge (Hz)	Upper band-edge (Hz)	Function value	Ripple or weight
1	0.00000	3.00000E+03	1.00000000	1.30000000
2	4.00000E+03	5.00000E+03	0.00000000	62.0000000
3	0.00000	0.00000	0.00000000	0.00000000
4	0.00000	0.00000	0.00000000	0.00000000
5	0.00000	0.00000	0.00000000	0.00000000
6	0.00000	0.00000	0.00000000	0.00000000
7	0.00000	0.00000	0.00000000	0.00000000
8	0.00000	0.00000	0.00000000	0.00000000
9	0.00000	0.00000	0.00000000	0.00000000
10	0.00000	0.00000	0.00000000	0.00000000

OK Read from file Cancel

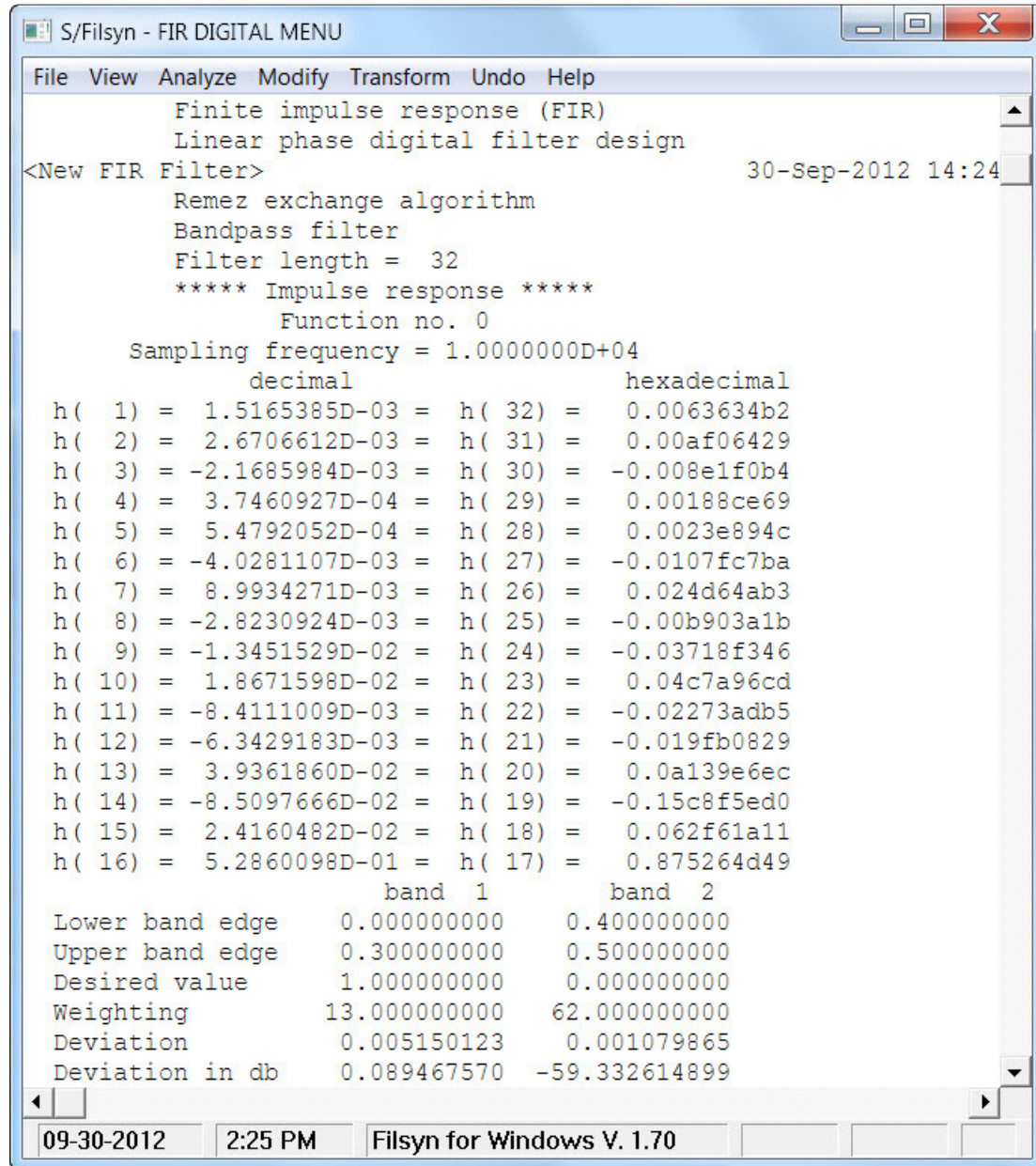
Next we need to enter the data from the file shown above. We click on the **Shaped passband** button and select the **Read from file** option and select the *test.dat* file from the available files. The results are as shown below:

Specified shape

	Frequency (Hz)	Loss (dB)
1	0.0000	0.0000
2	2000.0	0.0000
3	2100.0	0.30000
4	2200.0	0.60000
5	2300.0	0.90000
6	2400.0	1.2000
7	2500.0	1.5000
8	2600.0	1.8000
9	2700.0	2.1000
10	2800.0	2.4000
11	2900.0	2.7000

OK Read from file Cancel

The resulting filter data is:

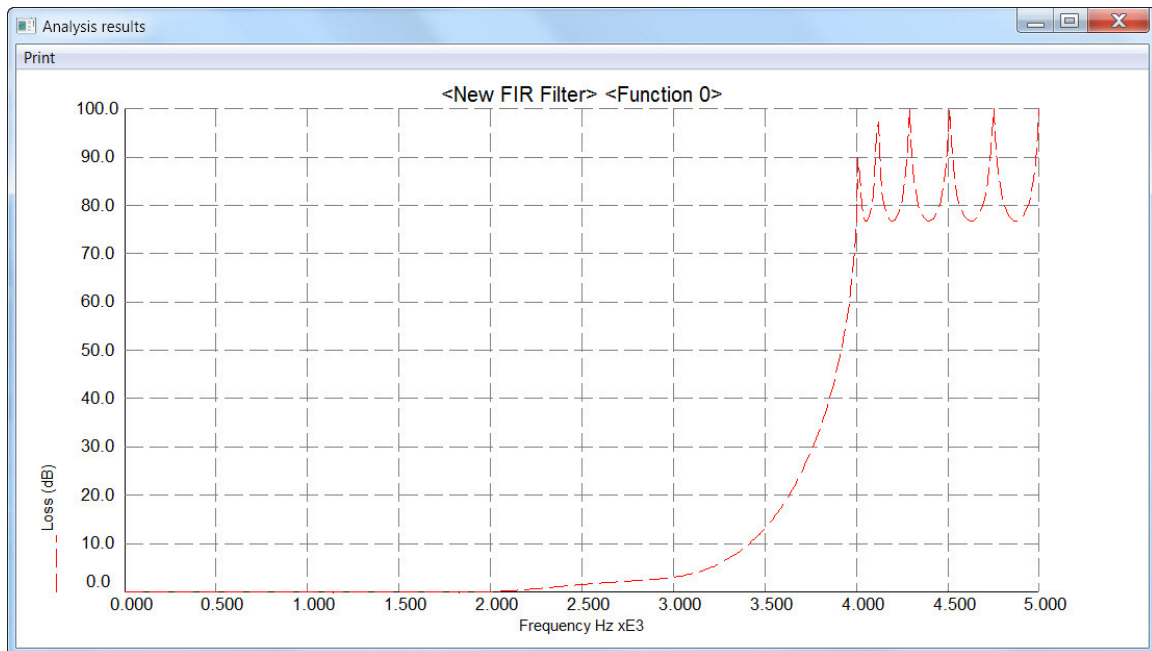


```

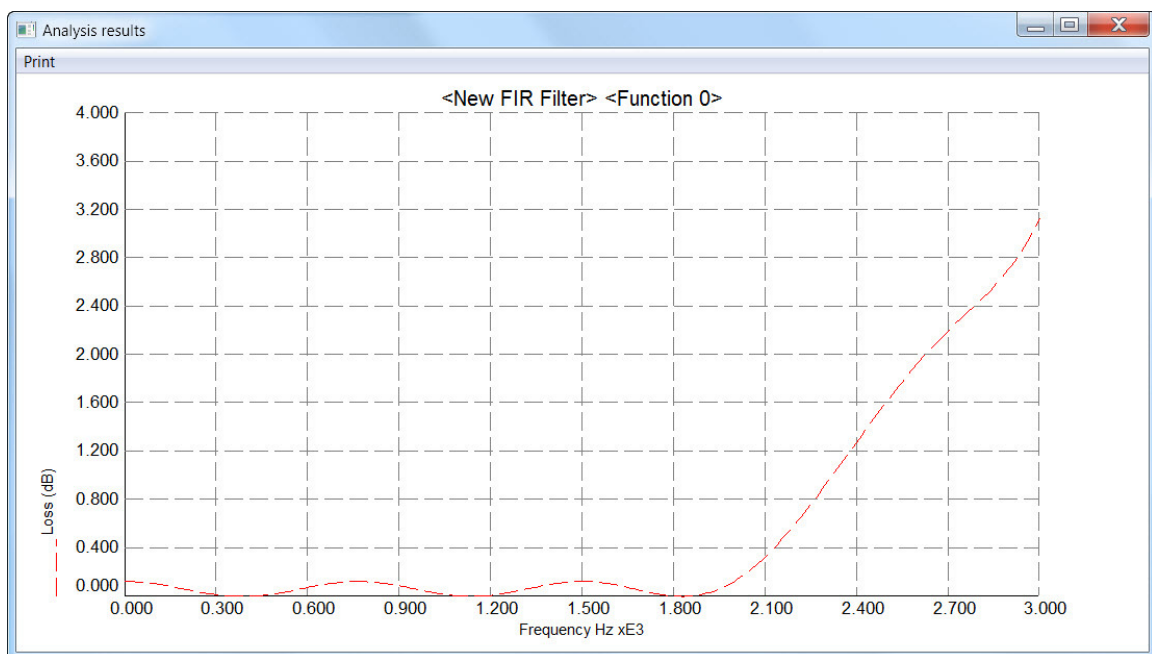
S/Filsyn - FIR DIGITAL MENU
File View Analyze Modify Transform Undo Help
Finite impulse response (FIR)
Linear phase digital filter design
<New FIR Filter> 30-Sep-2012 14:24
Remez exchange algorithm
Bandpass filter
Filter length = 32
***** Impulse response *****
Function no. 0
Sampling frequency = 1.0000000D+04
decimal hexadecimal
h( 1) = 1.5165385D-03 = h( 32) = 0.0063634b2
h( 2) = 2.6706612D-03 = h( 31) = 0.00af06429
h( 3) = -2.1685984D-03 = h( 30) = -0.008e1f0b4
h( 4) = 3.7460927D-04 = h( 29) = 0.00188ce69
h( 5) = 5.4792052D-04 = h( 28) = 0.0023e894c
h( 6) = -4.0281107D-03 = h( 27) = -0.0107fc7ba
h( 7) = 8.9934271D-03 = h( 26) = 0.024d64ab3
h( 8) = -2.8230924D-03 = h( 25) = -0.00b903a1b
h( 9) = -1.3451529D-02 = h( 24) = -0.03718f346
h(10) = 1.8671598D-02 = h( 23) = 0.04c7a96cd
h(11) = -8.4111009D-03 = h( 22) = -0.02273adb5
h(12) = -6.3429183D-03 = h( 21) = -0.019fb0829
h(13) = 3.9361860D-02 = h( 20) = 0.0a139e6ec
h(14) = -8.5097666D-02 = h( 19) = -0.15c8f5ed0
h(15) = 2.4160482D-02 = h( 18) = 0.062f61a11
h(16) = 5.2860098D-01 = h( 17) = 0.875264d49
band 1 band 2
Lower band edge 0.0000000000 0.4000000000
Upper band edge 0.3000000000 0.5000000000
Desired value 1.0000000000 0.0000000000
Weighting 13.0000000000 62.0000000000
Deviation 0.005150123 0.001079865
Deviation in db 0.089467570 -59.332614899
09-30-2012 2:25 PM Filsyn for Windows V. 1.70

```

Before displaying the analysis results graphically, it is advisable to run the **Transform -> Scale** menu item to make sure the loss is nowhere negative. The frequency domain analysis result is as expected:



and the details of the passband shows an excellent agreement with requirements:



9.7 Windowed design

As was mentioned above, we can also design linear phase digital FIR filters through the use of the familiar windowing technique. About 20 different windowing functions are built into the program. No discussion of the relative merits of these windows, or any comparison of the two techniques, will be offered here. However, the references in *Appendix A* are sufficient for that purpose.

We need only say that we may design lowpass, highpass, bandpass and bandreject filters with even fewer pieces of input needed, than for equal ripple design. We must enter the filter length (no estimate is available here), select its type, and specify the bandedge frequencies. Finally, we must select a window, and in a few cases, an associated parameter.

These entries are followed by the usual display of the impulse response but with no performance summary. In order to observe the performance, we must analyze the resulting filter.

9.8 Example 4: Windowed Bandpass

We will design a bandpass filter from 1000 Hz to 2500 Hz, with a sampling frequency of 10 kHz (i.e., a *Nyquist* rate of 5 kHz). We will use a Kaiser window with 32 taps and a 35 dB loss. The following is the data input screen:

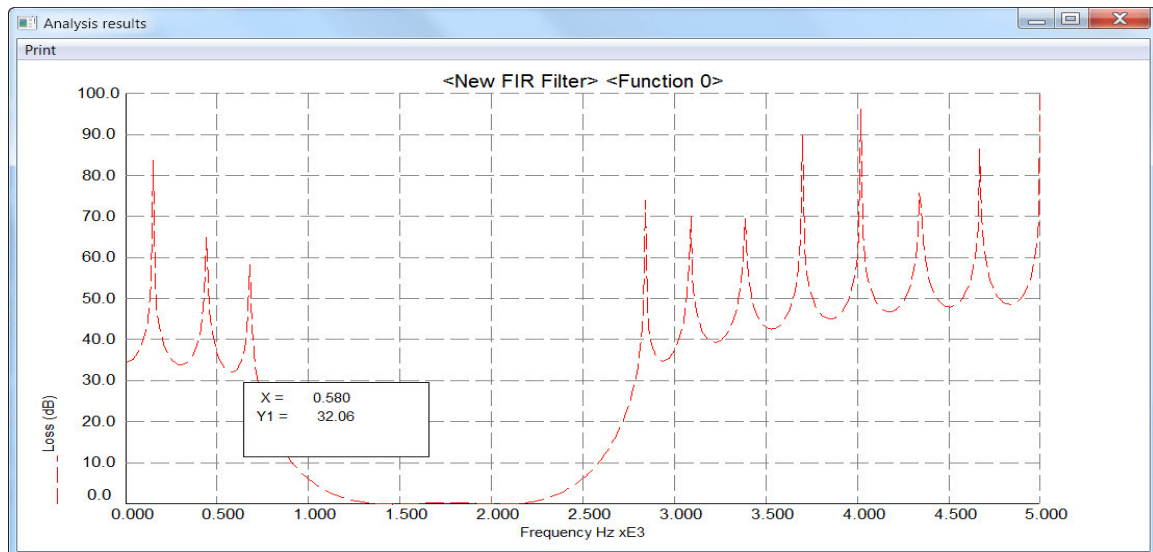
The screenshot shows the 'New FIR Filter' dialog box with the following settings:

- Filter length:** 32
- Design method:** ☒ Windowed
- Type:** ☒ Filter
- Specification type:** ☒ Ripple in dB
- Filter type:** ☒ Bandpass
- Lower passband freq (Hz):** 1.000000E+03
- Upper passband freq (Hz):** 2.500000E+03
- Sampling frequency (Hz):** 10.000000E+03
- Slope (dB/oct):** 0.00000000
- Window type:** ☒ Kaiser
- Enter loss value or parameter:** 35.0000000
- Save design data:** ☐

Buttons: Details, Shaped passband, OK, Cancel.

The results and the plotted frequency domain responses are shown next:

We have again removed a -0.2 dB flat loss from the filter using the **Transform->Scale** menu item in order to adjust the graphical display:



The results shown illustrate about a 6 dB loss at the band edges and a minimum of 35 dB stopband loss below about 500 Hz and above 2800 Hz. We also demonstrate above the way precise results may be displayed by moving the pointer close to the curve and holding the right mouse key down.

10. FIR DIGITAL ANALYSIS

10.1 Available operations

The FIR analysis segment takes a set of FIR filter coefficients, which were obtained either from the design segment or directly entered into the program, and manipulates them in many ways. The major possibilities are:

a) The transfer function may be factored into second order factors. This means that, if we are given the following impulse response coefficients h_k , we can calculate the coefficients a_i and b_i in the identity:

$$H(z) = \sum_{k=0}^{N-1} h_k z^{-k} = h_0 \prod_{i=1}^{[N/2]} (1 + a_i z^{-1} + b_i z^{-2}) \quad (1)$$

As an option, the factored form if we know it, can also be entered into this segment directly.

b) Any number of these factors can be remultiplied into up to five individual polynomial form impulse responses. This step is useful should we wish to break up a longer FIR filter into a cascade of two or more shorter ones. If we select the factors properly by keeping zeros and their inverses together, the individual components can also be made into linear phase functions, if the original was of a linear phase design.

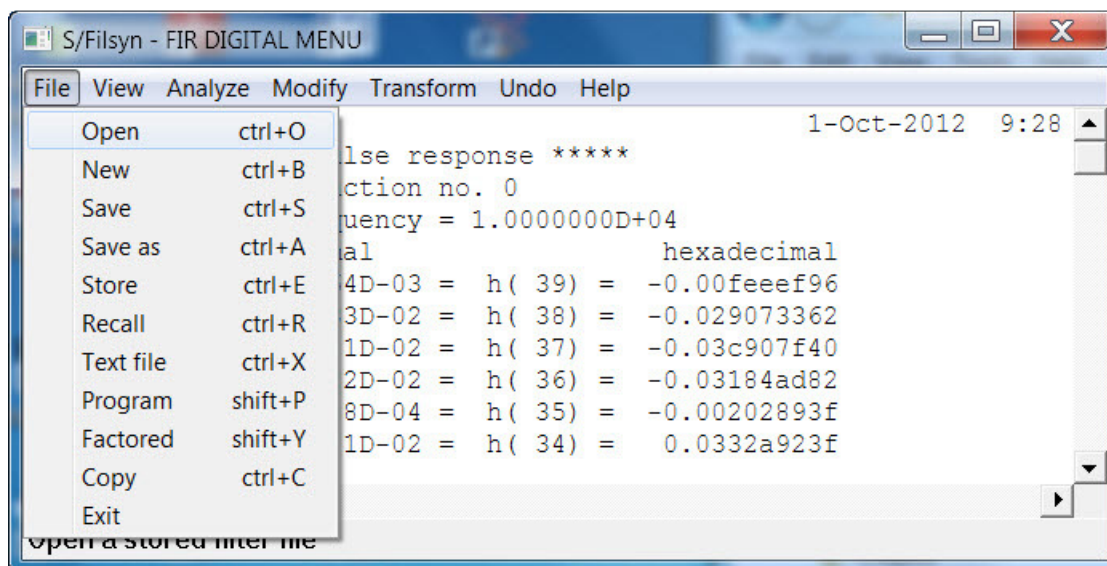
c) Before remultiplying, we may flip all the zeros of the factored transfer function located outside the unit circle in the z -plane, to their image location inside it, thereby converting the linear phase function into a minimum phase one.

Some other operations available at any stage are:

- coefficient truncation or rounding,
- frequency and time domain analysis and plotting,
- scaling,
- temporary or permanent storage and recall of data.

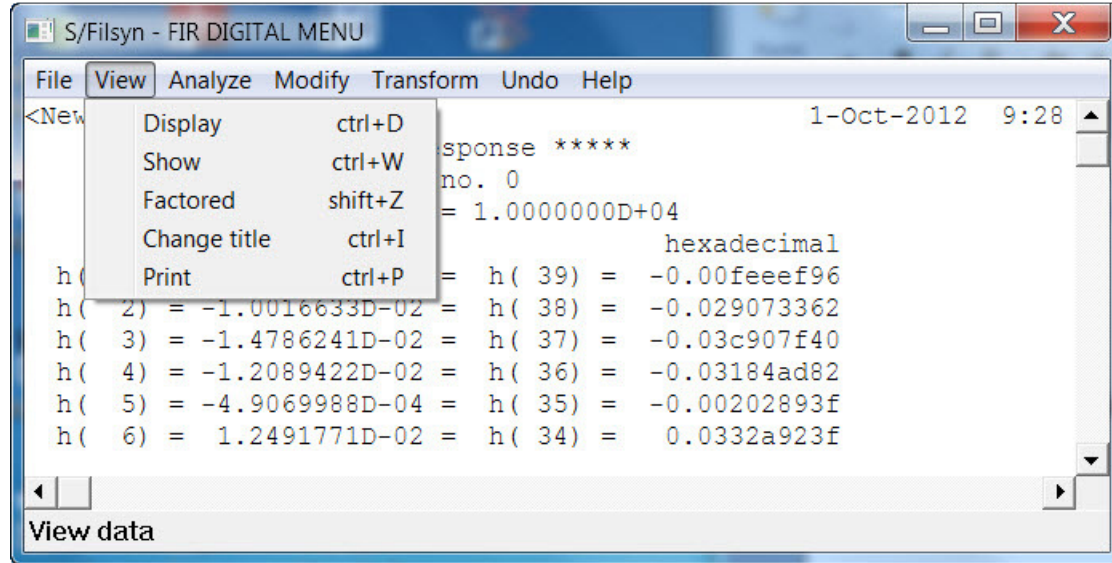
At this stage let us recall one of the previously saved designs (example1.qff) and see what menu options we have available and what they do.

10.2 The 'File' menu option has a number of submenu items available:



1. **Open** opens another save set of filter data and the current set would be overwritten.
2. **New** lets us restart the synthesis or start a completely new one without the need for stopping the program and restarting it.
3. **Save** and **Save as** do the usual thing, save the current design in the currently open file (if there is one open) or save it in a new permanent file.
4. **Store** and **Recall** saves the current set of data in a temporary file and recalls it from there. This temporary file is closed when we terminate the program and the data in it is then lost.
5. **Text file** save the filter data in a text file for documentation purposes only. This file cannot be recalled into **Filsyn**, but can be edited in any text editor or word processor.
6. **Program** creates a Fortran program that simulates the current filter and writes it to an editable text file.
7. **Factored** writes the factored function into a text file.
8. **Copy** does what we expect; you can select part of the output by the use of the mouse and then copy that to the clipboard.
9. Finally **Exit** terminates program execution, as usual.

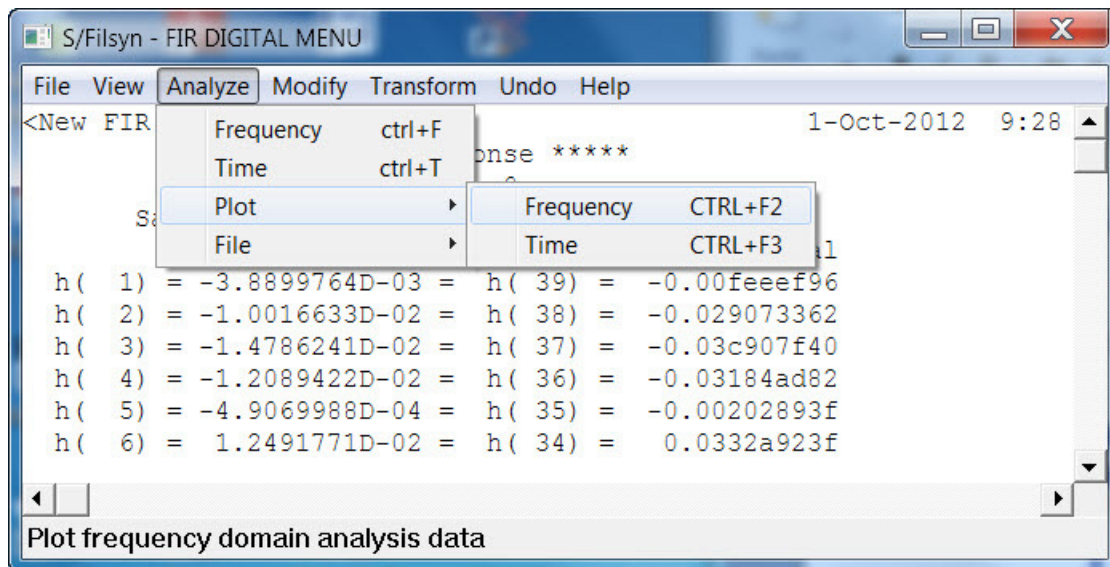
10.3 The 'View' menu option, we find another set of submenu items:



1. **Display** prints the filter specification to the window, while
2. **Show** displays the filter data itself.
3. **Factored** prints the factored form, if it exists, to the screen.
4. **Change title** does what it says, it gives you the opportunity to specify your own title. Finally,
5. **Print** is a switch that turns the printer on and off. It is off by default, but when it is on, everything that is printed to the screen, also goes to your printer, or to a text file.

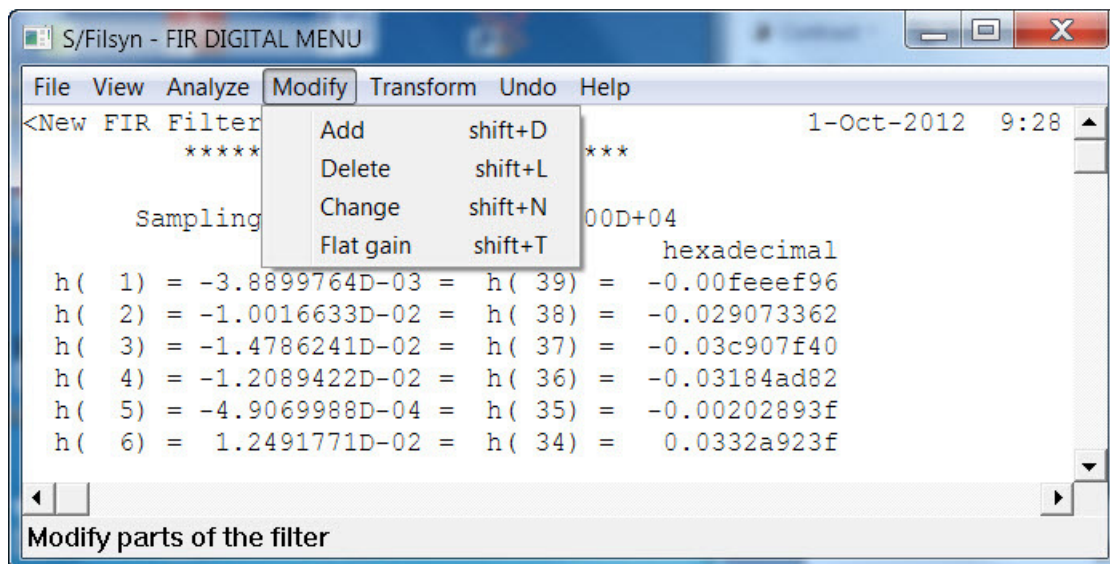
10.4 The 'Analyze' menu option has the usual submenus again:

1. **Frequency** and **Time** compute the corresponding responses. While the impulse response, being the $h(i)$ array displayed, needs no computation, the step response does, and if we have more than one function under consideration, we need a way to select the one to be used, hence the
2. **Time** option.
3. **Plot** of course allows us to display the computed responses graphically, while
4. **File** lets us print the computed responses to a text file for documentation purposes. Again, loss values in excess of 100 dB are truncated to 100 dB for the graphical display, but they are printed to the file unchanged.



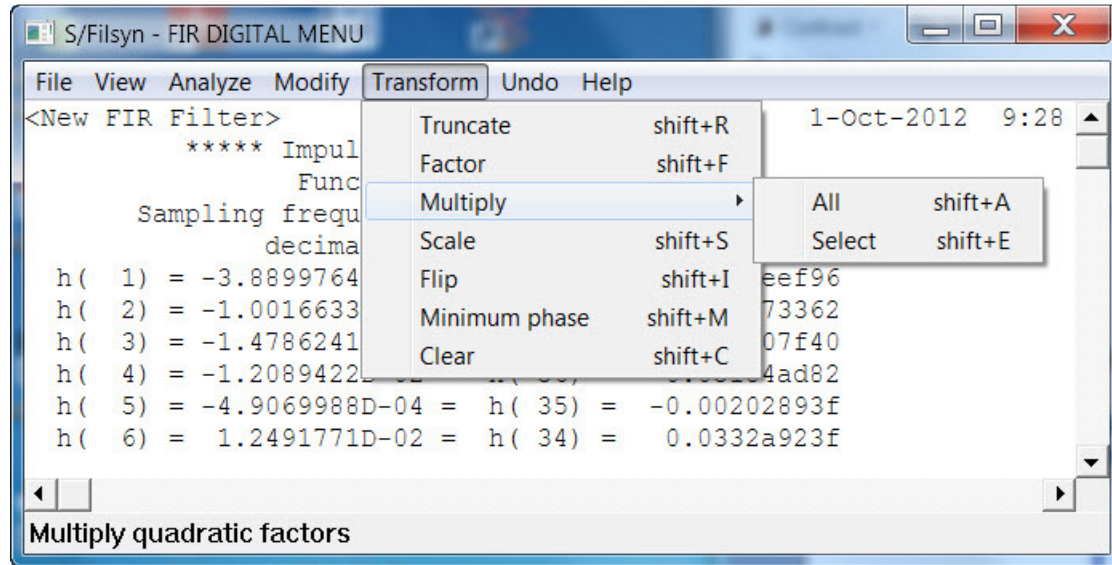
10.5 The 'Modify' menu option

Next come the menu items that can change the filter data we have.



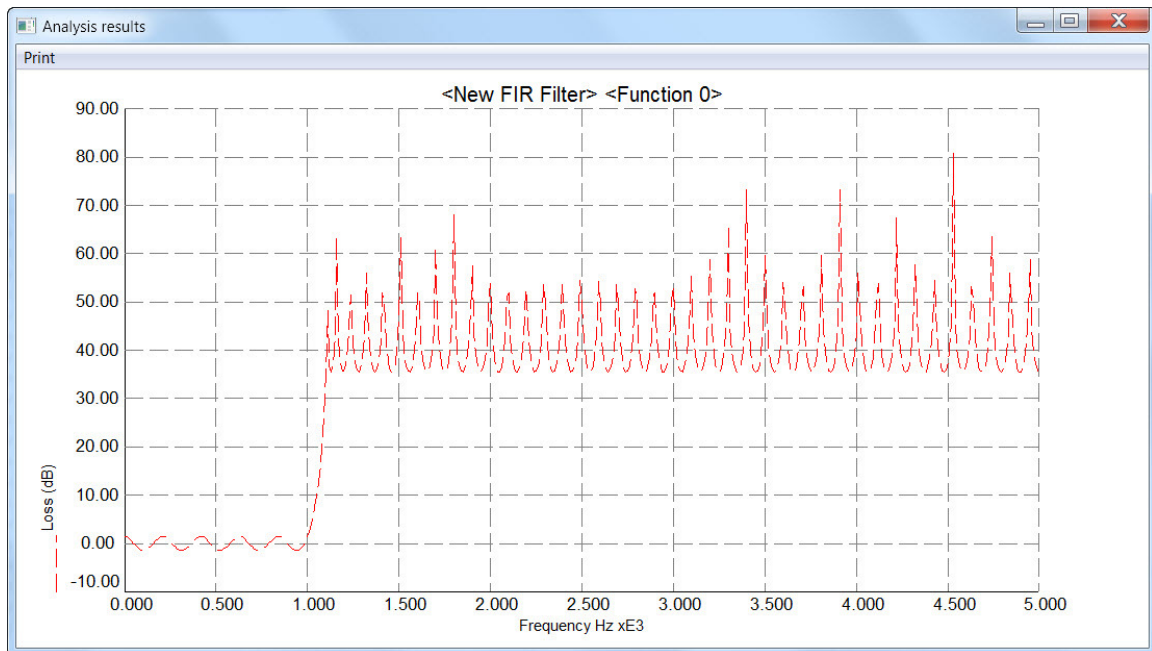
1. **Add**
2. **Delete** and
3. **Change** These three items apply when we have the transfer function in factored form. These three submenu items permit us to add, delete or change a factor.
4. **Flat gain** permits us to add a (positive or negative) flat gain to the filter.

10.6 The 'Transform' menu option

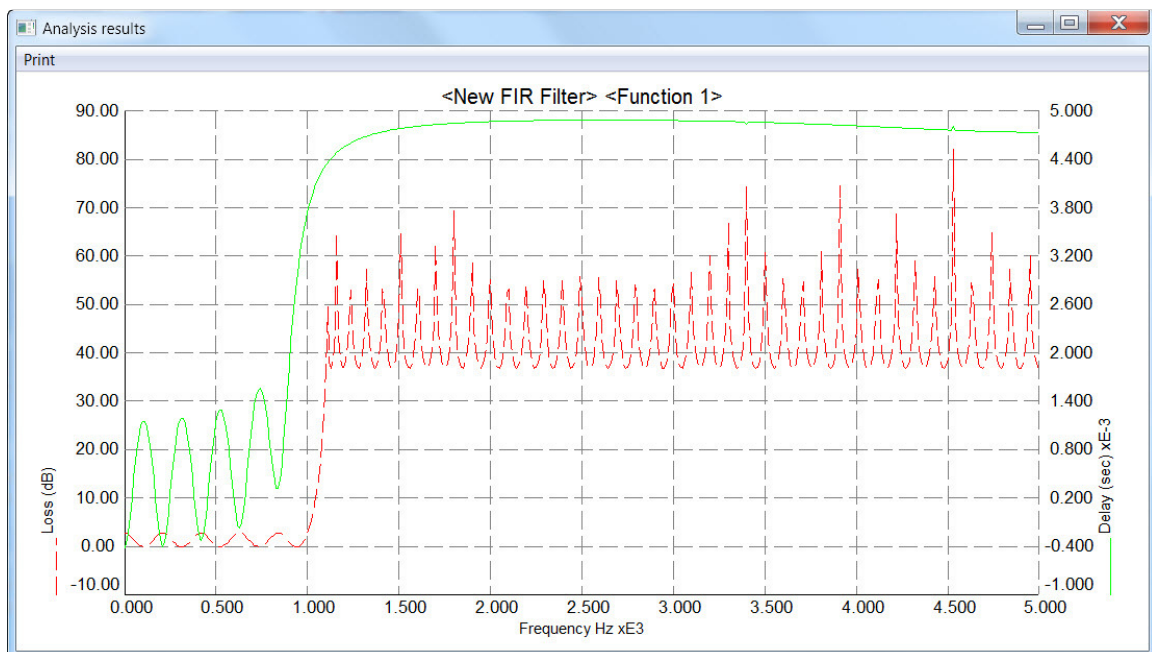


1. **Truncate**, lets us truncate the coefficients of the filter to a specific number of bits (binary digits).
2. **Factor** computes the factored form of the function, while
3. **Multiply** multiplies some or all of the factors back together again.
4. **Scale** performs a scaling operation on the transfer function to make the minimum loss equal to zero dB.
5. **Flip** is used to take all the zeros in the factored form and move them to their inverse positions if they were outside of the unit circle. This leaves the magnitude of the function unchanged, but the phase changes from linear phase to minimum phase.
6. **Minimum phase** is a combination of steps: first the function is factored using **Factor**, then the roots outside the unit circle are flipped inside using **Flip**, next all the factors are multiplied together (**Multiply**->**All**) and finally the new function is rescaled for zero minimum loss (**Scale**). This converts the original linear phase function to a minimum phase one.
7. **Clear** deletes all data from memory.

As an example of the minimum-phase option, consider a 101 tap lowpass with passband from 0 to 1 KHz, stopband from 1.1 KHz to the 5 KHz *Nyquist* rate. The analysis shows:

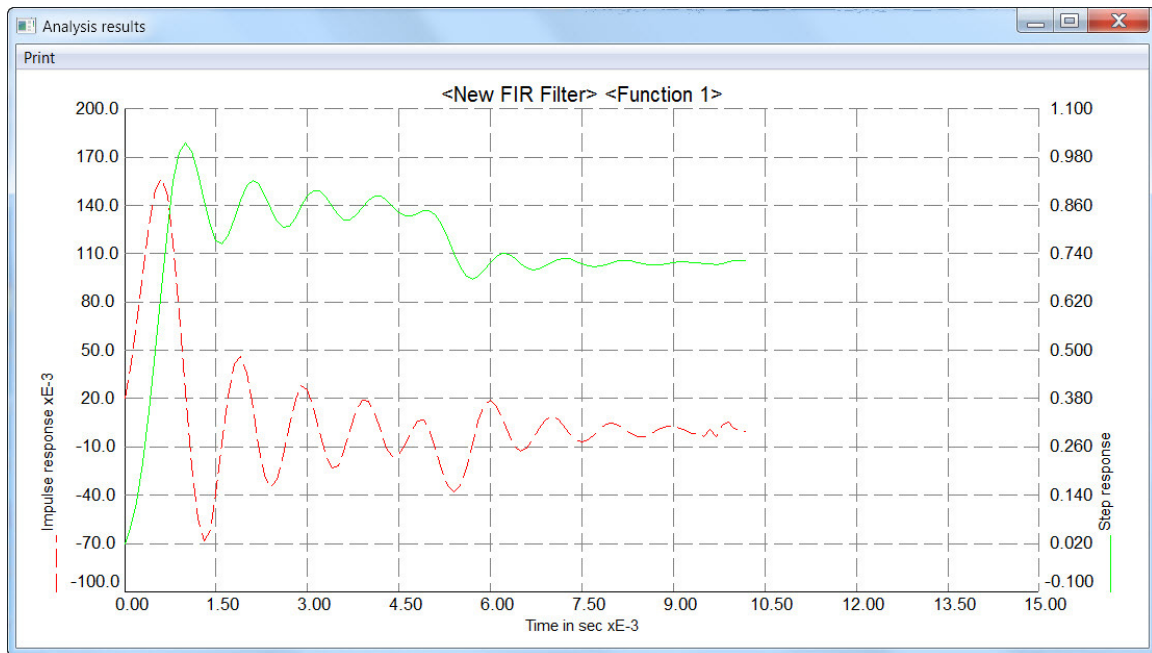


Applying the **Minimum phase** menu option and reanalyzing the modified function, we get, including the delay:

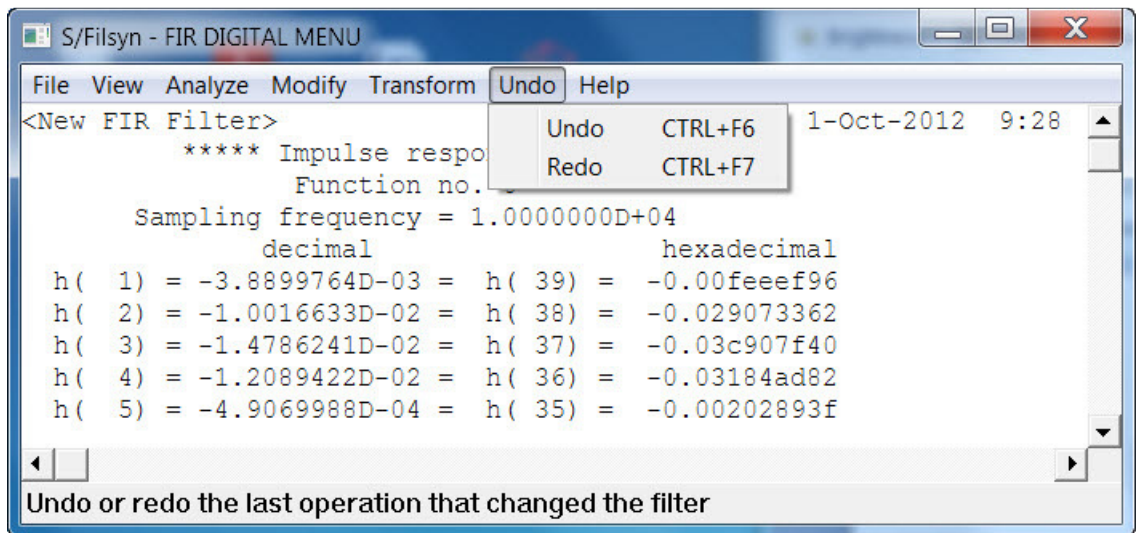


Please note that this is now Function 1 and that must be specified when the analysis is requested.

Of course, the time domain response is far from that of the linear phase case:



10.7 The 'Undo' menu option



Undo can undo the last ten operations that changed the current function in reverse sequence, while **Redo** can redo the operations we have undone, except the very first one.

10.8 The 'Help' menu option

Finally, **Help** contains the usual items.

11. EXISTING FILTERS

Existing filters may be entered into **Filsyn** for further modification and analysis in two ways depending on where the filter data originated. If it was designed in **Filsyn** and saved, the data can be recalled simply from the main menu using the **File->Recall** menu option. The three letter extension of the saved file will tell the program where it was saved and therefore where it should be read back.

If the filter was obtained from another source, then we need the **Analysis** menu option of the starting main menu. If the filter is an FIR digital filter, then we use the **FIR filter** submenu, otherwise we use the **Filter** submenu. In both cases, we have the further submenu options of **New** or **Recall**, just as in the case of entering a set of specifications and for the same purpose.

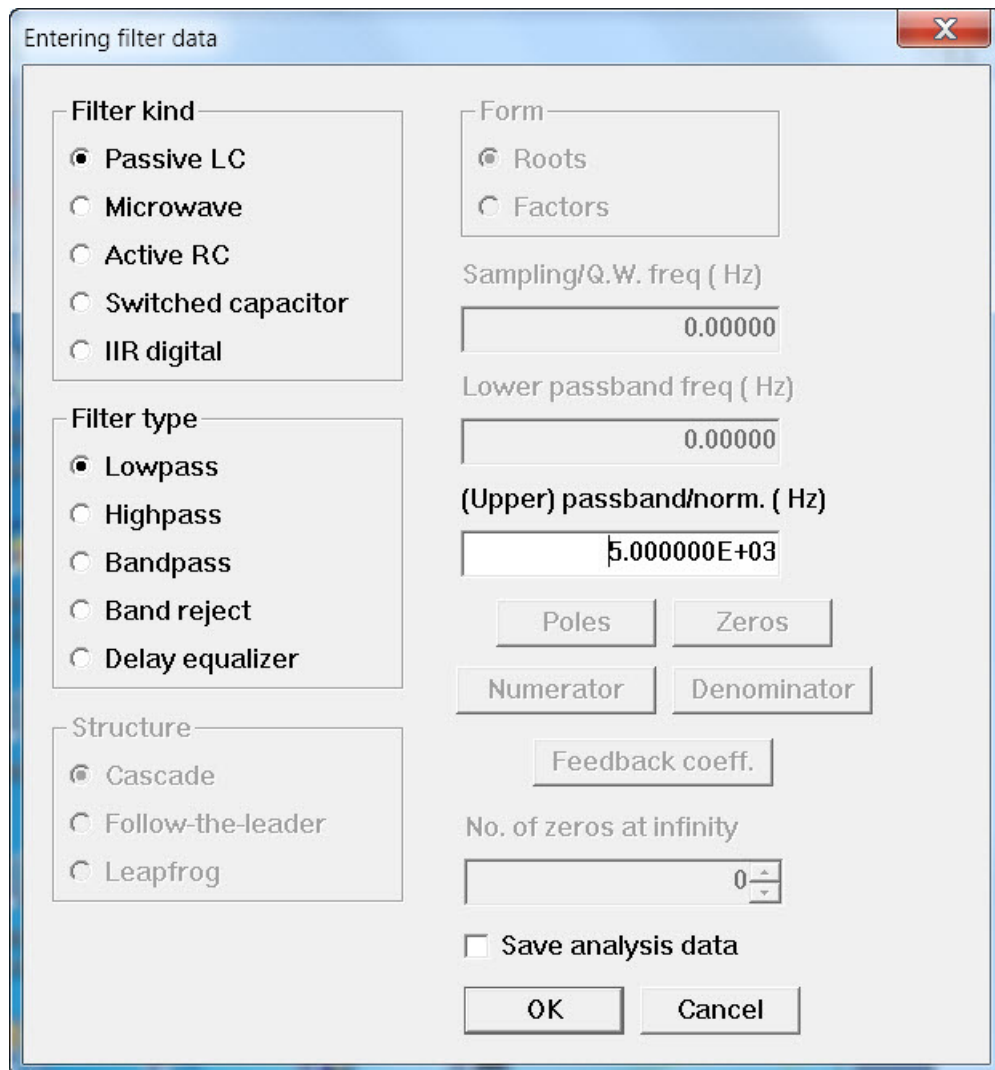
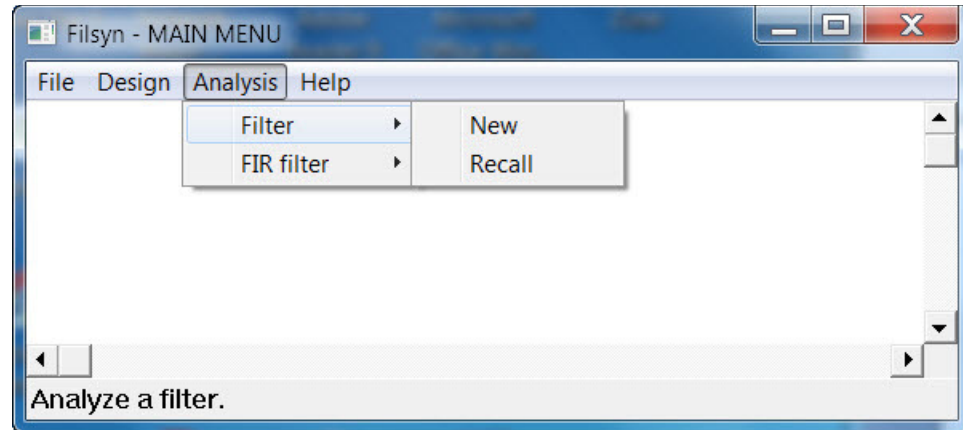
Note that passive LC/microwave filters are entered by their structure and element values (with the exception of delay lines/equalizers) since the program can then offer a large number of transformation and other modification options. All other filters (active RC, IIR digital, FIR digital and delay lines/equalizers) are entered by their transfer functions. Specifically, active RC and IIR digital filters are entered by their poles and zeros, FIR filters are entered by their coefficients or zeros and delay lines/equalizers are specified by their poles (natural modes). This way the program's implementation methods and other options are all available. If we were to enter an active RC filter, for instance, by its structure and element values, there would be no operation available. In any case, then the only operation of interest would be a frequency or time domain analysis, for which other programs are available.

One question may arise. If we specify active RC or IIR digital filters by their transfer function (poles and zeros) here, is that any different from using the functional input in the **Design** segment of the program? The answer to that question is yes. The reason is in the way the program handles complex transmission zeros. In the **Design** segment, complex zeros are treated as quadruplets of transmission zeros, but here in the **Analysis** segment they are just treated as a complex pair. This may occur for instance, if the transfer function contains one or more delay equalizer sections, where the zeros indeed represent just complex pairs, rather than quadruplets. However, if the transfer function contains only pure imaginary zeros, the two segments do indeed yield identical results.

Another difference is the fact that here in the **Analysis** segment one must specify both the numerator and denominator functions, while in the **Design** segment one can specify just the poles and request the zeros to be computed in order to obtain an equal-minima type stopband (at least for lowpass filters).

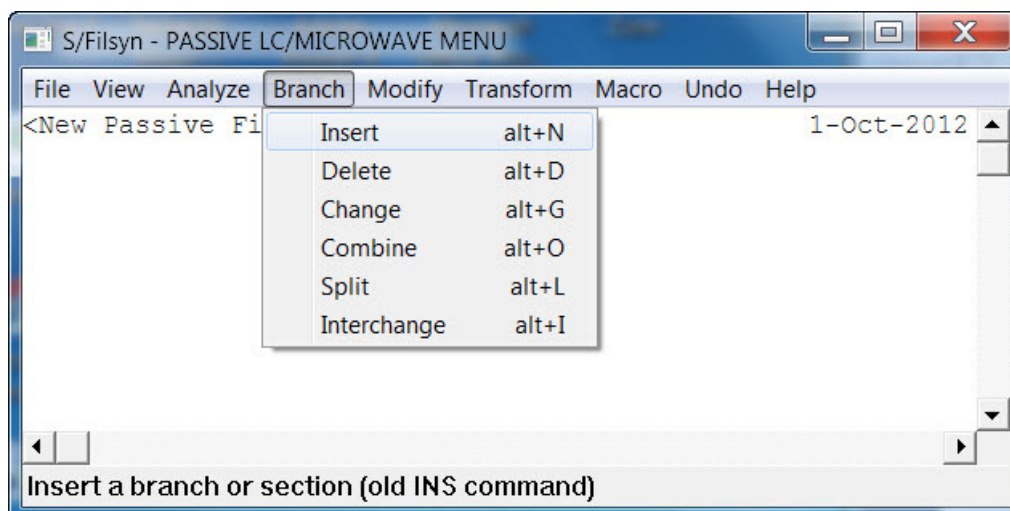
11.1 Passive LC filters

Consider a passive, so-called *image parameter* lowpass filter. First we select the entry point, followed by the necessary data entry. The information we enter is very little:

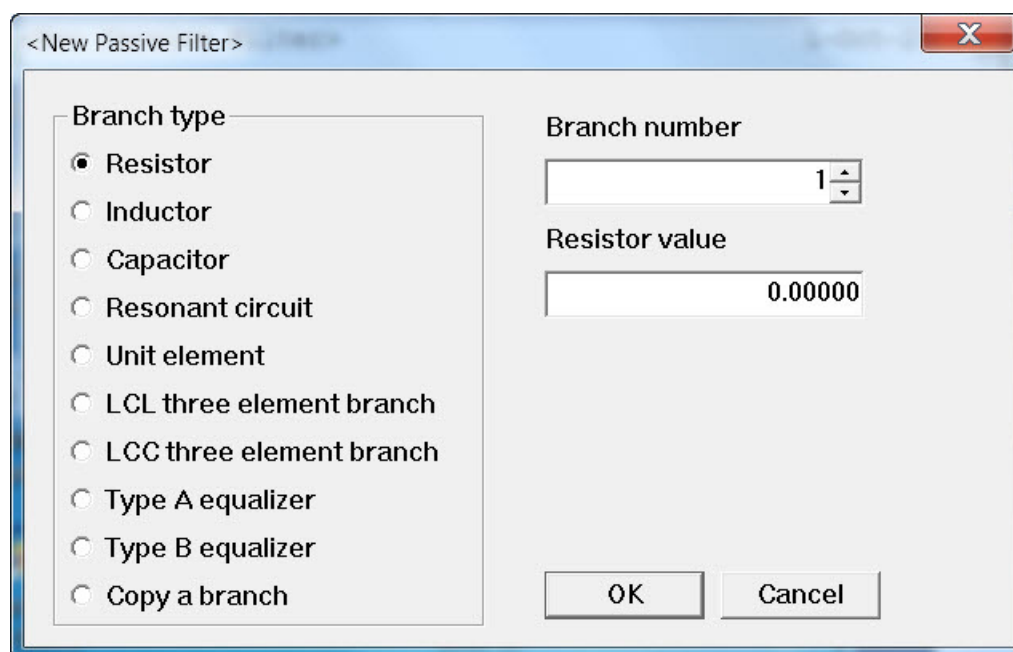


Most of the filter data will be entered after we click on the **OK** button. The frequency entered above is mainly for normalization and the exact value is not significant.

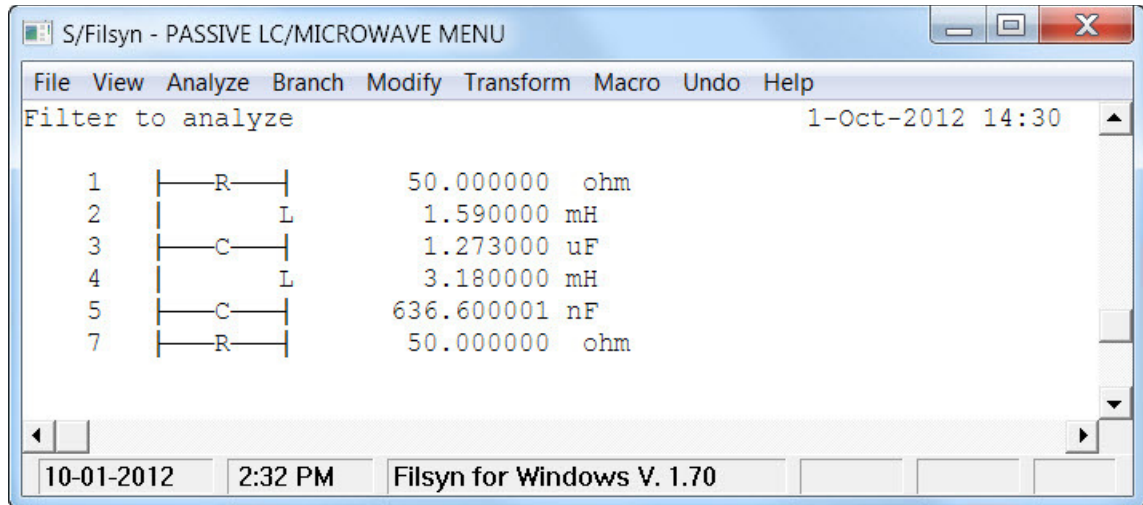
After this screen clears, we are in the analysis segment and only the title and the date is shown. To enter the actual circuit, we need the **Branch->Insert** menu option:



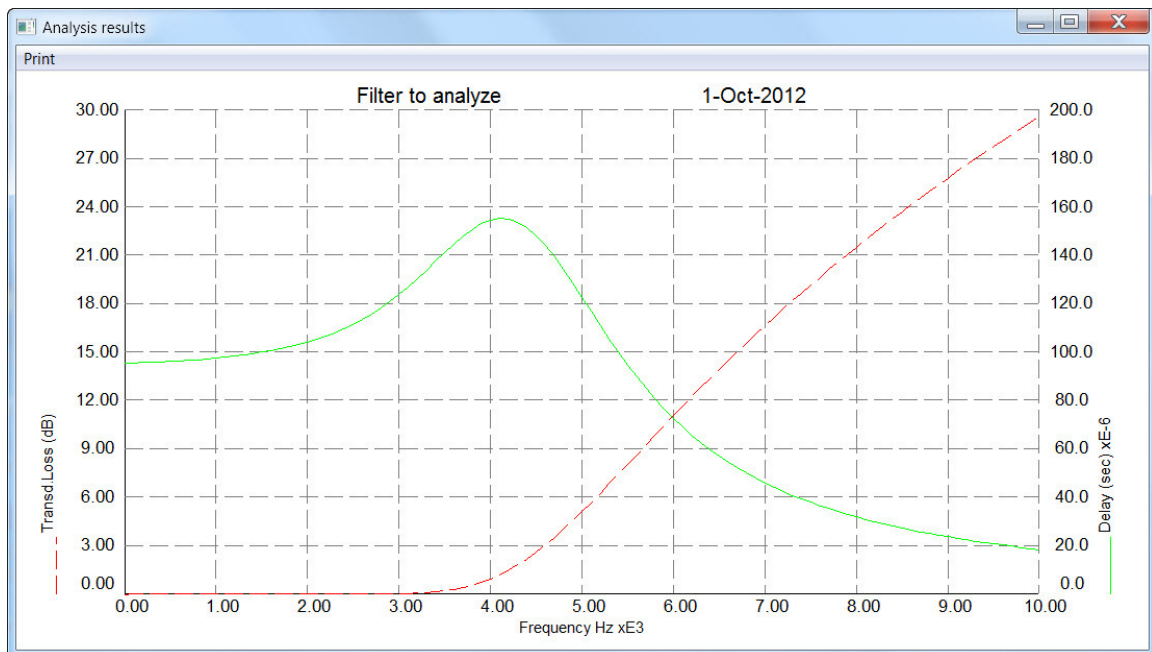
that brings up the following data entry screen:



Here we may enter any type of branch, except bridged-T's, twin-T's or lattices. The only thing we must remember is that entering a two-terminal branch if the branch number is even, the branch is a series one, otherwise it is in shunt. We must start with a termination, followed in this example by four reactive branches and finally the output termination, yielding the circuit. After entering all the branches we used the **View-> Change title** menu item:



At this stage we can do whatever we need to, here we only show the results of a frequency domain analysis:



Entering a microwave filter is entirely analogous and needs no further demonstration.

11.2 Active RC filters

The situation is somewhat different when we enter an existing active RC or IIR digital filter. In this case we enter the complete set of filter data in that first entry window above; furthermore, we have the option of entering the numerator and denominator polynomials in either factored form or by their zeros. In some cases, in particular the active RC filters which have one of the feedback forms, only the factored form is possible. As an example, consider the case of a bandpass filter in a leapfrog feedback design form that we have

obtained from **Filsyn** and which we now would like to reenter into it. The data input screen is as shown, where we already entered most of the data:

Entering filter data

Filter kind

- ☐ Passive LC
- ☐ Microwave
- ☒ Active RC
- ☐ Switched capacitor
- ☐ IIR digital

Filter type

- ☐ Lowpass
- ☐ Highpass
- ☒ Bandpass
- ☐ Band reject
- ☐ Delay equalizer

Structure

- ☐ Cascade
- ☐ Follow-the-leader
- ☒ Leapfrog

Form

- ☐ Roots
- ☒ Factors

Sampling/Q.W. freq (Hz)

0.00000

Lower passband freq (Hz)

1.000000E+03

(Upper) passband/norm. (Hz)

2K

Poles Zeros

Numerator Denominator

Feedback coeff.

No. of zeros at infinity

0

☐ Save analysis data

OK Cancel

Since this is a feedback structure, only the factored form can be used and this needs the numerator and denominator factors as well as the feedback coefficients, in sequence:

Specified numerator factors

	Quadratic	Linear	Constant
1	0.00000	0.00000	35.252000E+06
2	794.390000E-03	0.00000	20.071000E+06
3	336.650000E-03	0.00000	67.282000E+06
4	1.575700E+00	0.00000	0.00000
5	0.00000	0.00000	0.00000
6	0.00000	0.00000	0.00000
7	0.00000	0.00000	0.00000
8	0.00000	0.00000	0.00000
9	0.00000	0.00000	0.00000
10	0.00000	0.00000	0.00000
11	0.00000	0.00000	0.00000

OK Read from file Cancel

Specified denominator factors

	Quadratic	Linear	Constant
1	1.000000E+00	3.685400E+03	68.374000E+06
2	1.000000E+00	0.00000	73.351000E+06
3	1.000000E+00	0.00000	98.637000E+06
4	1.000000E+00	4.583500E+03	9.7786E7
5	0.00000	0.00000	0.00000
6	0.00000	0.00000	0.00000
7	0.00000	0.00000	0.00000
8	0.00000	0.00000	0.00000
9	0.00000	0.00000	0.00000
10	0.00000	0.00000	0.00000
11	0.00000	0.00000	0.00000

OK Read from file Cancel

and finally:

The main dialog box is titled "Entering filter data". It contains three sections:

- Filter kind:**
 - ☐ Passive LC
 - ☐ Microwave
 - ☒ Active RC
 - ☐ Switched capacitor
 - ☐ IIR digital
- Filter type:**
 - ☐ Lowpass
 - ☐ Highpass
 - ☒ Bandpass
 - ☐ Band reject
 - ☐ Delay equalizer
- Structure:**
 - ☐ Cascade
 - ☐ Follow-the-leader
 - ☒ Leapfrog

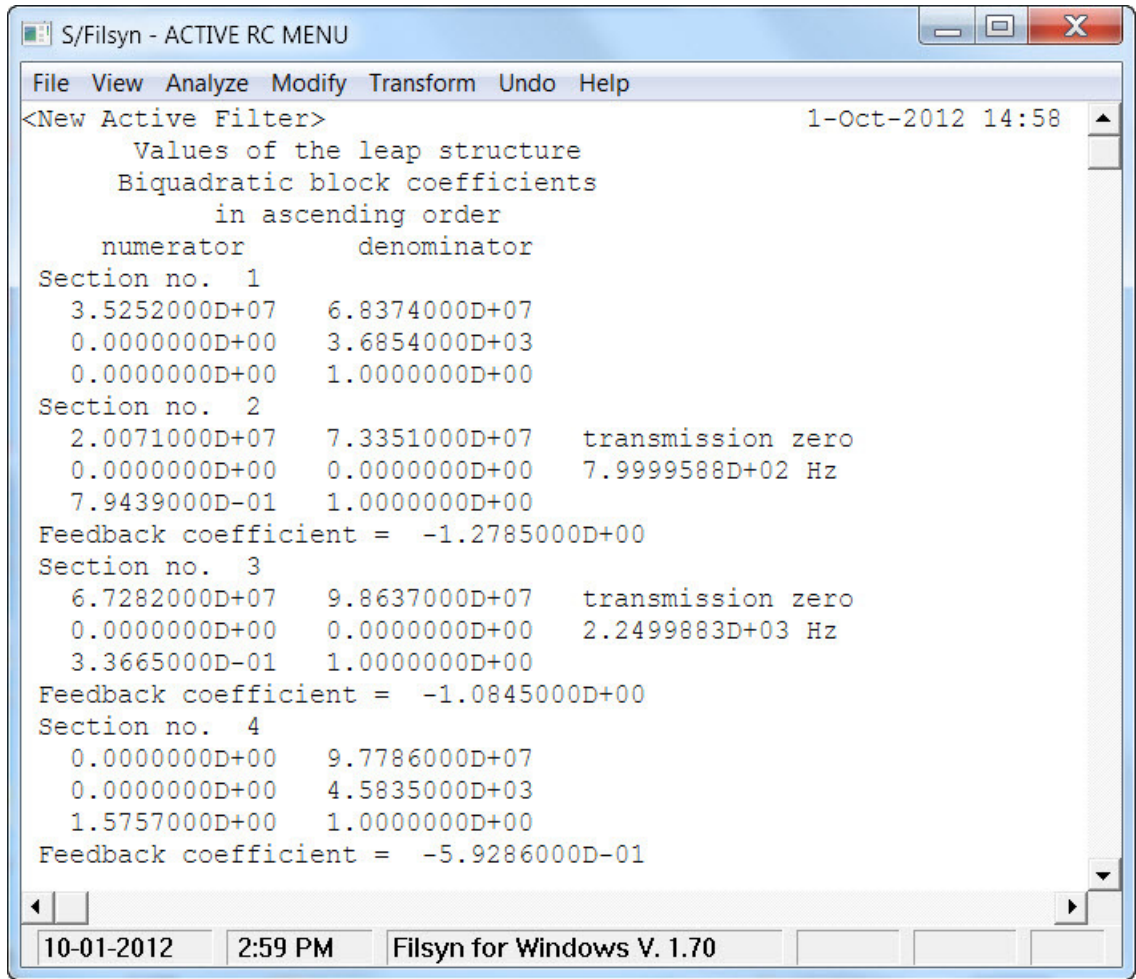
A sub-dialog box titled "Feedback structure" is open, showing a table of coefficients:

	Coefficients
1	-1.27850000
2	-1.08450000
3	-0.59286
4	0.00000000
5	0.00000000
6	0.00000000
7	0.00000000
8	0.00000000
9	0.00000000
10	0.00000000
11	0.00000000
12	0.00000000

Buttons at the bottom of the sub-dialog: OK, Cancel, and Read from file.

Please note that instead of entering all this data from the keyboard, we could have read it in from a file. We shall have more to say about this later.

Clicking the **OK** buttons, we finally get the filter data, which is exactly the same as the original design information, save for differences caused by truncating the data entry:



From this point on, the complete list of options are available for implementing, analyzing and modifying this filter.

11.3 IIR digital filters

If we were entering an active RC filter in cascaded second order sections form, transfer function can be entered either in factored form as above, or by its (complex) poles and zeros. The same is true for IIR digital filters, where, of course, no feedback form exists. As an example, consider a lowpass design that has a maximally flat delay and was obtained using the procedure described by *Thiran* (see ref. [55]). The transfer function numerator has all the zeros at $z = -1.0$, while the poles are computed by *Thiran's* procedure and entered as follows:

Entering filter data

Filter kind

- ☐ Passive LC
- ☐ Microwave
- ☐ Active RC
- ☐ Switched capacitor
- ☒ IIR digital

Filter type

- ☐ Lowpass
- ☐ Highpass
- ☐ Bandpass
- ☐ Band reject
- ☒ Delay equalizer

Structure

- ☒ Cascade
- ☐ Follow-the-leader
- ☐ Leapfrog

Form

- ☒ Roots
- ☐ Factors

Sampling/Q.W. freq (Hz)

10.000000E+03

Lower passband freq (Hz)

0.00000

(Upper) passband/norm. (Hz)

2.000000E+03

Poles Zeros

Numerator Denominator

Feedback coeff.

No. of zeros at infinity

0

☐ Save analysis data

OK Cancel

and the zeros and poles:

Specified zeros in Hz

	Real parts	Imaginary parts
1	-1.000000E+00	0.000000
2	-1.000000E+00	0.000000
3	-1.000000E+00	0.000000
4	-1.000000E+00	0.000000
5	-1.000000E+00	0.000000
6	-1	0.000000
7	0.000000	0.000000
8	0.000000	0.000000
9	0.000000	0.000000
10	0.000000	0.000000
11	0.000000	0.000000

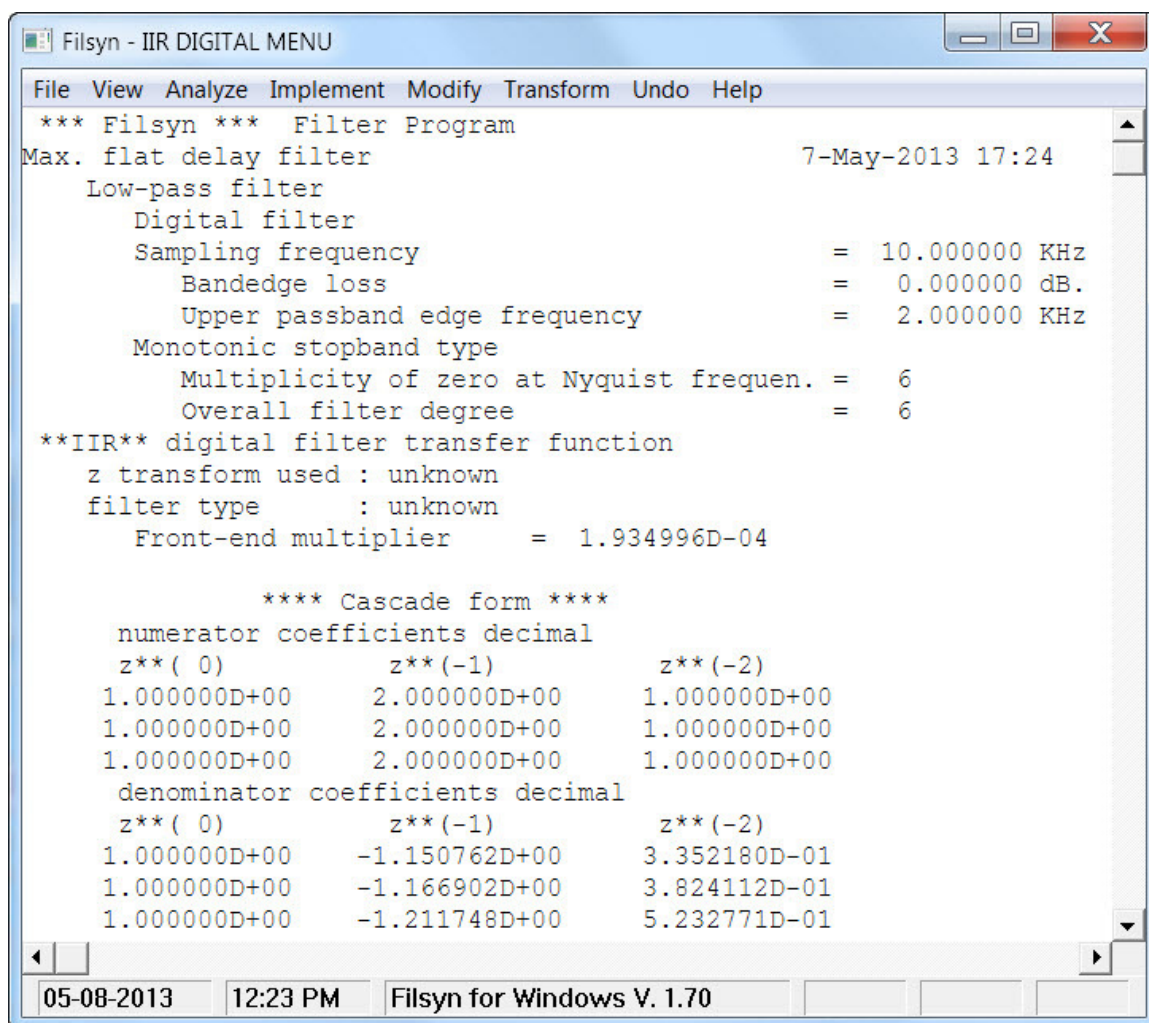
OK Read from file Cancel

Specified poles (normalized)

	Real parts	Imaginary parts
1	575.381100E-03	644.561000E-03
2	583.450000E-03	204.930000E-03
3	605.874000E-03	0.3952136
4	0.000000	0.000000
5	0.000000	0.000000
6	0.000000	0.000000
7	0.000000	0.000000
8	0.000000	0.000000
9	0.000000	0.000000
10	0.000000	0.000000
11	0.000000	0.000000

OK Read from file Cancel

The resulting filter is displayed as:



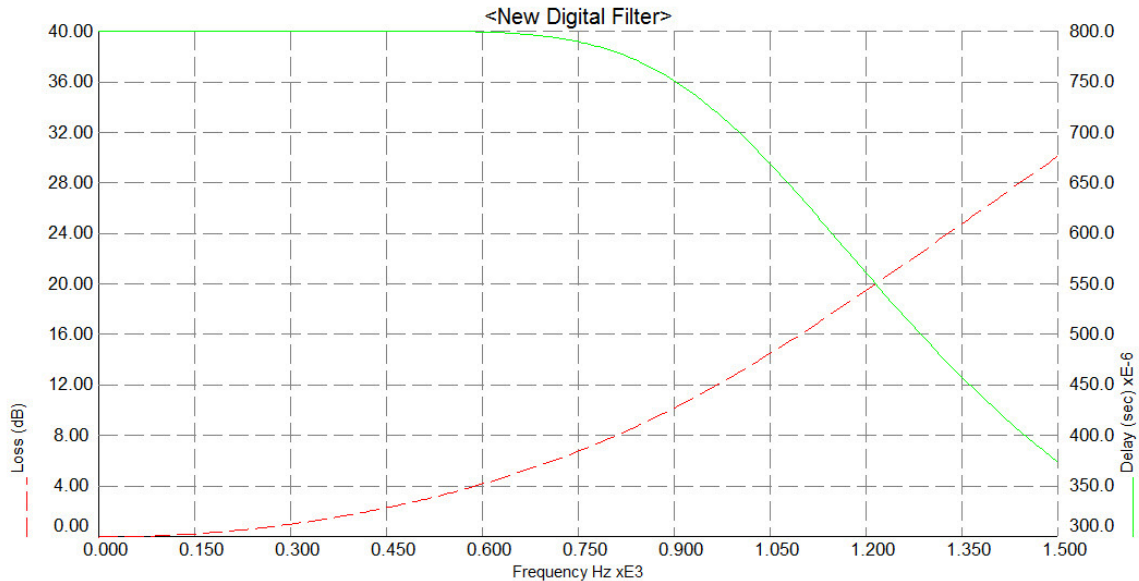
```

Filsyn - IIR DIGITAL MENU
File View Analyze Implement Modify Transform Undo Help
*** Filsyn *** Filter Program
Max. flat delay filter                                     7-May-2013 17:24
  Low-pass filter
    Digital filter
      Sampling frequency                                   = 10.000000 KHz
      Bandedge loss                                       = 0.000000 dB.
      Upper passband edge frequency                       = 2.000000 KHz
      Monotonic stopband type
        Multiplicity of zero at Nyquist frequen. = 6
        Overall filter degree                       = 6
**IIR** digital filter transfer function
  z transform used : unknown
  filter type      : unknown
  Front-end multiplier = 1.934996D-04

      **** Cascade form ****
      numerator coefficients decimal
      z**( 0)      z**(-1)      z**(-2)
      1.000000D+00  2.000000D+00  1.000000D+00
      1.000000D+00  2.000000D+00  1.000000D+00
      1.000000D+00  2.000000D+00  1.000000D+00
      denominator coefficients decimal
      z**( 0)      z**(-1)      z**(-2)
      1.000000D+00 -1.150762D+00  3.352180D-01
      1.000000D+00 -1.166902D+00  3.824112D-01
      1.000000D+00 -1.211748D+00  5.232771D-01
05-08-2013 12:23 PM Filsyn for Windows V. 1.70

```

Here we have already added a flat loss to bring the loss up to zero at zero frequency and the subsequent analysis shows the expected behavior of this structure:



Note again, that instead of entering the pole-zero data from the keyboard, it could have been read from a data file.

11.4 Delay equalizers

In order to demonstrate the generation of delay-equalizers/delay-lines in this segment of the program, we have created several delay equalizers in the **Design->Delayline/equ.** segment of the program. These all equalized a linearly decreasing delay over the 0 – 2 KHz band (see chapter 12) using a 10th order (five section) equalizer. Then we generated a file containing the poles/zeros of the delay equalizer using the **File->Pole-zero** menu option. This generated the following file for a passive LC implementation, called *test_lc.pz*:

```
! Pole-zero data of:
! Linear delay case
! Poles (in rad/sec):
-3.397924751E+03    2.736969297E+03
-3.171652627E+03    5.987241762E+03
-2.844046468E+03    8.699697397E+03
-2.402989953E+03    1.103038498E+04
-1.657196166E+03    1.303162893E+04
! Zeros (in Hz):
2.637509615E+02    2.074048161E+03
3.824477292E+02    1.755540294E+03
4.526440537E+02    1.384599844E+03
5.047841934E+02    9.528991220E+02
5.407965203E+02    4.356021928E+02
```

For our purposes we only need the poles and the procedure is as follows:

We call the **Analysis->New** program segment, select the **Passive LC** and **Delay equalizer** options, specify a 2 KHz normalization frequency. Next we click on the **Poles** button, select **Read from file** button and specify the *test_lc.pz* file:

Entering filter data

Filter kind

- ☒ Passive LC
- ☐ Microwave
- ☐ Active RC
- ☐ Switched capacitor
- ☐ IIR digital

Form

- ☒ Roots
- ☐ Factors

Filter type

- ☐ Lowpass
- ☐ Highpass
- ☐ Bandpass
- ☐ Band reject
- ☒ Delay equalizer

Structure

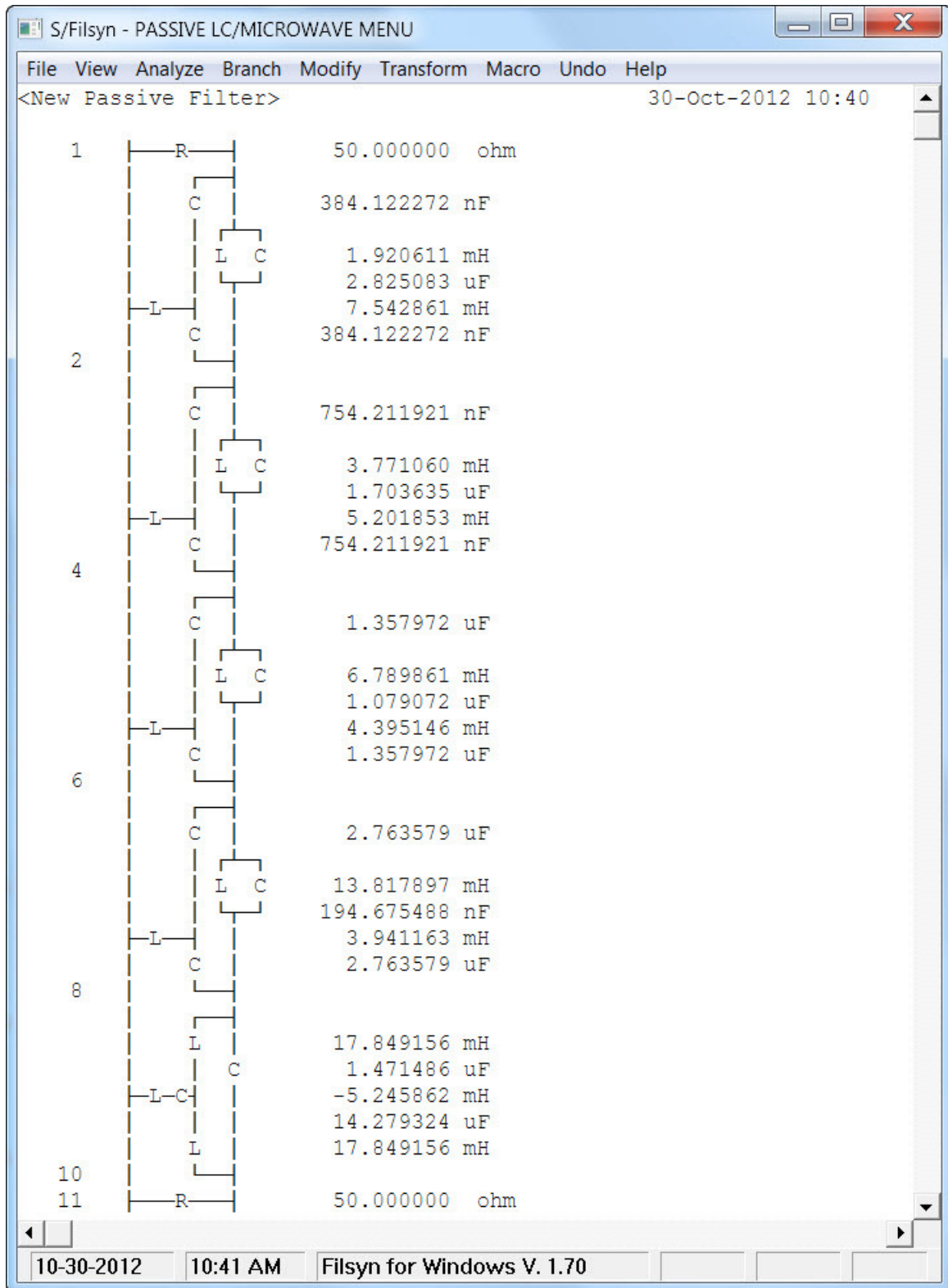
- ☒ Cascade
- ☐ Follow-the-leader
- ☐ Leapfrog

Specified poles (in rad/sec)

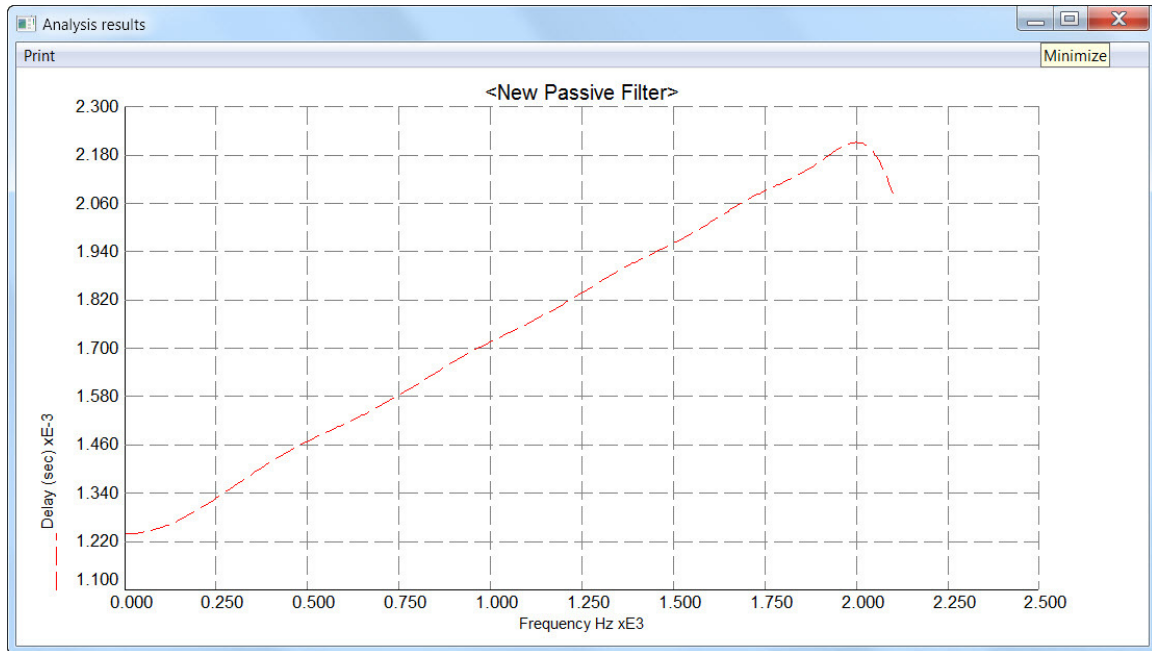
	Real parts	Imaginary parts
1	-3.397925E+03	2.736969E+03
2	-3.171653E+03	5.987242E+03
3	-2.844046E+03	8.699697E+03
4	-2.402990E+03	11.030385E+03
5	-1.657196E+03	13.031629E+03
6	0.00000	0.00000
7	0.00000	0.00000
8	0.00000	0.00000
9	0.00000	0.00000
10	0.00000	0.00000
11	0.00000	0.00000

OK Read from file Cancel

Clicking on the **OK** buttons, we immediately have the passive LC implementation of this delay equalizer:



To check the correctness of this circuit, we do a frequency domain analysis and get the following result:



which is correct.

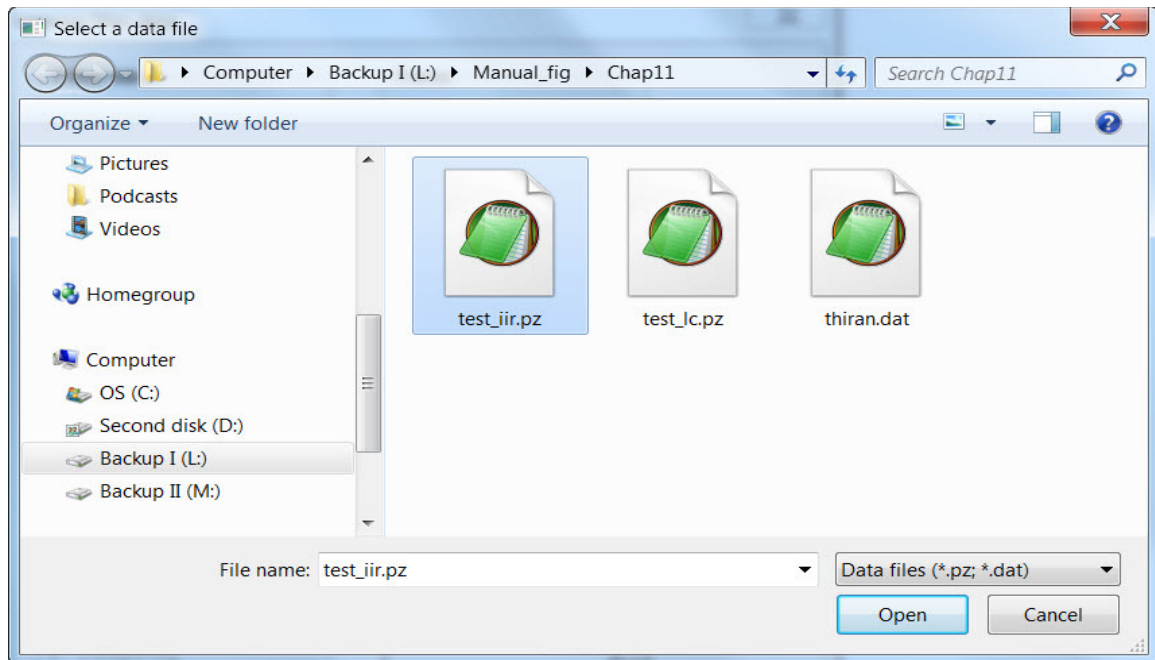
Just to show the IIR digital version, the original equalizer design generated the following pole-zero file, called *test_iir.pz*:

```

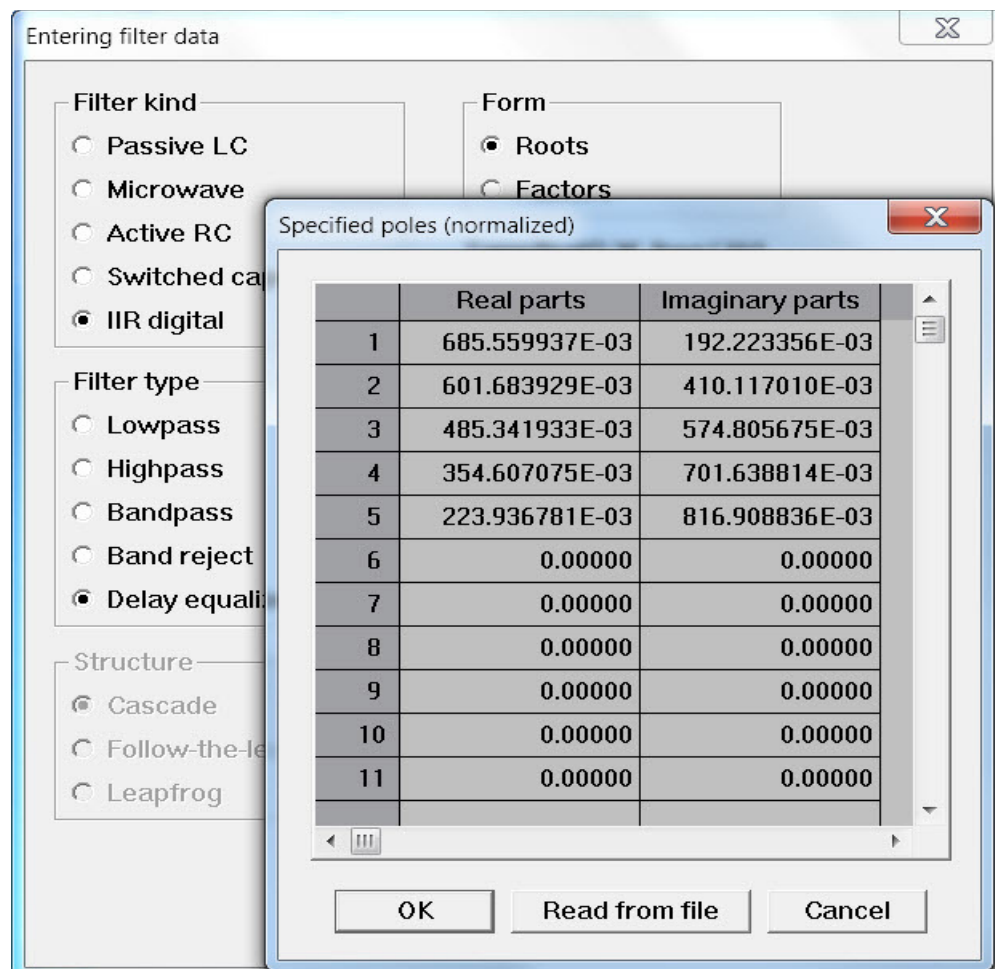
! Pole-zero data of:
! near delay case
! Poles (normalized):
6.855599365E-01    1.922233559E-01
6.016839288E-01    4.101170105E-01
4.853419331E-01    5.748056753E-01
3.546070753E-01    7.016388142E-01
2.239367808E-01    8.169088358E-01
! Zeros (in Hz):
5.406160758E+02    4.350806563E+02
5.048905552E+02    9.521928730E+02
4.529826440E+02    1.383990221E+03
3.829243386E+02    1.755223414E+03
2.641967564E+02    2.074173911E+03

```

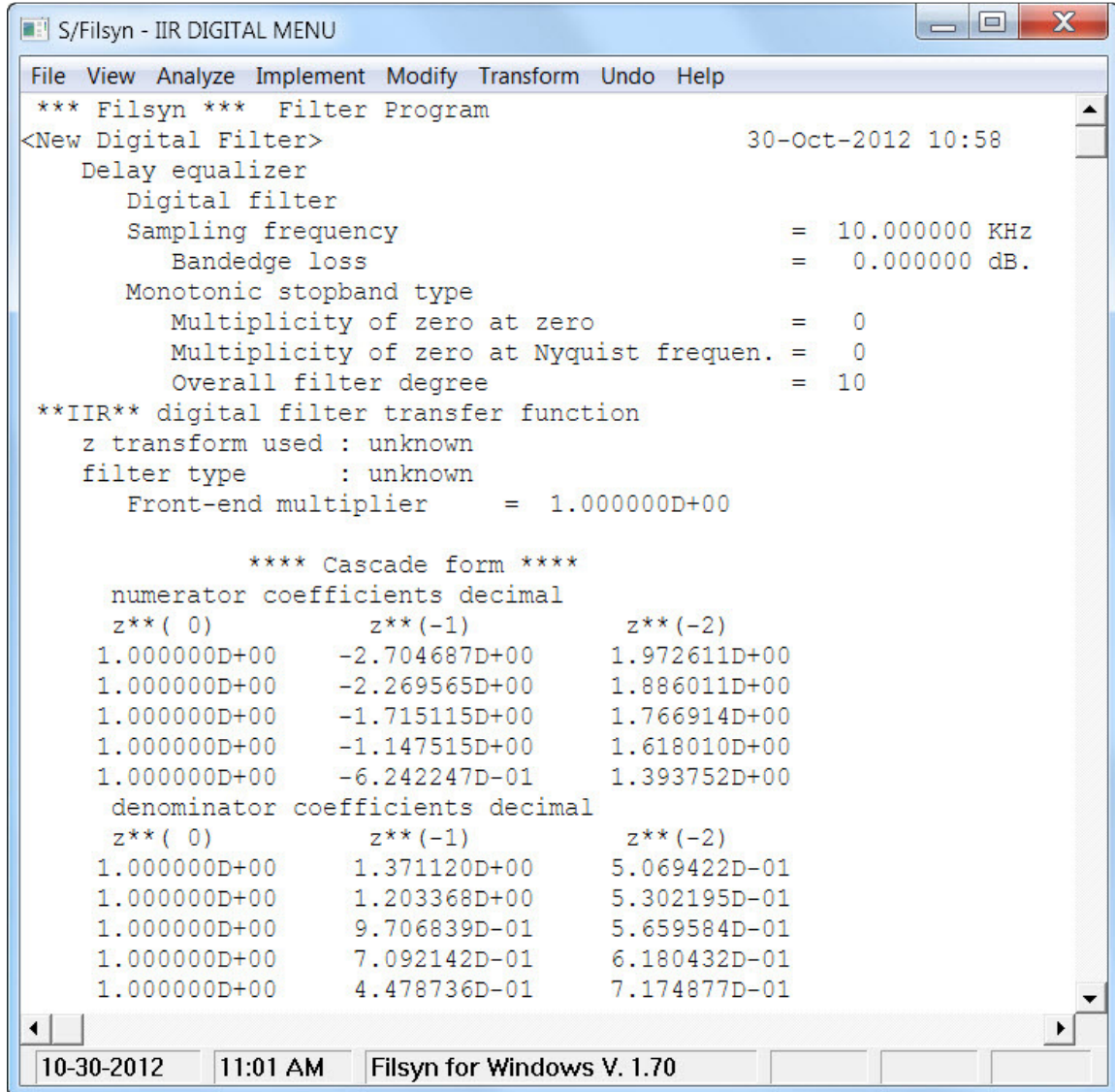
Again we only need the poles of the function. Using a 10 KHz sampling rate and 2 KHz normalization, when we read in this file:



which yields:



we get the final filter as follows, selecting decimal display:



```

S/Filsyn - IIR DIGITAL MENU
File View Analyze Implement Modify Transform Undo Help
*** Filsyn *** Filter Program
<New Digital Filter> 30-Oct-2012 10:58
  Delay equalizer
    Digital filter
      Sampling frequency          = 10.000000 KHz
      Bandedge loss               = 0.000000 dB.
      Monotonic stopband type
        Multiplicity of zero at zero      = 0
        Multiplicity of zero at Nyquist frequen. = 0
        Overall filter degree            = 10
**IIR** digital filter transfer function
  z transform used : unknown
  filter type      : unknown
  Front-end multiplier = 1.000000D+00

      **** Cascade form ****
      numerator coefficients decimal
      z**( 0)      z**(-1)      z**(-2)
      1.000000D+00 -2.704687D+00  1.972611D+00
      1.000000D+00 -2.269565D+00  1.886011D+00
      1.000000D+00 -1.715115D+00  1.766914D+00
      1.000000D+00 -1.147515D+00  1.618010D+00
      1.000000D+00 -6.242247D-01  1.393752D+00
      denominator coefficients decimal
      z**( 0)      z**(-1)      z**(-2)
      1.000000D+00  1.371120D+00  5.069422D-01
      1.000000D+00  1.203368D+00  5.302195D-01
      1.000000D+00  9.706839D-01  5.659584D-01
      1.000000D+00  7.092142D-01  6.180432D-01
      1.000000D+00  4.478736D-01  7.174877D-01
10-30-2012 11:01 AM Filsyn for Windows V. 1.70

```

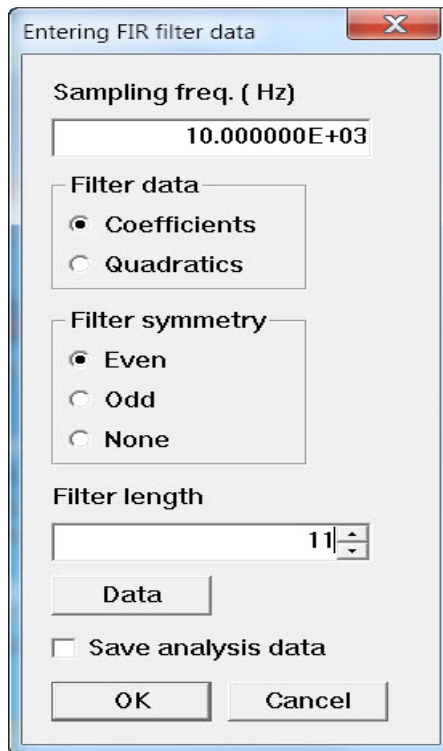
and the analysis of this filter yields results essentially identical to the one above.

11.5 FIR digital filters

Our last example is an FIR digital filter. The data entry screen is very simple here, and we can enter either the coefficients or the function zeros to specify the filter. Since this is a very simple procedure, we use this occasion to demonstrate the use of **Filsyn** to compute the zeros of high order polynomials. Assume that we wish to find the zeros of the 10th order polynomial:

$$z^{10} + 2z^9 + 3z^8 + 4z^7 + 5z^6 + 6z^5 + 5z^4 + 4z^3 + 3z^2 + 2z + 1$$

We may consider this to be an FIR filter, which happens to be symmetrical and having 11 coefficients. Selecting the **Analysis->FIR filters->New** menu option and an arbitrary sampling rate we enter the data as follows:



Entering FIR filter data

Sampling freq. (Hz)

10.000000E+03

Filter data

☒ Coefficients

☐ Quadratics

Filter symmetry

☒ Even

☐ Odd

☐ None

Filter length

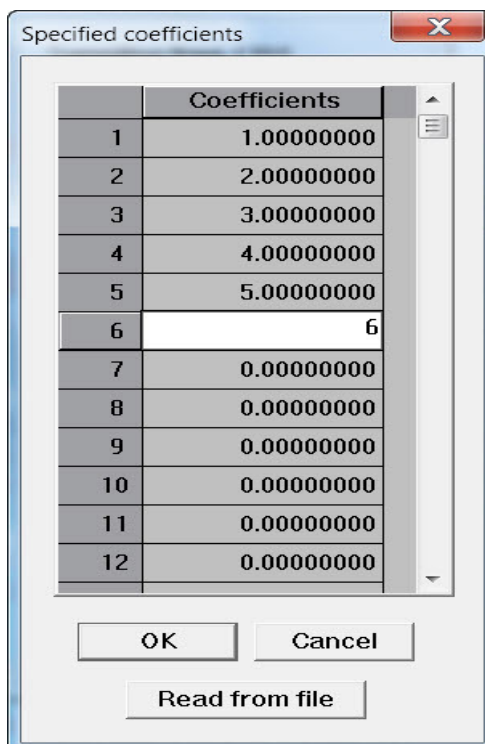
11

Data

☐ Save analysis data

OK Cancel

and in the **Data** submenu:



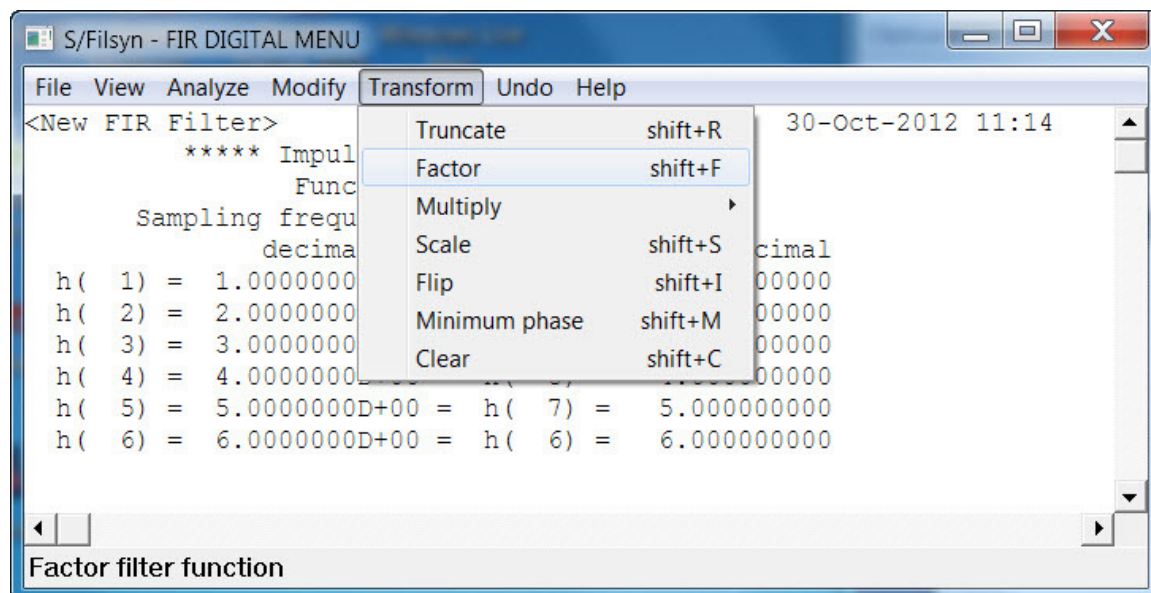
Specified coefficients

	Coefficients
1	1.00000000
2	2.00000000
3	3.00000000
4	4.00000000
5	5.00000000
6	6
7	0.00000000
8	0.00000000
9	0.00000000
10	0.00000000
11	0.00000000
12	0.00000000

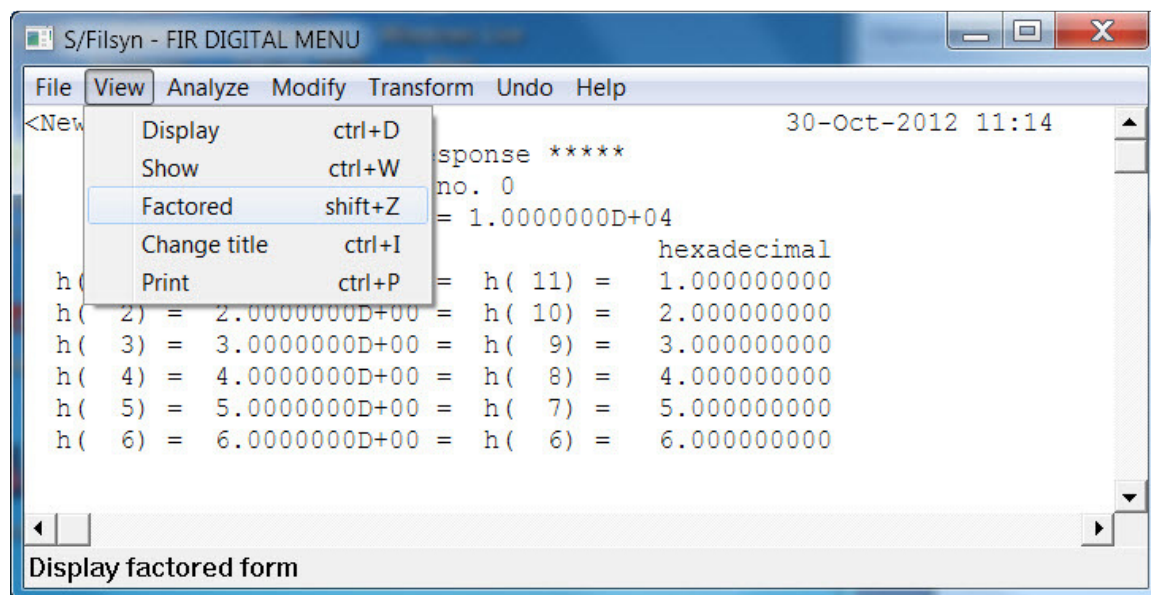
OK Cancel

Read from file

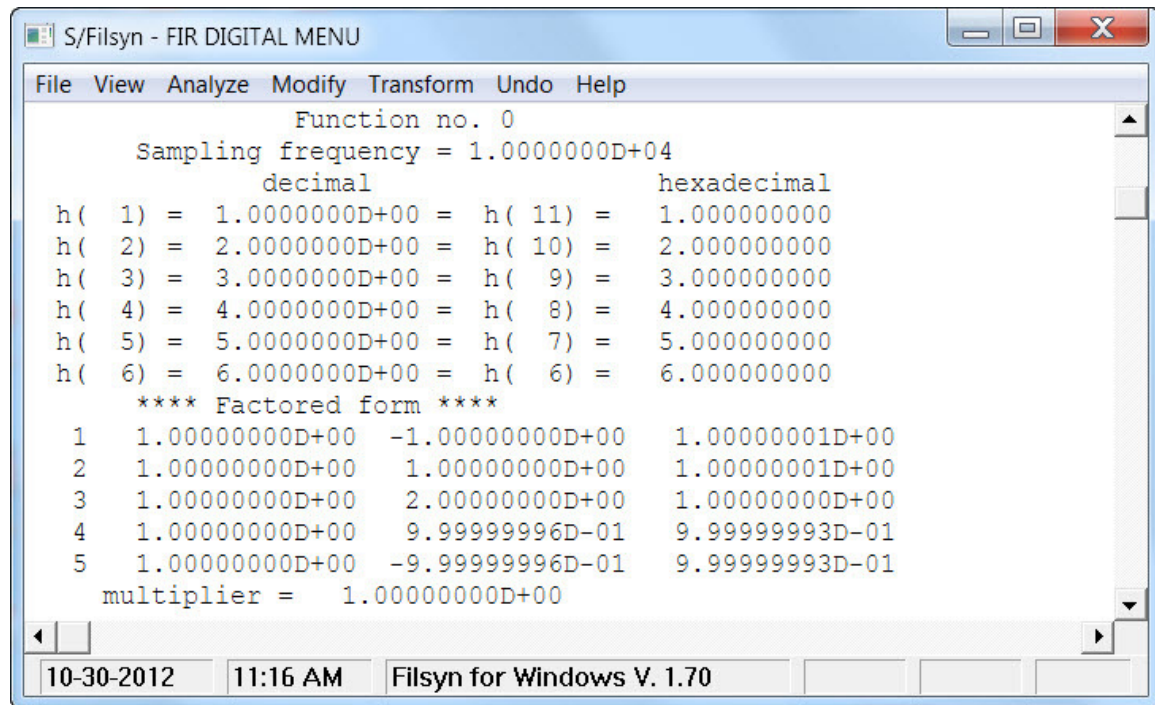
Due to the symmetry, we only enter about half the coefficients. Next we select the **Transform->Factor** menu option:



Next we request the resulting factored form:



which shows that the polynomial can be factored into:



```

S/Filsyn - FIR DIGITAL MENU
File View Analyze Modify Transform Undo Help
Function no. 0
Sampling frequency = 1.0000000D+04
decimal
h( 1) = 1.0000000D+00 = h( 11) = 1.0000000000
h( 2) = 2.0000000D+00 = h( 10) = 2.0000000000
h( 3) = 3.0000000D+00 = h( 9) = 3.0000000000
h( 4) = 4.0000000D+00 = h( 8) = 4.0000000000
h( 5) = 5.0000000D+00 = h( 7) = 5.0000000000
h( 6) = 6.0000000D+00 = h( 6) = 6.0000000000
**** Factored form ****
1 1.00000000D+00 -1.00000000D+00 1.00000001D+00
2 1.00000000D+00 1.00000000D+00 1.00000001D+00
3 1.00000000D+00 2.00000000D+00 1.00000000D+00
4 1.00000000D+00 9.9999996D-01 9.9999993D-01
5 1.00000000D+00 -9.9999996D-01 9.9999993D-01
multiplier = 1.0000000D+00
10-30-2012 11:16 AM Filsyn for Windows V. 1.70

```

$$[(z + 1)(z^2 + z + 1)(z^2 - z + 1)]^2$$

which is correct. The precision of the root extraction procedure is about 8 decimal digits, which is very good considering the existence of several multiple zeros.

11.6 Preparing a file to read

As shown above, any tabulated data can be entered from the keyboard, but can also be read from a data file. These files are simple text files with the following properties:

1. A title and comments may be at the top of the file, each line beginning with an ! (exclamation mark). These lines will be ignored.
2. Blank lines may be used anywhere, also ignored.
3. If poles and zeros or numerator and denominator factors (as well as feedback coefficients) are in the same file, they should be in that sequence and separated by a line that begins again with an !. They should also be read in in that sequence.
4. Numerator and denominator factors are represented by three numbers (quadratic, linear and constant coefficients) in that order on the same line. Zeros and poles are represented by two numbers (real and imaginary parts) again in that sequence on the same line. Complex numbers do *not* need their complex conjugate. Zeros at zero frequency should *not* be included in the file. They will be found by the program.

5. Numbers must be in scientific form (*no* engineering notation) and items on the same line must be separated by blank(s). Do not use tabs, that may cause the program to crash.
6. Files containing numerator-denominator data must have a “.dat” extension, those containing pole-zero data a “.pz” extension.

Apart from the use of scientific notation only, the structure of these data files follows closely the structure you see, when you enter these numbers directly from the keyboard. Also files generated by the **Filsyn** program have just such a structure.

11.7 Switched-capacitor example

Switched-capacitor filters are designed in the active-RC segment of the program, by using the **Transform->Z transform** menu option to convert an analog transfer function into a sampled data form. However, this introduces the familiar frequency warping effect. We could avoid this in the IIR digital design segment, where the data can be prewarped before the transfer function is actually generated. We can utilize this for the design of switched-capacitor filters in the following way:

We design the filter as an IIR digital filter to meet specifications. When the transfer function is determined, we use the **View->Switched-C** menu option to write that data to a text file with a “.dat” extension. Finally we read that data back here in this segment of the program as an **Active RC->Switched C** filter.

Consider the following IIR bandpass design, with passband from 1 KHz to 2 KHz and 10 KHz sampling rate:

Clicking the **OK** button, we get to the synthesis stage and select the default (bilinear) transform:

S/Filsyn - IIR DIGITAL MENU

File View Analyze Implement Modify Transform Undo Help

*** Filsyn *** Filter Program

<New Filter> 30-Oct-2012 11:31

Band-pass filter

Equal ripple pass band

Bandedge loss = 0.500000 dB.

Preshifted Lower passband edge frequency = 894.427192 Hz

Lower passband edge frequency = 1.000000 KHz

Upper passband edge frequency = 2.000000 KHz

Sampling frequency = 10.000000 KHz

Equal minima stop band with edge frequency = 900.000001 Hz

Equal minima stop band with edge frequency = 2.200000 KHz

Preshifted stopband edge frequency = 799.751827 Hz

Preshifted stopband edge frequency = 2.277284 KHz

Required stop band loss = 35.000000 dB.

Multiplicity of zero at zero = 1

Multiplicity of zero at Nyquist frequen. = 1

Number of finite transmission zero pairs = 4

Overall filter degree = 10

Transmission zeros

real part	imaginary part
0.0000000D+00	8.8700883D+02
0.0000000D+00	7.3004284D+02
0.0000000D+00	2.1959022D+03
0.0000000D+00	2.5176970D+03

**** IIR digital analysis segment ****

10-30-2012 11:33 AM Filsyn for Windows V. 1.70

<New Digital Filter>

Z-transform type

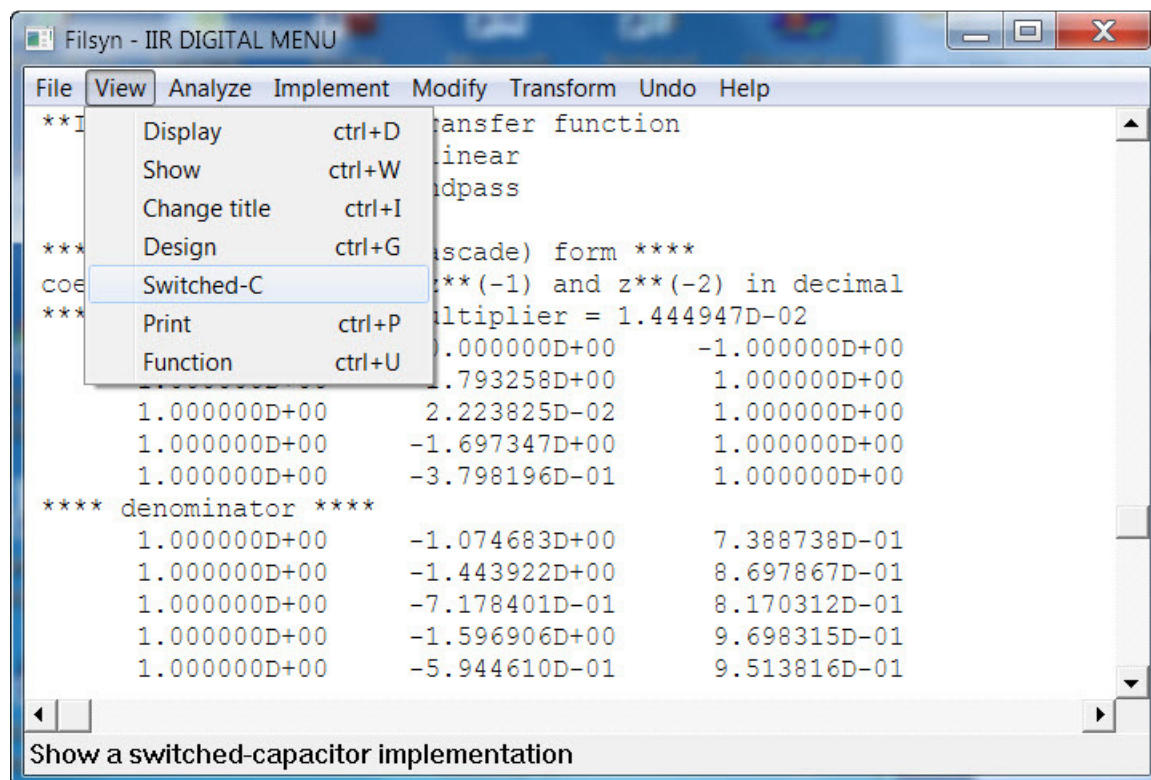
☐ Impulse-invariant

☐ Matched

☒ Bilinear

OK Cancel

The design is shown next, where we selected the **View->Switched-C** menu option:



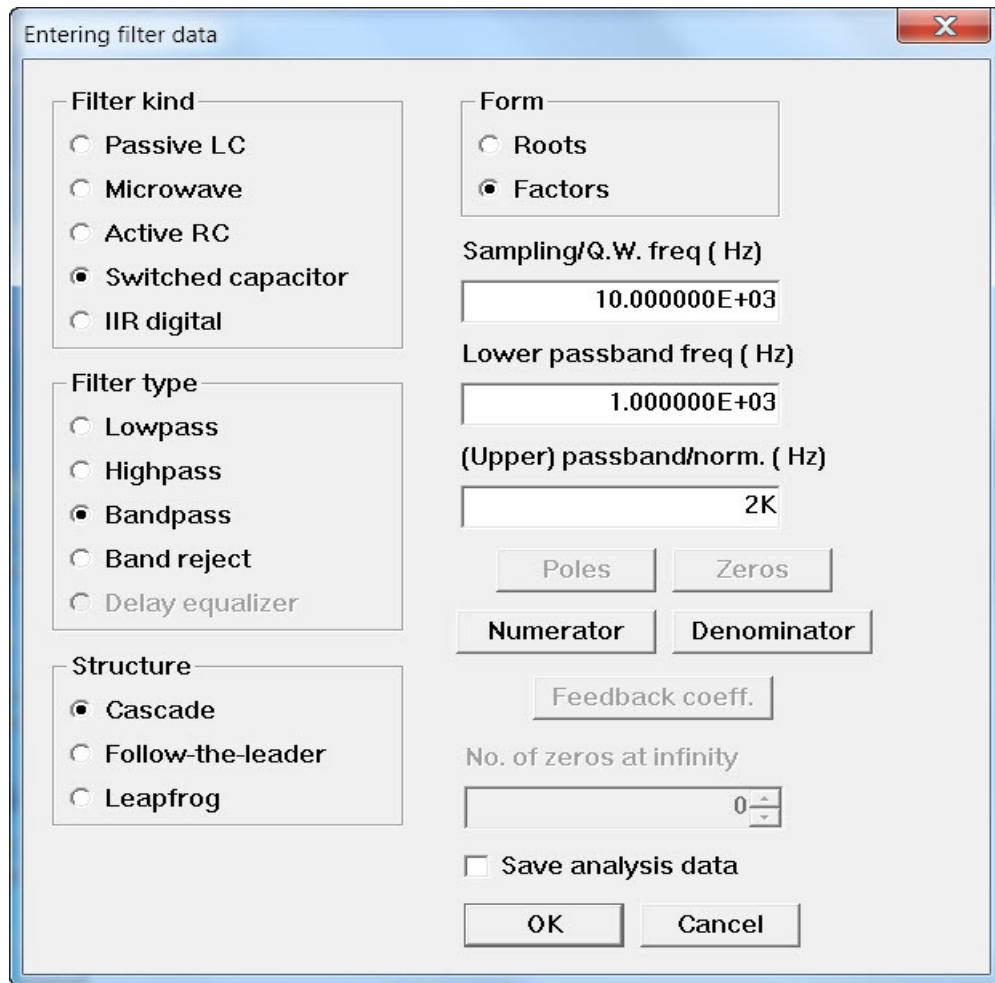
This option wrote the pole-zero data to a file, we called "swc.dat", which contains the data:

```

! pole-zero data for switched-capacitor filter
! <New Digital Filter>          30-Oct-2012 11:36
! numerator quadratic factors
1.444947D-02    0.000000D+00    -1.444947D-02
1.000000D+00   -1.793258D+00    1.000000D+00
1.000000D+00    2.223825D-02    1.000000D+00
1.000000D+00   -1.697347D+00    1.000000D+00
1.000000D+00   -3.798196D-01    1.000000D+00
! denominator quadratic factors
1.000000D+00   -1.074683D+00    7.388738D-01
1.000000D+00   -1.443922D+00    8.697867D-01
1.000000D+00   -7.178401D-01    8.170312D-01
1.000000D+00   -1.596906D+00    9.698315D-01
1.000000D+00   -5.944610D-01    9.513816D-01

```

Next we terminate **Filsyn** and start it again but this time using the **Analysis->Filter->New** menu option. There we enter all the information, but when we get to the **Numerator** button, we click on **Read from file** button:



The 'Entering filter data' dialog box is used to configure filter parameters. It contains three main sections on the left: 'Filter kind', 'Filter type', and 'Structure'. The 'Filter kind' section has radio buttons for Passive LC, Microwave, Active RC, Switched capacitor (selected), and IIR digital. The 'Filter type' section has radio buttons for Lowpass, Highpass, Bandpass (selected), Band reject, and Delay equalizer. The 'Structure' section has radio buttons for Cascade (selected), Follow-the-leader, and Leapfrog. On the right, the 'Form' section has radio buttons for Roots and Factors (selected). Below this are input fields for 'Sampling/Q.W. freq (Hz)' (10.000000E+03), 'Lower passband freq (Hz)' (1.000000E+03), and '(Upper) passband/norm. (Hz)' (2K). There are buttons for 'Poles', 'Zeros', 'Numerator', 'Denominator', and 'Feedback coeff.'. A 'No. of zeros at infinity' spinner is set to 0. A 'Save analysis data' checkbox is unchecked. 'OK' and 'Cancel' buttons are at the bottom.

Entering filter data

Filter kind

- ☐ Passive LC
- ☐ Microwave
- ☐ Active RC
- ☒ Switched capacitor
- ☐ IIR digital

Filter type

- ☐ Lowpass
- ☐ Highpass
- ☒ Bandpass
- ☐ Band reject
- ☐ Delay equalizer

Structure

- ☒ Cascade
- ☐ Follow-the-leader
- ☐ Leapfrog

Form

- ☐ Roots
- ☒ Factors

Sampling/Q.W. freq (Hz): 10.000000E+03

Lower passband freq (Hz): 1.000000E+03

(Upper) passband/norm. (Hz): 2K

Poles Zeros

Numerator Denominator

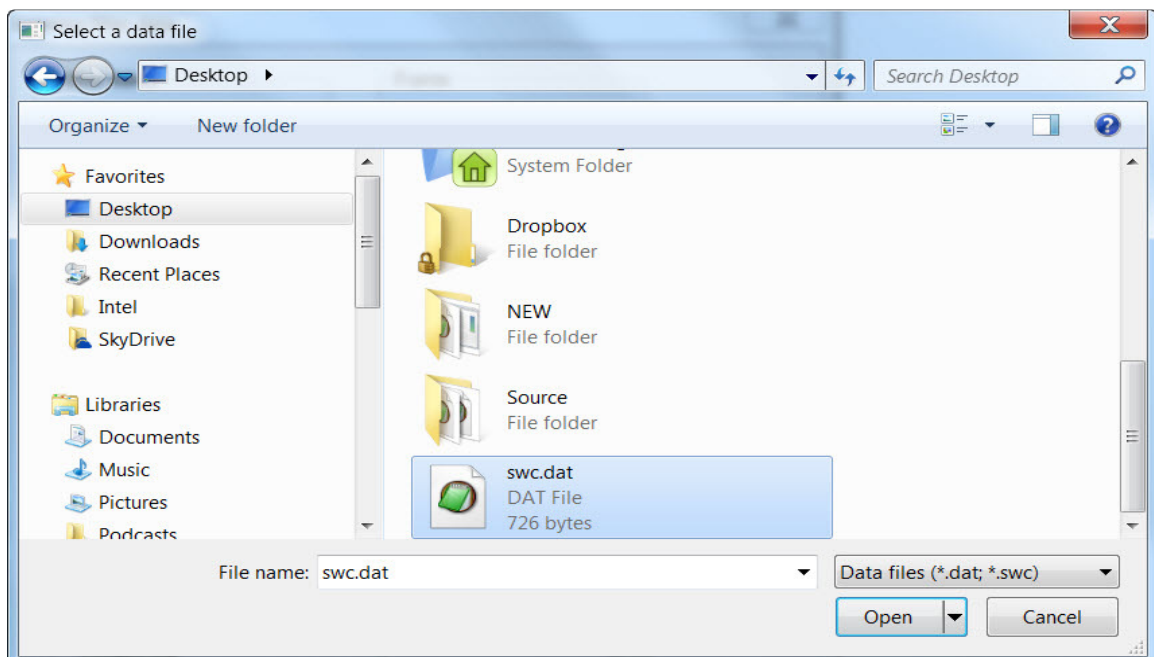
Feedback coeff.

No. of zeros at infinity: 0

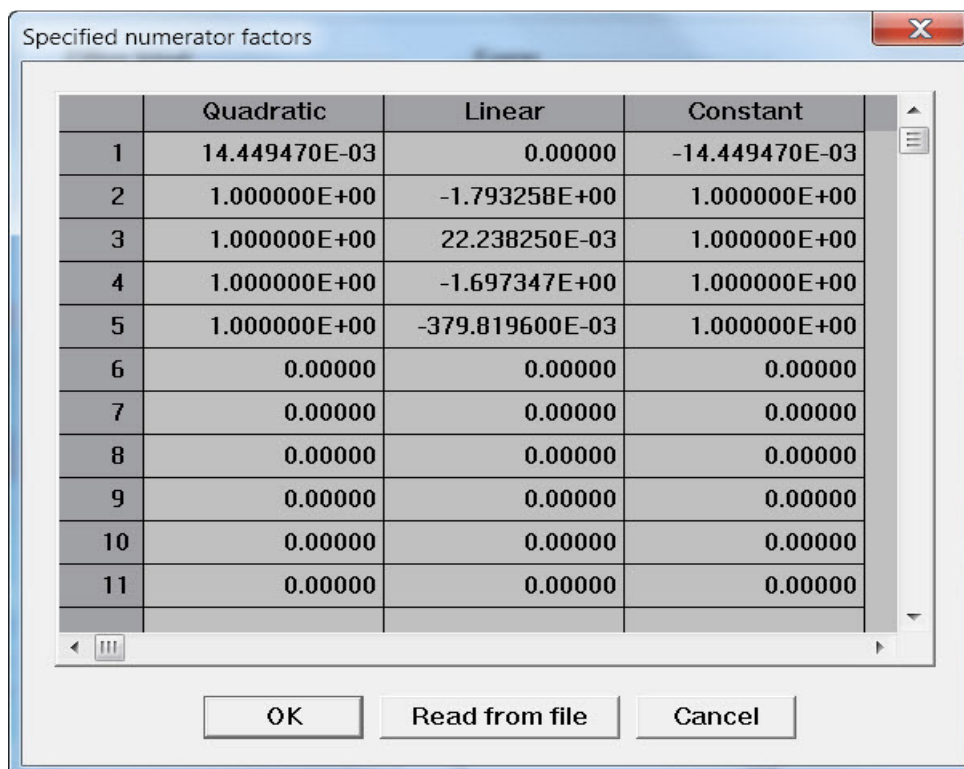
☐ Save analysis data

OK Cancel

This brings up the standard file selection window:



Clicking on the **Numerator** button again, we see the data already there:

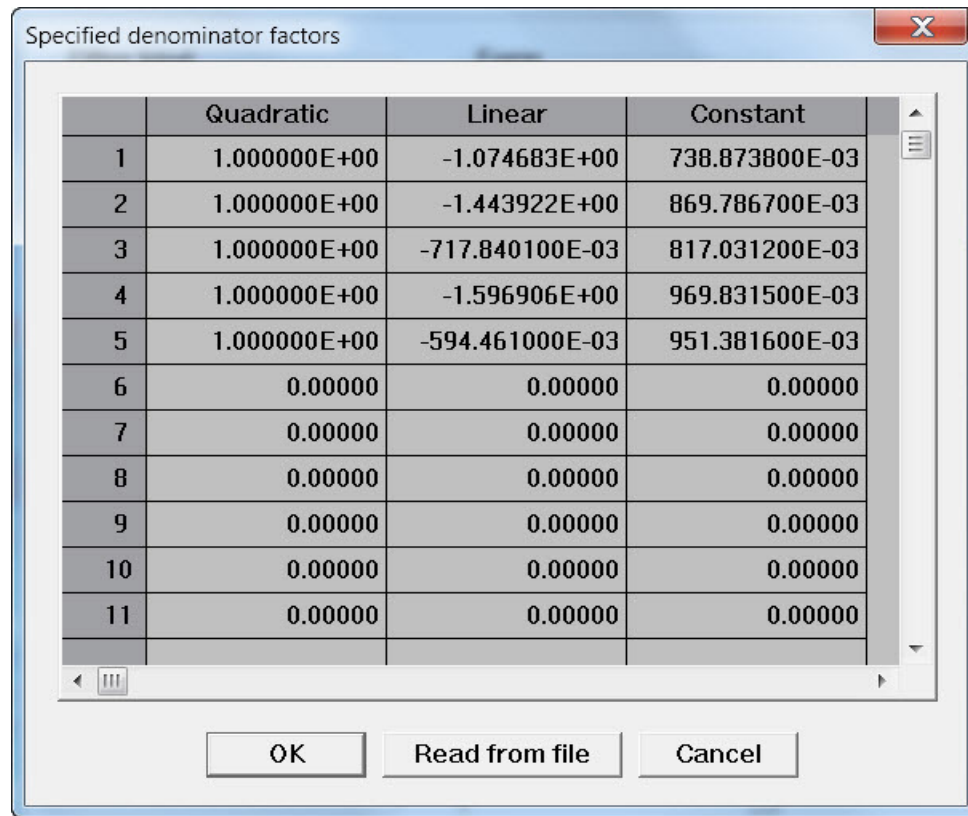


Specified numerator factors

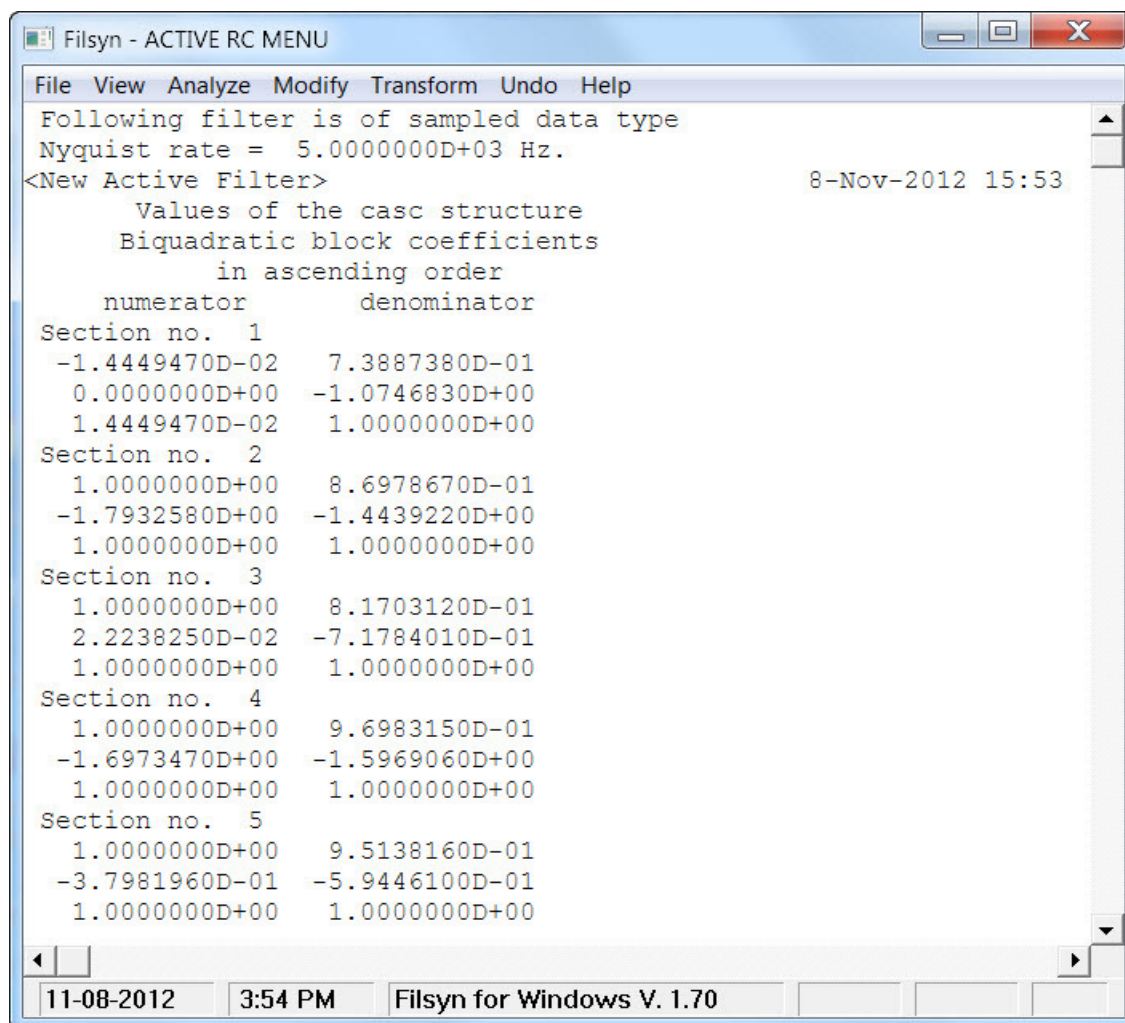
	Quadratic	Linear	Constant
1	14.449470E-03	0.00000	-14.449470E-03
2	1.000000E+00	-1.793258E+00	1.000000E+00
3	1.000000E+00	22.238250E-03	1.000000E+00
4	1.000000E+00	-1.697347E+00	1.000000E+00
5	1.000000E+00	-379.819600E-03	1.000000E+00
6	0.00000	0.00000	0.00000
7	0.00000	0.00000	0.00000
8	0.00000	0.00000	0.00000
9	0.00000	0.00000	0.00000
10	0.00000	0.00000	0.00000
11	0.00000	0.00000	0.00000

OK Read from file Cancel

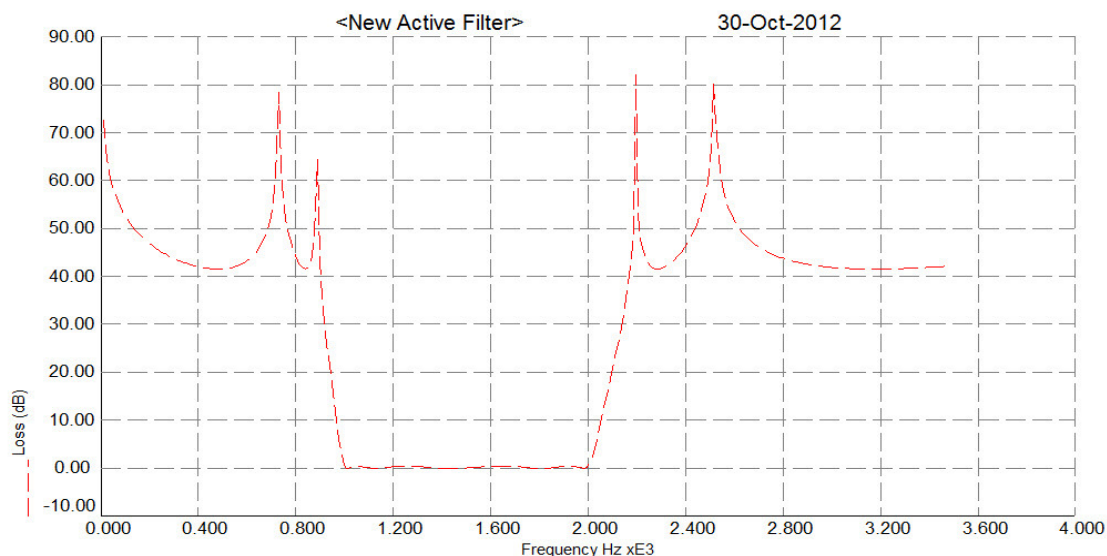
Clicking on the **Denominator** button and again selecting the **Read from file** option does not need a file selection, the file has already been selected and contains the proper data:



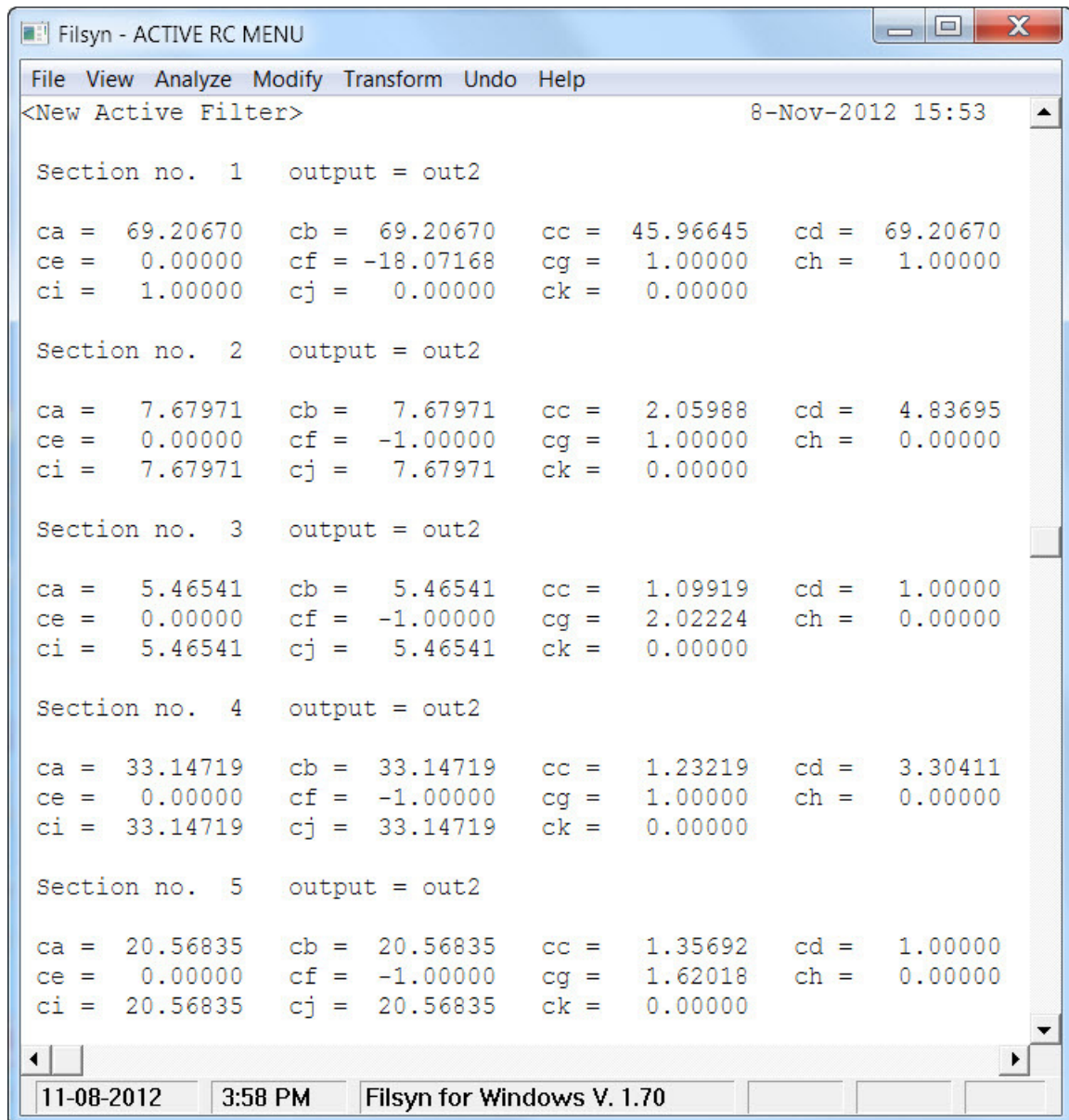
Clicking on the **OK** buttons, we get to the active RC analysis part of the program, with the filter data present:



Just to check the design, we show the frequency domain analysis that indicates perfect agreement and prewarping:



We can, of course, continue the design and obtain the capacitance values for the cascade implementation by using the **View->Design** menu option and letting the program determine the configurations of the sections:



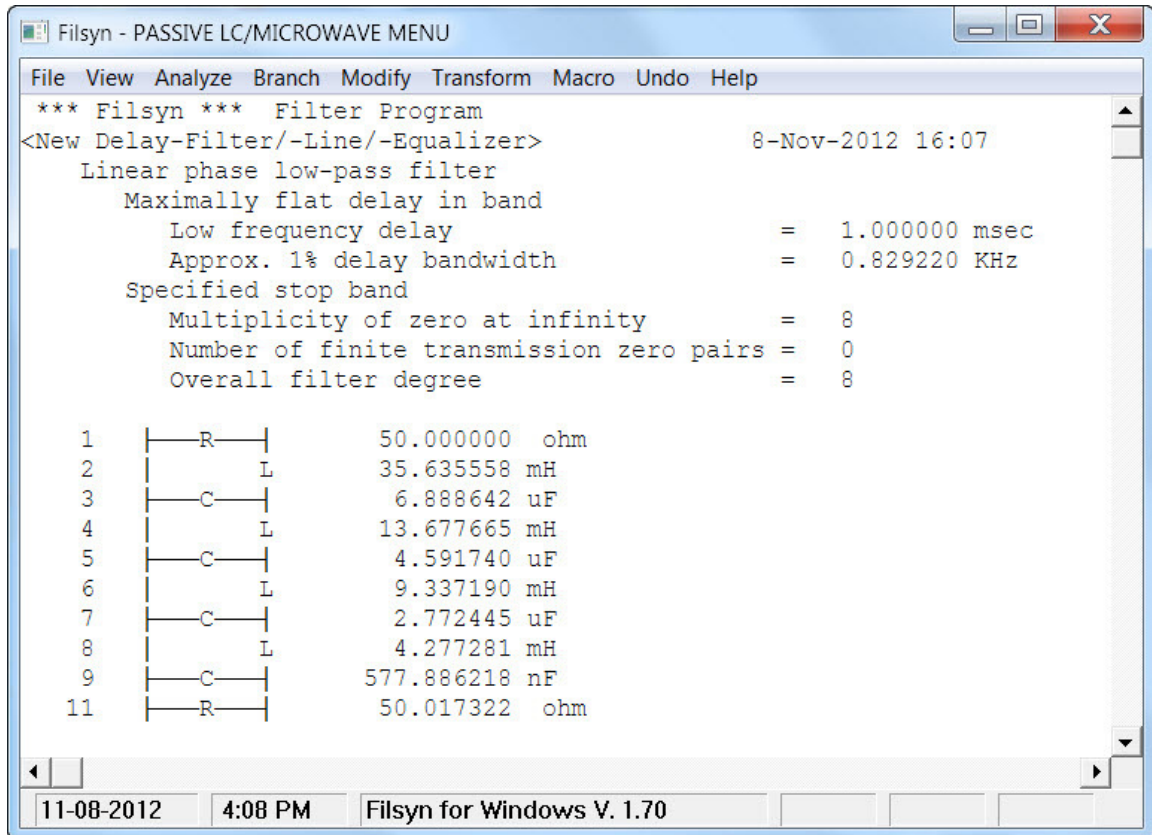
12. LINEAR PHASE FILTERS, DELAY LINES AND EQUALIZERS

The **Filsyn** program is capable of designing filters with linear phase (not considering the FIR class of digital filters). The linear phase, or more accurately, constant delay, may be approximated either in the maximally-flat manner, leading to the class of filters usually called *Bessel* (or *Thomson*) filters, or in an equal ripple manner. The resulting structure may be a delay line (zero loss at all frequencies) or a lowpass filter with either specified stopband or an equal-minima type stopband. Bandpass approximations can subsequently be obtained by the **shifted bandpass** transformation. Finally, the filters can be implemented either in passive LC or active RC forms.

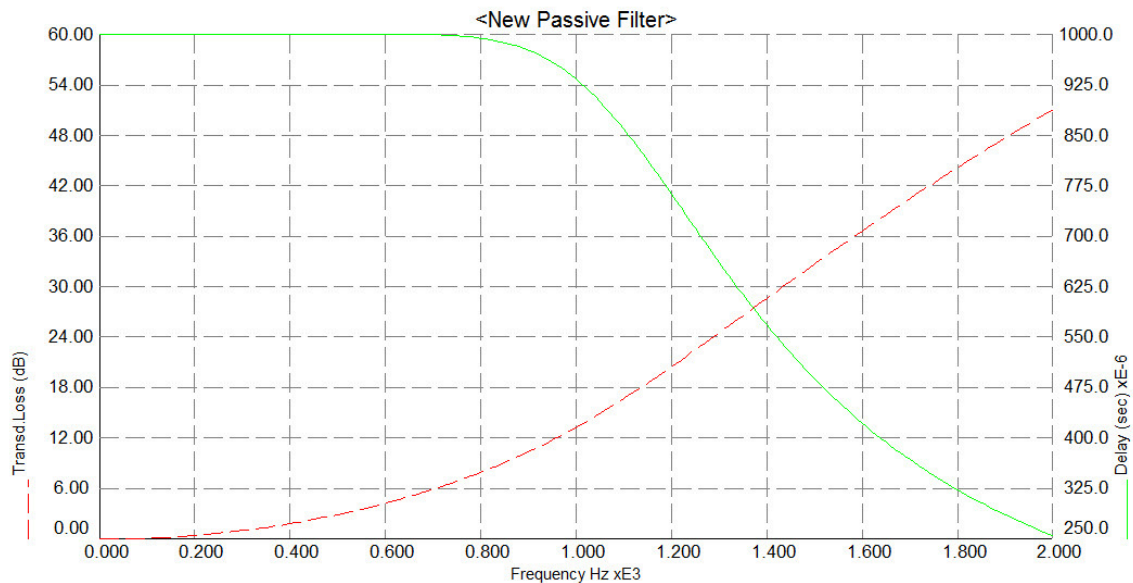
We reach the data input screen for this group of structures through the **Design->Delayline/equ->New.** menu option:

We have specified here an 8th order *Bessel* lowpass filter with 1 msec delay at zero frequency, in a passive LC implementation with 50 ohms terminations. After accepting all the default values, the resulting structure is familiar:

Linear phase filters...



The performance is also what we should expect:

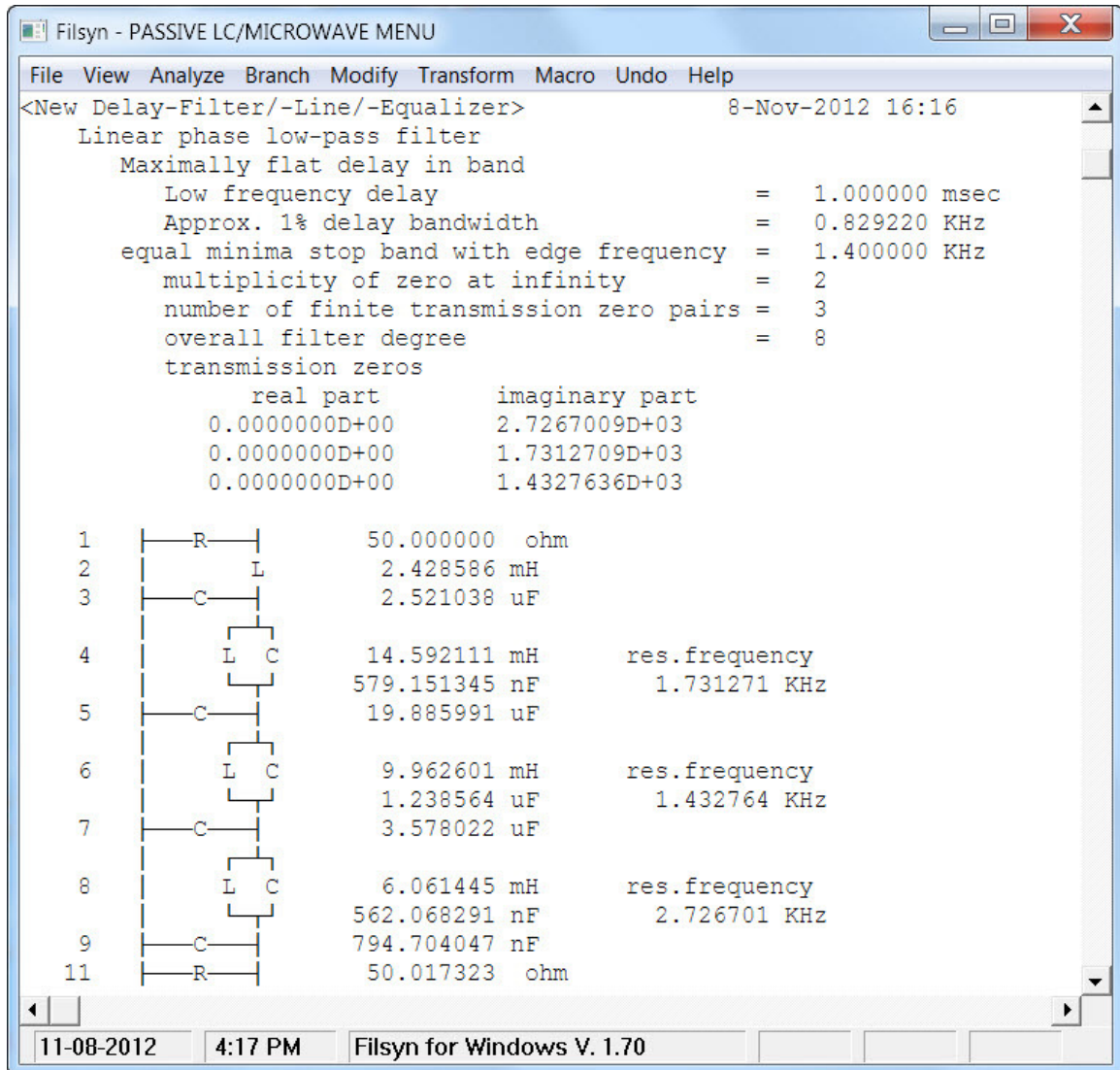


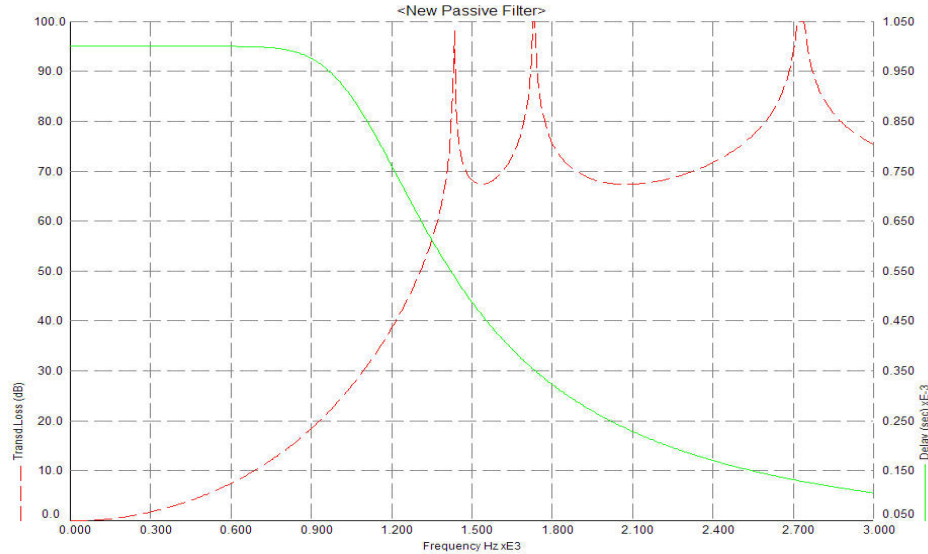
Next we request the same passband with an equal minima type stopband behavior from 1400Hz.

<New Delay-Filter/-Line/-Equalizer> X

<p>Delay type</p> <p><input checked="" type="radio"/> Max. flat</p> <p><input type="radio"/> Equal ripple</p> <p><input type="radio"/> Equalizer</p> <p>Implementation</p> <p><input checked="" type="radio"/> LC elements</p> <p><input type="radio"/> Active RC</p> <p><input type="radio"/> Microwave</p> <p><input type="radio"/> IIR digital</p> <p>Data</p> <p>FQ or FS (KHz)</p> <p><input type="text" value="0.00000"/></p> <p>Normalization freq (KHz)</p> <p><input type="text" value="0.00000"/></p> <p>Network type</p> <p><input checked="" type="radio"/> Filter</p> <p><input type="radio"/> Delay line</p>	<p>Delay value (sec)</p> <p><input type="text" value="1.000000E-03"/></p> <p>Stopband</p> <p><input checked="" type="radio"/> Equal min.</p> <p><input type="radio"/> Specified</p> <p>End of passband (KHz)</p> <p><input type="text" value="0.00000"/></p> <p>Start of stopband (KHz)</p> <p><input type="text" value="1.400000E+00"/></p> <p>Degree</p> <p><input type="text" value="8"/></p> <p>Zeros at infinity</p> <p><input type="text" value="0"/></p> <p>Transm. zeros</p> <p><input type="text" value="50.000000E+00"/></p> <p>Terminations</p> <p><input type="text" value="50.000000E+00"/></p> <p><input type="text" value="50.000000E+00"/></p>	<p>Q-value</p> <p><input type="text" value="0.00000000"/></p> <p>Hurwitz type</p> <p><input type="radio"/> Hurwitz</p> <p><input checked="" type="radio"/> Non Hurwitz</p> <p>Selection</p> <p><input checked="" type="radio"/> Auto</p> <p><input type="radio"/> Manual</p> <p>Indicators</p> <p>Shifted bandpass trans.</p> <p>Center frequency (KHz)</p> <p><input type="text" value="0.00000"/></p> <p><input type="checkbox"/> Odd parametric</p> <p><input type="checkbox"/> Save design data</p> <p>OK Cancel</p>
--	---	--

Note that we also specified a **Non Hurwitz** design. The reason for that is that with the default **Hurwitz** option selected, the resulting filter cannot be implemented in a passive LC form, using the computer-selected configuration. Manual synthesis might work, but is more complex. With the **Non Hurwitz** option selected, the computer-selected implementation works fine. The resulting filter is shown below, with the expected frequency domain behavior following.





Next we modify the design to yield an equal-ripple type delay performance. In the case of equal ripple delay the function is obtained by an iterative approximation method and for the procedure to converge the average delay τ_0 times the passband width f_B should be between

$$X < 2\pi\tau_0 f_B < 2X$$

where

$$X \cong 0.946N - 3.342 + 11.5/(N + 3.73)$$

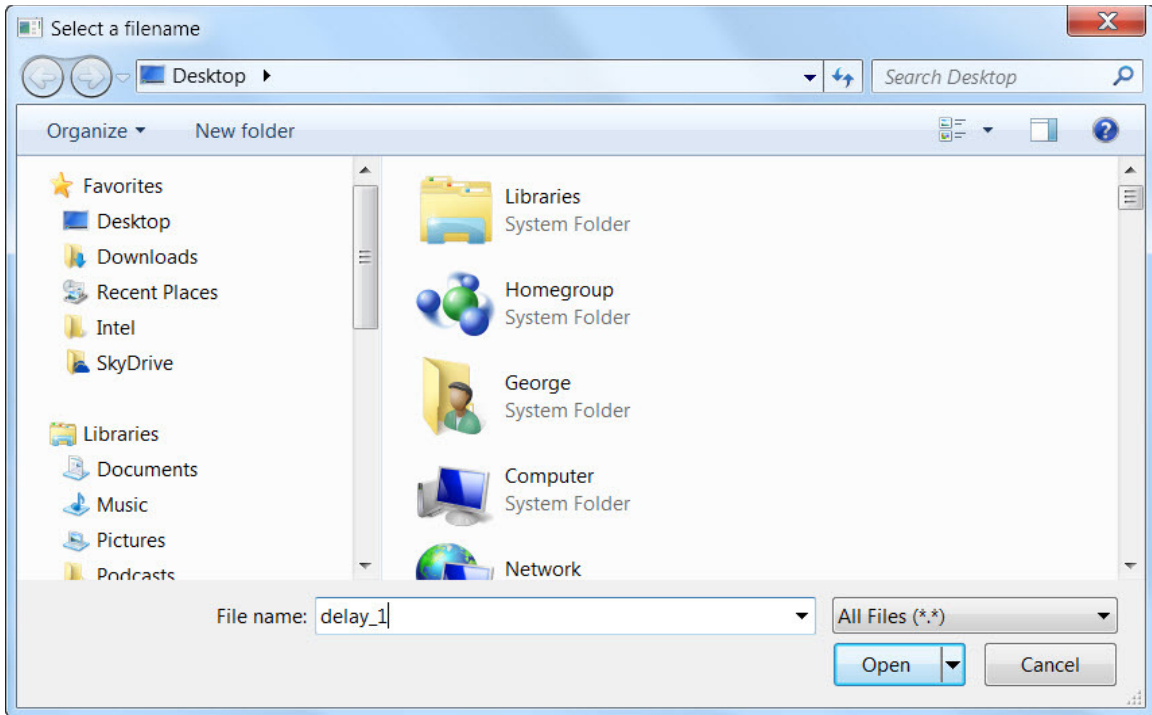
and N is the degree. If the quantity is close to the edge or outside of the range, the iteration will converge slowly or not at all. At the lower edge the delay ripple will be extremely small, while at the upper one it will be quite substantial (15% to 20%). All this is valid for filters. If we are dealing with a delay line, the value of X should be doubled.

In our case we have selected 1400 Hz as the end of the equal-ripple delay range, that is about two thirds of the way to the upper limit above. Note that the passband edge and the beginning of the stopband are at the same frequency and they may even overlap, since one refers to the delay, the other to the loss.

<New Delay-Filter/-Line/-Equalizer> X

<p>Delay type</p> <p><input type="radio"/> Max. flat</p> <p><input checked="" type="radio"/> Equal ripple</p> <p><input type="radio"/> Equalizer</p>	<p>Delay value (sec)</p> <p><input type="text" value="1.000000E-03"/></p>	<p>Q-value</p> <p><input type="text" value="0.00000000"/></p>
<p>Implementation</p> <p><input checked="" type="radio"/> LC elements</p> <p><input type="radio"/> Active RC</p> <p><input type="radio"/> Microwave</p> <p><input type="radio"/> IIR digital</p>	<p>Stopband</p> <p><input checked="" type="radio"/> Equal min.</p> <p><input type="radio"/> Specified</p>	<p>Hurwitz type</p> <p><input type="radio"/> Hurwitz</p> <p><input checked="" type="radio"/> Non Hurwitz</p>
<p>Data</p> <p>FQ or FS (KHz)</p> <p><input type="text" value="0.00000"/></p> <p>Normalization freq (KHz)</p> <p><input type="text" value="0.00000"/></p>	<p>End of passband (KHz)</p> <p><input type="text" value="1.400000E+00"/></p> <p>Start of stopband (KHz)</p> <p><input type="text" value="1.400000E+00"/></p>	<p>Selection</p> <p><input checked="" type="radio"/> Auto</p> <p><input type="radio"/> Manual</p>
<p>Network type</p> <p><input checked="" type="radio"/> Filter</p> <p><input type="radio"/> Delay line</p>	<p>Degree</p> <p><input type="text" value="8"/></p> <p>Zeros at infinity</p> <p><input type="text" value="0"/></p> <p>Transm. zeros</p> <p><input type="text" value="50.000000E+00"/></p> <p>Terminations</p> <p><input type="text" value="50.000000E+00"/></p>	<p>Indicators</p> <p>Shifted bandpass trans.</p> <p>Center frequency (KHz)</p> <p><input type="text" value="0.00000"/></p> <p><input type="checkbox"/> Odd parametric</p> <p><input checked="" type="checkbox"/> Save design data</p> <p><input type="button" value="OK"/> <input type="button" value="Cancel"/></p>

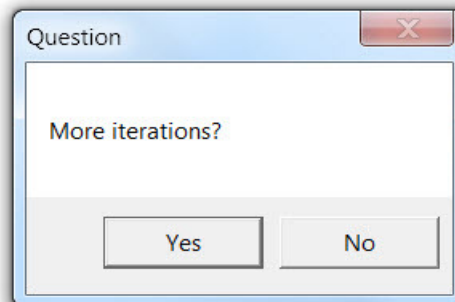
The **Non-Hurwitz** selection is again necessary here, unless we are willing to perform a manual synthesis. We have selected to save the design data to a file for later use, which brings up the standard file name selection window:



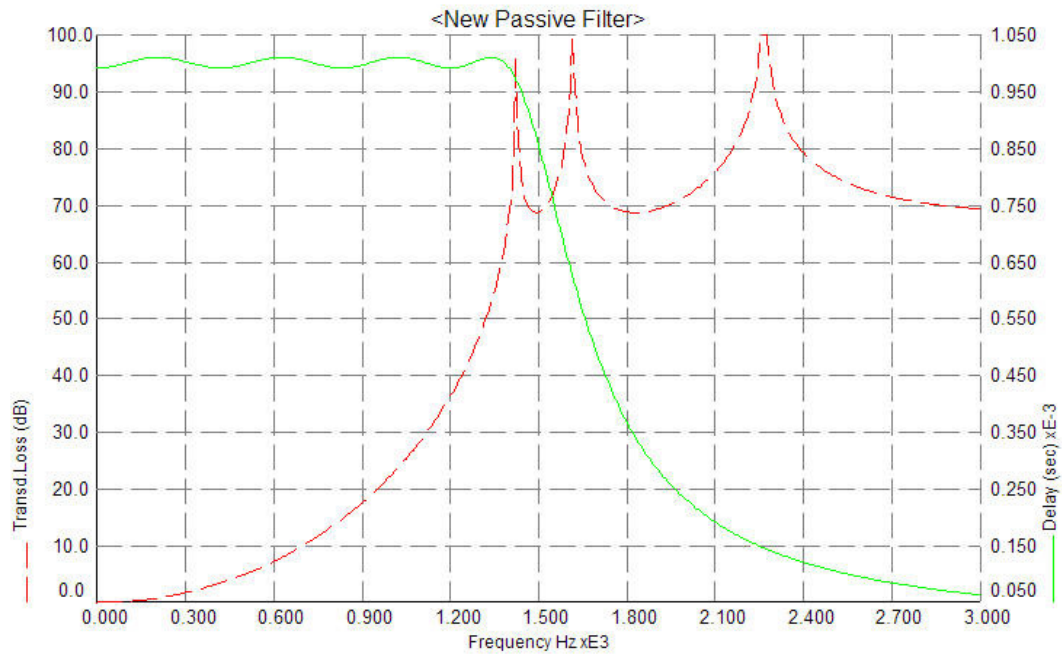
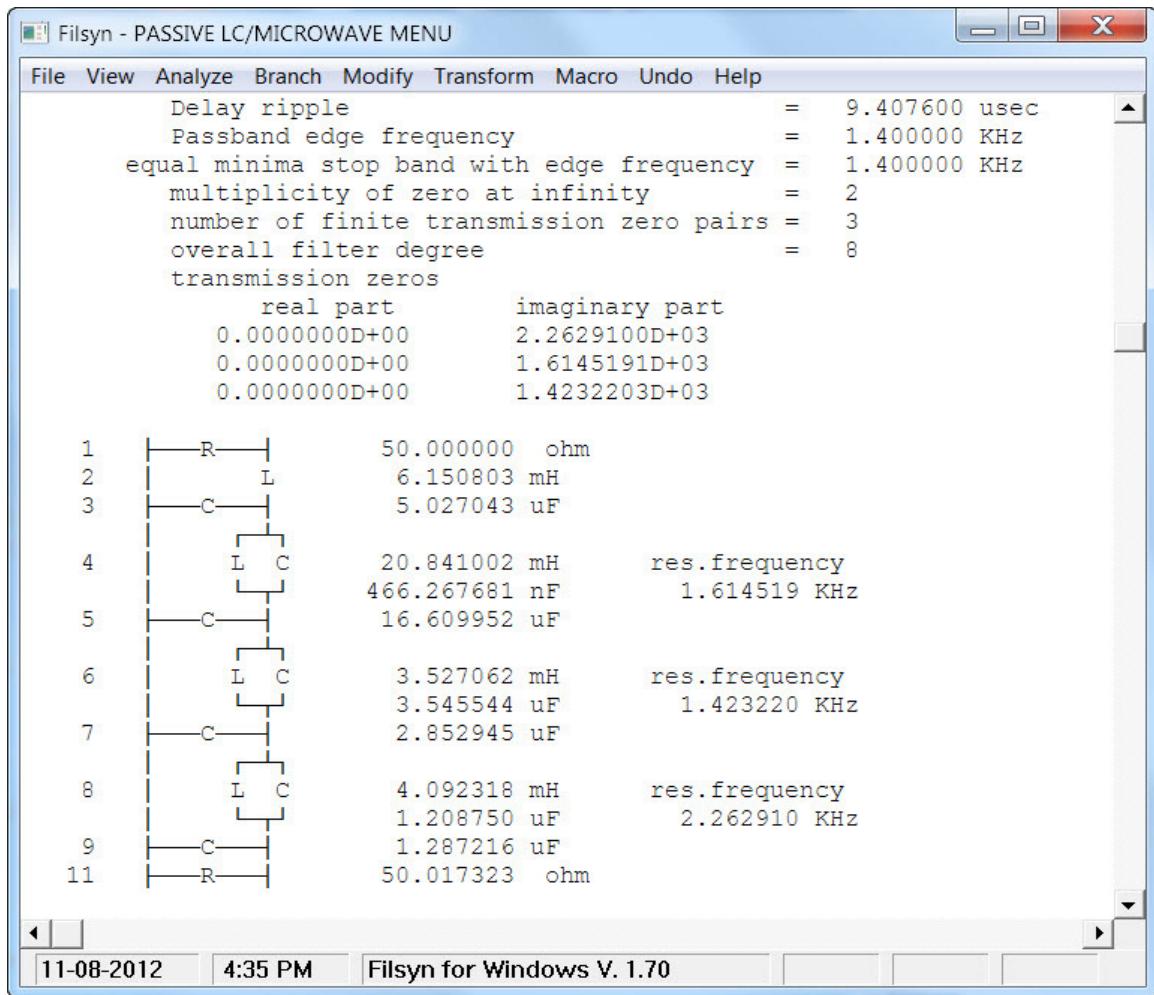
The optimization procedure takes very few iterations, but we can perform additional ones if we desire. If the printed rms (root mean square) error is less than about $1.e-3$, the iteration has converged for all practical purposes:

```
File  Design  Analysis  Help
*** Filsyn ***  Filter Program
<New Delay-Filter/-Line/-Equalizer>          8-Nov-2012 16:31
  Linear phase low-pass filter
    Equal ripple delay passband
      Low frequency delay                    = 1.000000 msec
      Upper passband edge frequency          = 1.400000 KHz

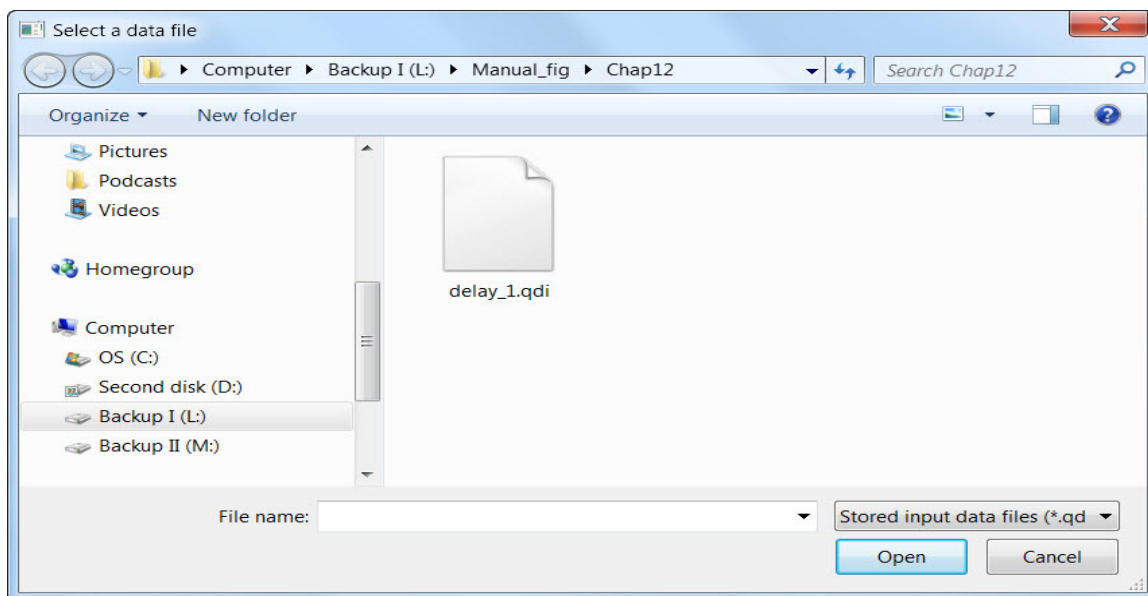
at iteration no. 2  delay ripple: 9.4212D-06  rms error: 2.0612D-03
at iteration no. 2  delay ripple: 9.4076D-06  rms error: 5.5531D-10
at iteration no. 2  delay ripple: 9.4076D-06  rms error: 5.9819D-15
```



The resulting filter looks very much like the one before and the frequency domain performance is also what is to be expected:



As our next example, we keep the specification as above with minor changes, so we recall the previously saved design data first using **Design->Delayline/equ->Recall** menu:

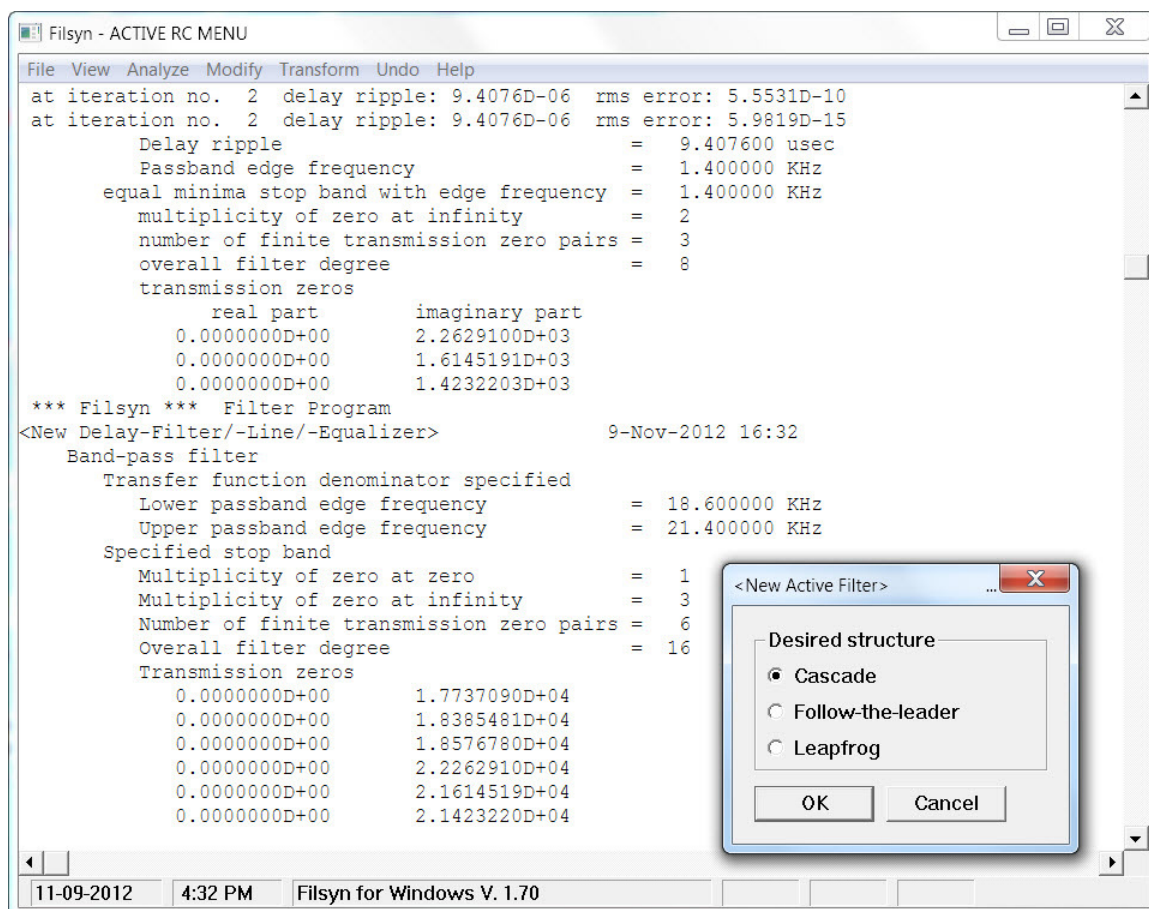


Next we modify the specification to select an active RC implementation and specify a **Shifted bandpass** transformation with a 20 KHz center frequency. Note that the **Hurwitz-Non Hurwitz** option is now grayed out as being completely immaterial.

<New Delay-Filter/-Line/-Equalizer> X

Delay type <input type="radio"/> Max. flat <input checked="" type="radio"/> Equal ripple <input type="radio"/> Equalizer	Delay value (sec) <input type="text" value="1.000000E-03"/>	Q-value <input type="text" value="0.00000000"/>
Implementation <input type="radio"/> LC elements <input checked="" type="radio"/> Active RC <input type="radio"/> Microwave <input type="radio"/> IIR digital	Stopband <input checked="" type="radio"/> Equal min. <input type="radio"/> Specified	Hurwitz type <input checked="" type="radio"/> Hurwitz <input type="radio"/> Non Hurwitz
Data FQ or FS (Hz) <input type="text" value="0.00000"/>	End of passband (Hz) <input type="text" value="1.400000E+03"/>	Selection <input checked="" type="radio"/> Auto <input type="radio"/> Manual
Normalization freq (Hz) <input type="text" value="0.00000"/>	Start of stopband (Hz) <input type="text" value="1.400000E+03"/>	Indicators
Network type <input checked="" type="radio"/> Filter <input type="radio"/> Delay line	Degree <input type="text" value="8"/>	Shifted bandpass trans.
	Zeros at infinity <input type="text" value="0"/>	Center frequency (Hz) <input type="text" value="20K"/>
	Transm. zeros <input type="text" value="50.000000E+00"/>	<input type="checkbox"/> Odd parametric
	Terminations <input type="text" value="50.000000E+00"/>	<input type="checkbox"/> Save design data
		<input type="button" value="OK"/> <input type="button" value="Cancel"/>

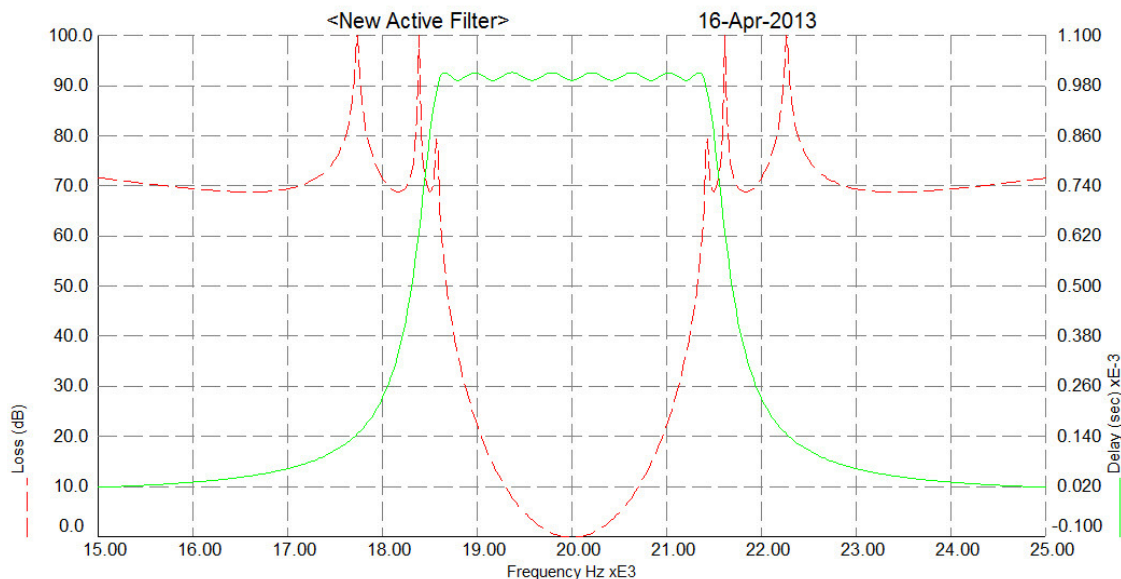
After clicking on the OK button we select the leapfrog implementation of the active RC filter and use the computer-selected sequence of transmission zeros (not shown):



The complete filter now is:

Filsyn - ACTIVE RC MENU			
File View Analyze Modify Transform Undo Help			
Values of the casc structure			
Biquadratic block coefficients			
in ascending order			
	numerator	denominator	
Section no. 1			
	5.8246396D+08	1.5461839D+10	
	0.0000000D+00	4.6842851D+03	
	0.0000000D+00	1.0000000D+00	
Section no. 2			
	0.0000000D+00	1.4809219D+10	
	6.0201906D+03	4.5826841D+03	
	0.0000000D+00	1.0000000D+00	
Section no. 3			
	4.1460941D+09	1.6135458D+10	transmission zero
	0.0000000D+00	4.6842851D+03	1.7737090D+04 Hz
	3.3382179D-01	1.0000000D+00	
Section no. 4			
	8.7922604D+09	1.4194172D+10	transmission zero
	0.0000000D+00	4.2811940D+03	1.8385481D+04 Hz
	6.5885647D-01	1.0000000D+00	
Section no. 5			
	6.6710031D+09	1.6815890D+10	transmission zero
	0.0000000D+00	4.5826841D+03	2.2262910D+04 Hz
	3.4093183D-01	1.0000000D+00	
Section no. 6			
	1.2445487D+10	1.7483365D+10	transmission zero
	0.0000000D+00	4.2811940D+03	2.1614519D+04 Hz
	6.7477816D-01	1.0000000D+00	
Section no. 7			
	1.3773009D+10	1.3640289D+10	transmission zero
	0.0000000D+00	3.3223914D+03	1.8576780D+04 Hz
	1.0109467D+00	1.0000000D+00	
Section no. 8			
	1.8308542D+10	1.8105811D+10	transmission zero
	0.0000000D+00	3.3223914D+03	2.1423220D+04 Hz
	1.0104726D+00	1.0000000D+00	
11-09-2012 4:34 PM Filsyn for Windows V. 1.70			

while the analysis results are:



Our next example is an LC filter with flat delay and specified stopband:

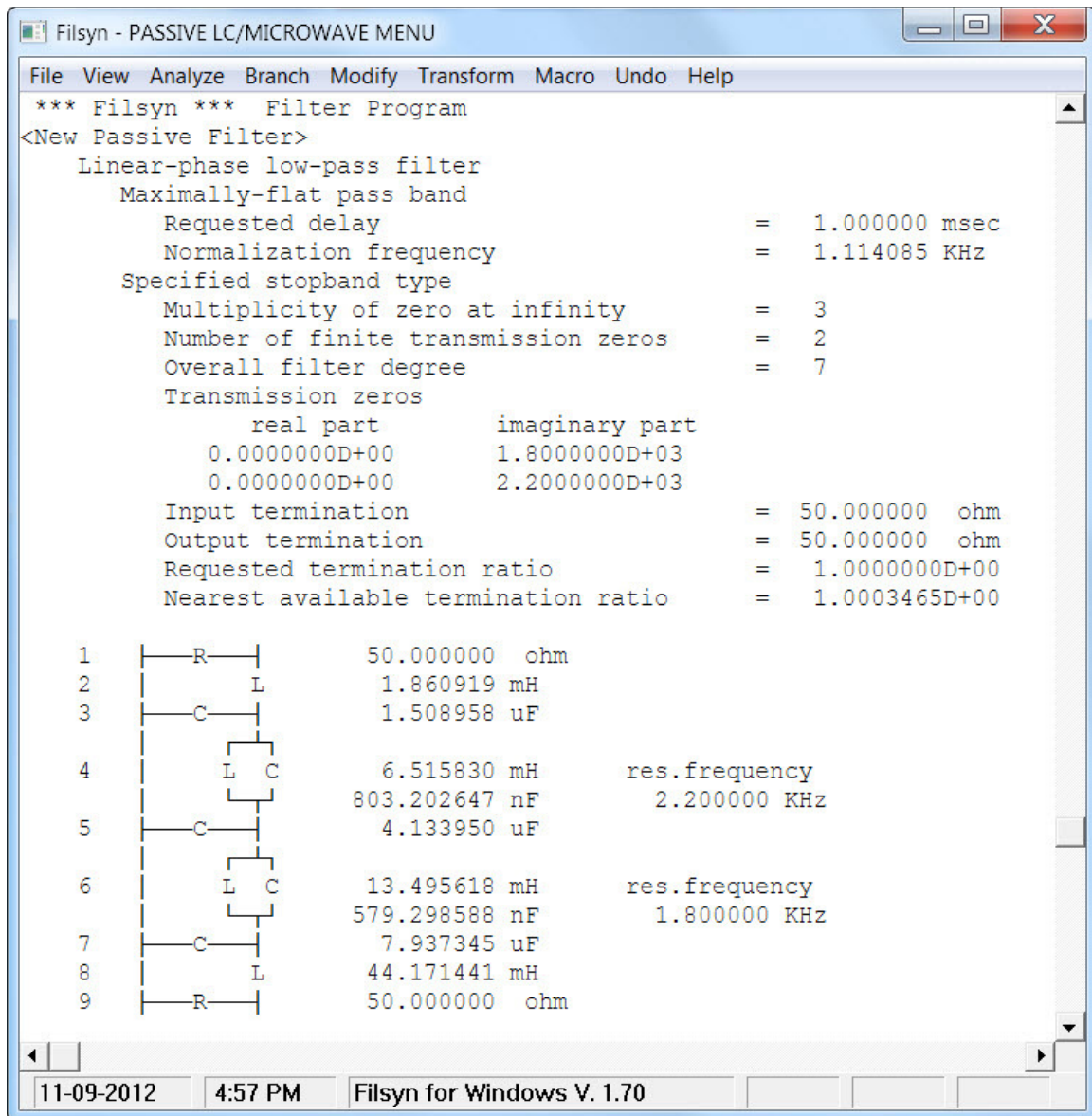
<New Delay-Filter/-Line/-Equalizer> [X]

Delay type <input checked="" type="radio"/> Max. flat <input type="radio"/> Equal ripple <input type="radio"/> Equalizer	Delay value (sec) <input type="text" value="1.000000E-03"/>	Q-value <input type="text" value="0.00000000"/>
Implementation <input checked="" type="radio"/> LC elements <input type="radio"/> Active RC <input type="radio"/> Microwave <input type="radio"/> IIR digital	Stopband <input type="radio"/> Equal min. <input checked="" type="radio"/> Specified	Hurwitz type <input checked="" type="radio"/> Hurwitz <input type="radio"/> Non Hurwitz
<input type="button" value="Data"/>	End of passband (Hz) <input type="text" value="0.00000"/>	Selection <input checked="" type="radio"/> Auto <input type="radio"/> Manual
FQ or FS (Hz) <input type="text" value="0.00000"/>	Start of stopband (Hz) <input type="text" value="0.00000"/>	<input type="button" value="Indicators"/>
Normalization freq (Hz) <input type="text" value="0.00000"/>	Degree <input type="text" value="0"/>	Shifted bandpass trans. Center frequency (Hz) <input type="text" value="0.00000"/>
Network type <input checked="" type="radio"/> Filter <input type="radio"/> Delay line	Zeros at infinity <input type="text" value="3"/>	<input type="checkbox"/> Odd parametric <input type="checkbox"/> Save design data
	<input type="button" value="Transm. zeros"/>	<input type="button" value="OK"/> <input type="button" value="Cancel"/>
	Terminations <input type="text" value="50.000000E+00"/> <input type="text" value="50.000000E+00"/>	

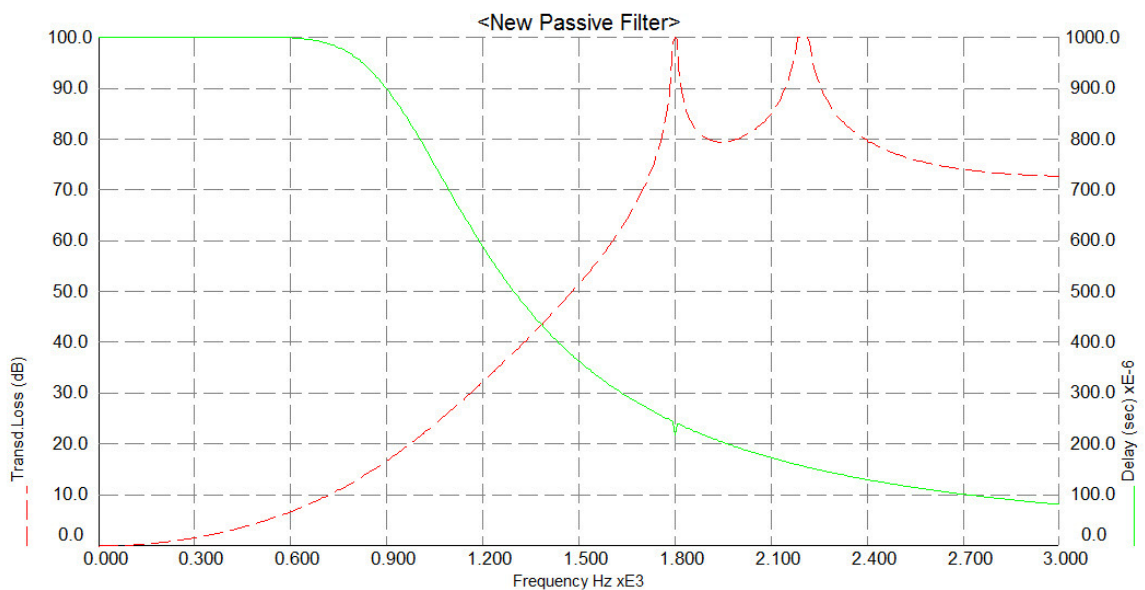
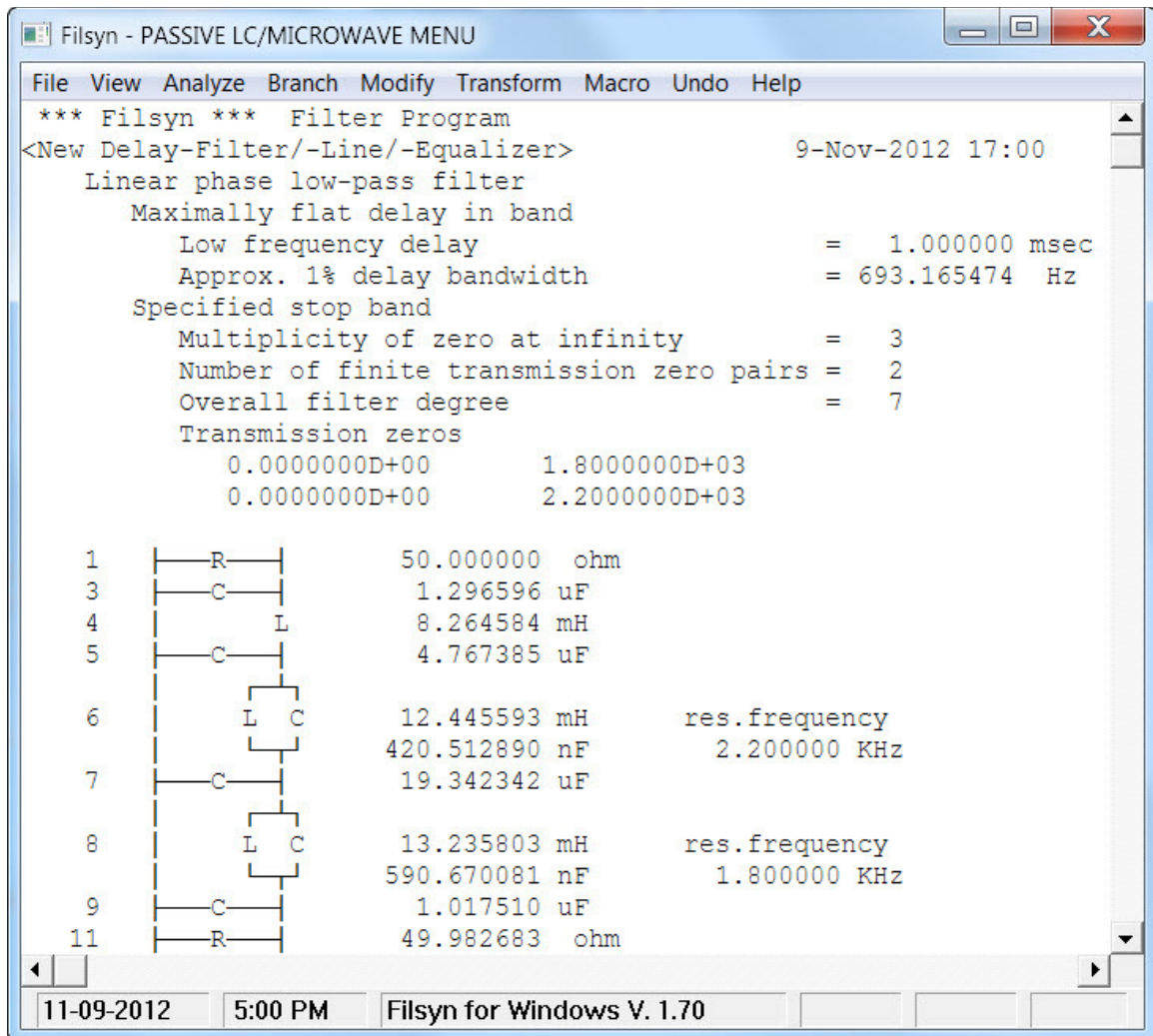
We have two finite transmission zeros at 1.8 KHz and 2.2 KHz entered manually here and three zeros at infinity:

	Real part	Imaginary part
1	0.00000	1.800000E+03
2	0.00000	2.200000E+03
3	0.00000	0.00000
4	0.00000	0.00000
5	0.00000	0.00000
6	0.00000	0.00000
7	0.00000	0.00000
8	0.00000	0.00000
9	0.00000	0.00000
10	0.00000	0.00000
11	0.00000	0.00000

yielding a 7th order filter:



The synthesis above was done manually, because the computer selected structure was again not realizable. If we were to specify the **Non Hurwitz** option (see below), we would have obtained the following circuit automatically. Note that not only is the circuit different, but it has one fewer inductor, and surprisingly enough, the frequency domain behavior of the two circuits are identical.



These were all linear phase filters. The same approach can be used for the design of delay lines. As the next example, consider a 6th order delay line with equal-ripple delay up to 1 KHz in passive LC form:

<New Delay-Filter/-Line/-Equalizer>

Delay type

- ☐ Max. flat
- ☒ Equal ripple
- ☐ Equalizer

Implementation

- ☒ LC elements
- ☐ Active RC
- ☐ Microwave
- ☐ IIR digital

Data

FQ or FS (Hz)

0.00000

Normalization freq (Hz)

0.00000

Network type

- ☐ Filter
- ☒ Delay line

Delay value (sec)

1.000000E-03

Stopband

- ☐ Equal min.
- ☒ Specified

End of passband (Hz)

1.000000E+03

Start of stopband (Hz)

0.00000

Degree

6

Zeros at infinity

0

Transm. zeros

Terminations

50.000000E+00

50.000000E+00

Q-value

0.00000000

Hurwitz type

- ☒ Hurwitz
- ☐ Non Hurwitz

Selection

- ☒ Auto
- ☐ Manual

Indicators

Shifted bandpass trans.

Center frequency (Hz)

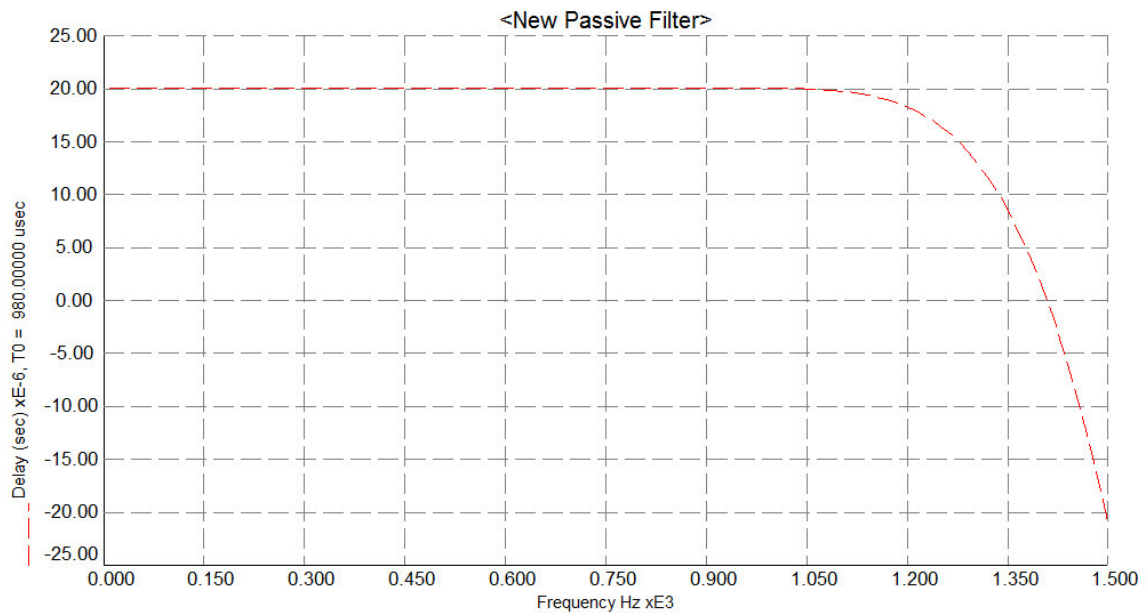
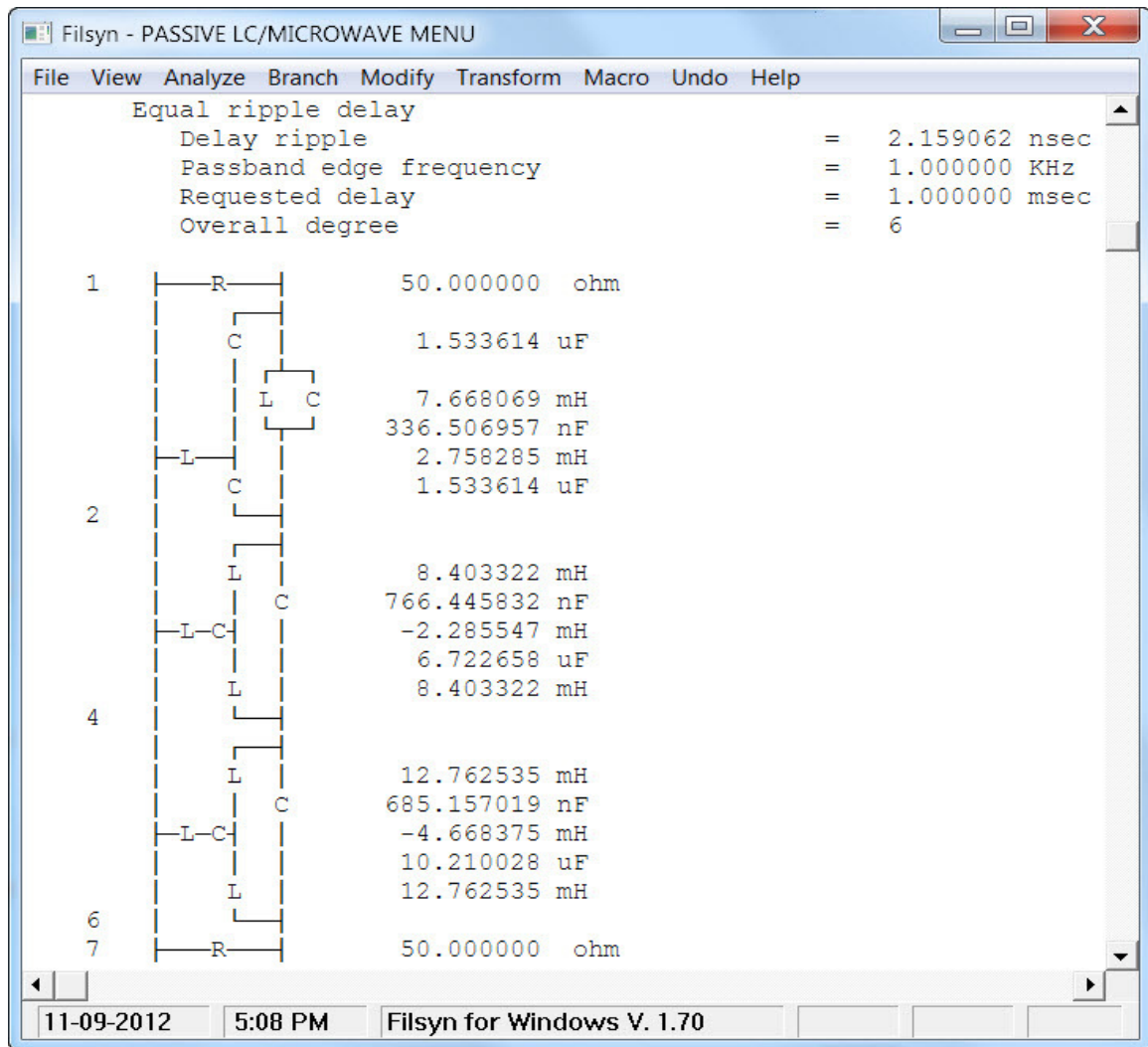
0.00000

☐ Odd parametric

☐ Save design data

OK **Cancel**

The implementation is as follows, with the computed performance next. The loss, of course, is identically zero.



In parenthesis, please note that the negative inductors have positive series inductors on both sides (once we interchange the negative inductor with the series capacitor) and these three inductors may be implemented as a pair of coupled inductors with a specific coupling factor.

The last set of options we have not demonstrated yet is the design of delay equalizers that equalize tabulated delay characteristics, or implement specified pole-zero locations. For that purpose we select the **Equalizer** option under the **Delay type** header, which activates the **Data** button. Clicking on this button we can enter the frequency-delay value pairs that define the delay to be equalized, either from the keyboard, or read them into the program from a previously prepared simple text file. The file must have a “.tab” extension and contains the data in the following format. The first line contains the title. Subsequent lines contain frequency-delay pairs in scientific notation, but otherwise in free form. Blank lines and lines that begin with an “!” are ignored. The first 501 values are read in and used for the equalization.

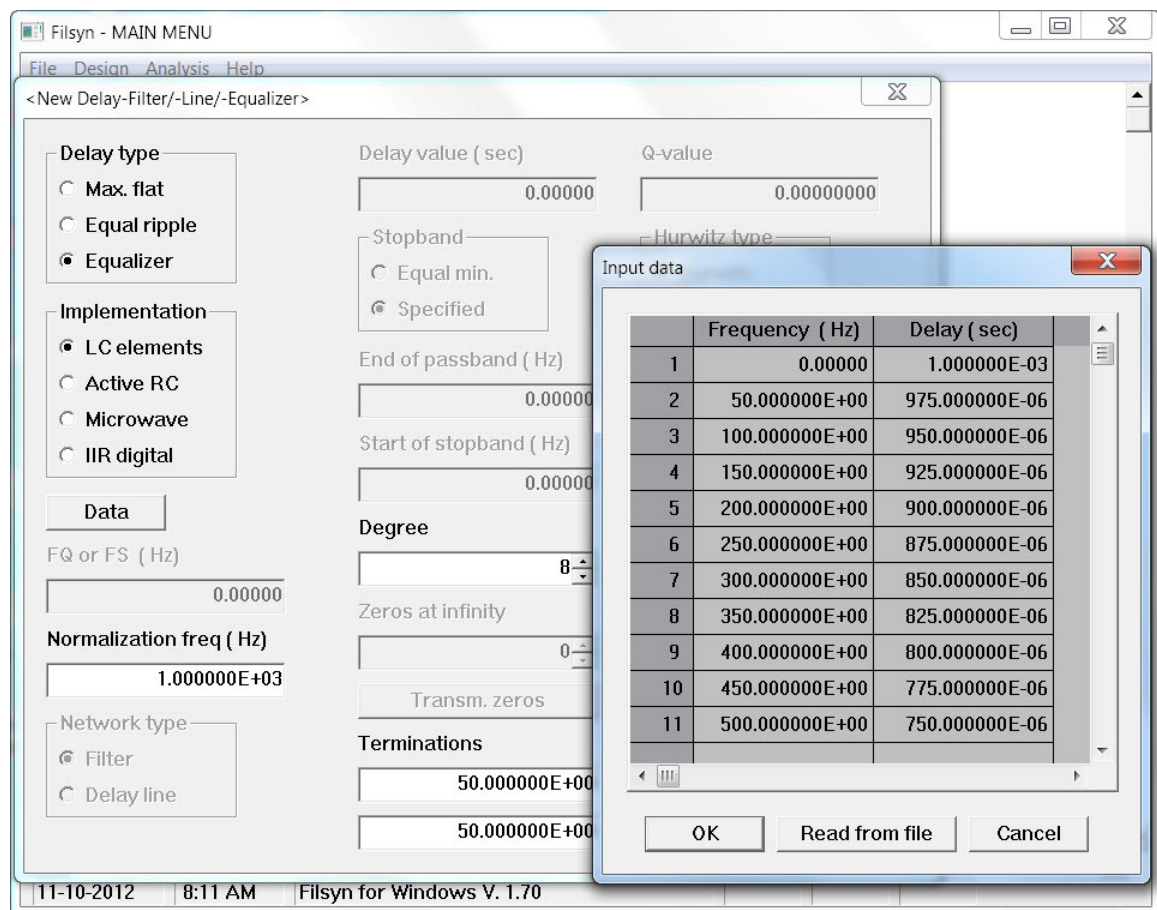
As an example, consider the following file, called *test.tab*:

Linear delay case

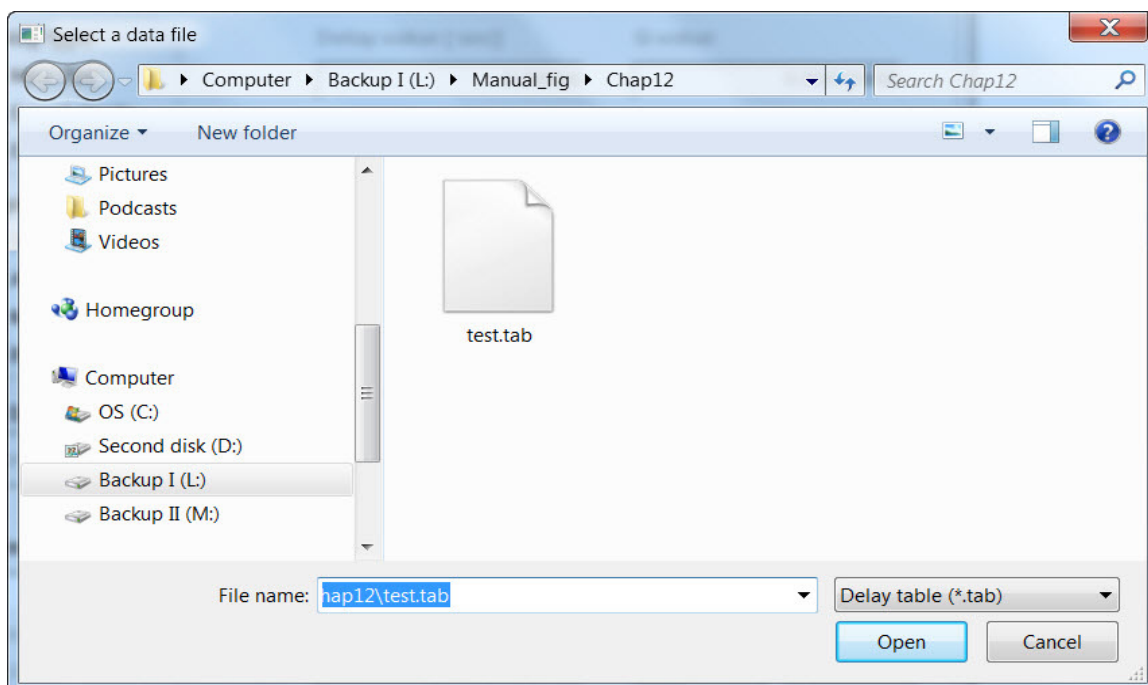
0.0	100.0e-5
50.0	97.5e-5
100.0	95.0e-5
150.0	92.5e-5
200.0	90.0e-5
250.0	87.5e-5
300.0	85.0e-5
350.0	82.5e-5
400.0	80.0e-5
450.0	77.5e-5
500.0	75.0e-5
550.0	72.5e-5
600.0	70.0e-5
650.0	67.5e-5
700.0	65.0e-5
750.0	62.5e-5
800.0	60.0e-5
850.0	57.5e-5
900.0	55.0e-5
950.0	52.5e-5
1000.0	50.0e-5
1050.0	47.5e-5
1100.0	45.0e-5
1150.0	42.5e-5
1200.0	40.0e-5
1250.0	37.5e-5
1300.0	35.0e-5

1350.0	32.5e-5
1400.0	30.0e-5
1450.0	27.5e-5
1500.0	25.0e-5
1550.0	22.5e-5
1600.0	20.0e-5
1650.0	17.5e-5
1700.0	15.0e-5
1750.0	12.5e-5
1800.0	10.0e-5
1850.0	7.5e-5
1900.0	5.0e-5
1950.0	2.5e-5
2000.0	0.0

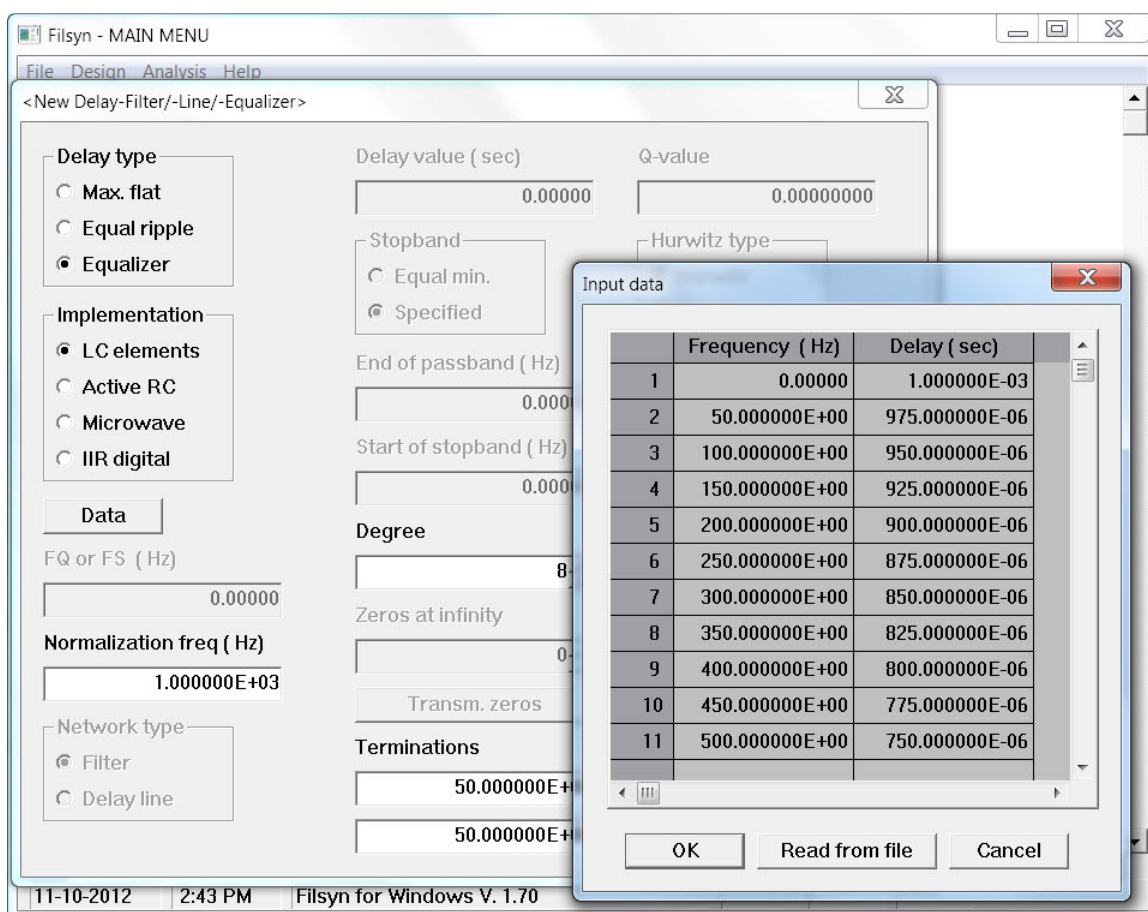
As we can see, this defines a linearly decreasing delay, hence when we equalize this, we shall get a linearly increasing delay. When we click on the **Data** button, we can enter the data from the keyboard:



If we then click on the **Read from file** button, it brings up the standard file selection menu:

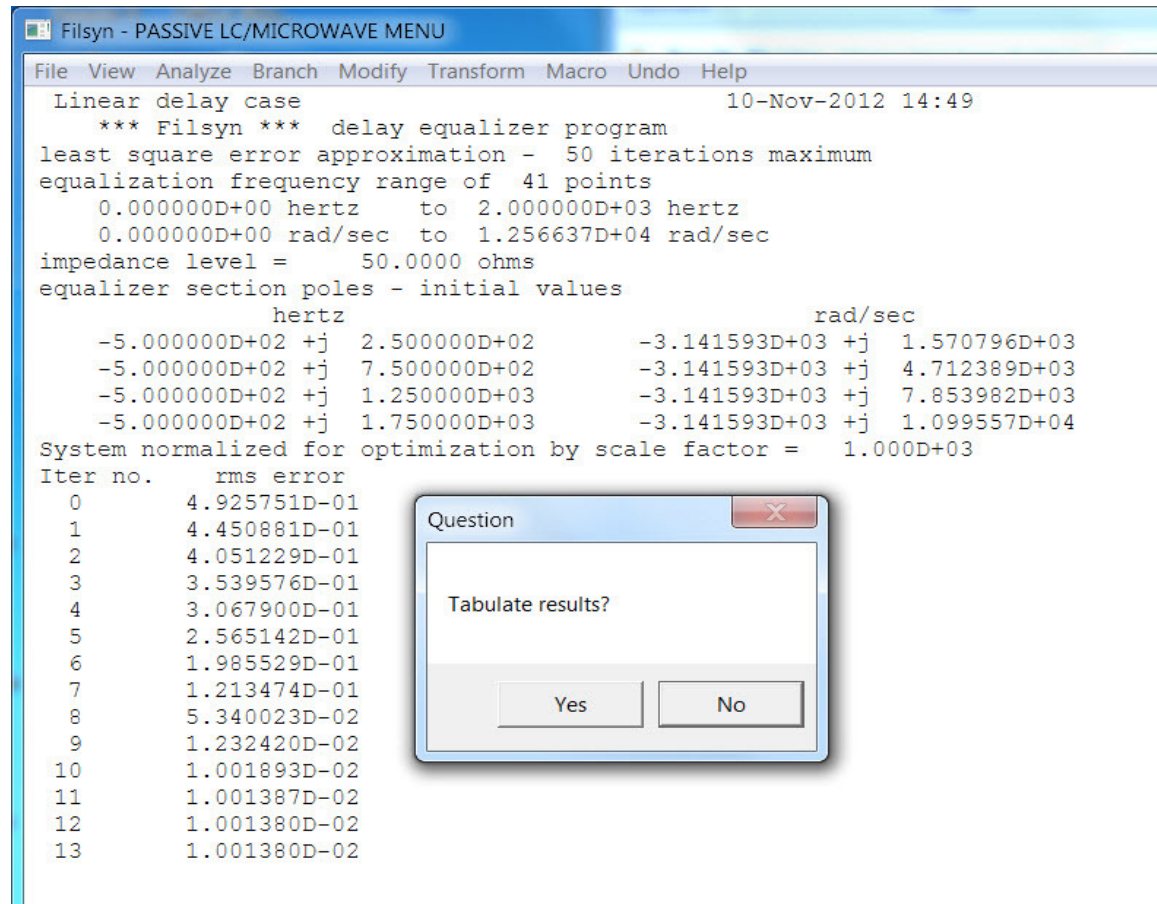


When we click the **Open** button, the program reads the data in. In fact, we can check this by clicking on the **Data** button again and we see the data already there:

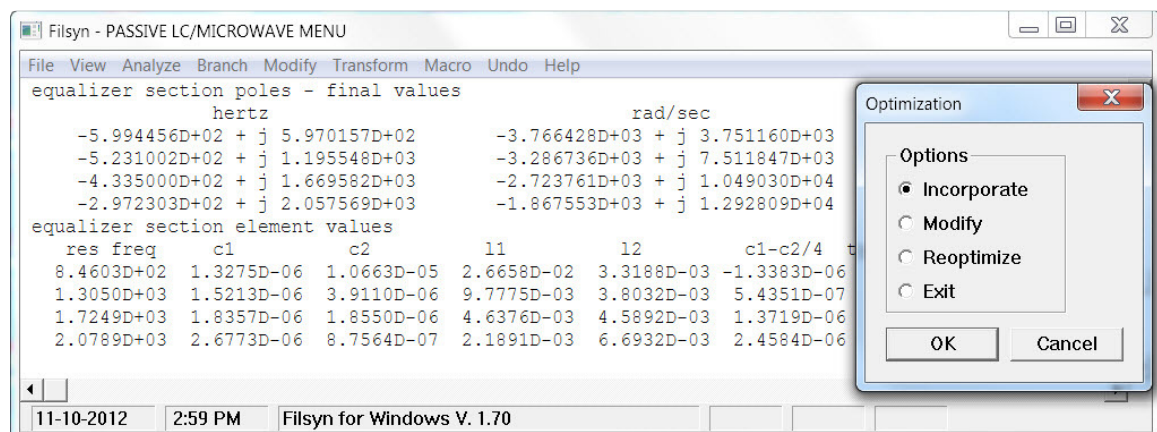


Note that now the only data necessary is the equalizer degree and the impedance level, if passive LC implementation is selected, as we do here.

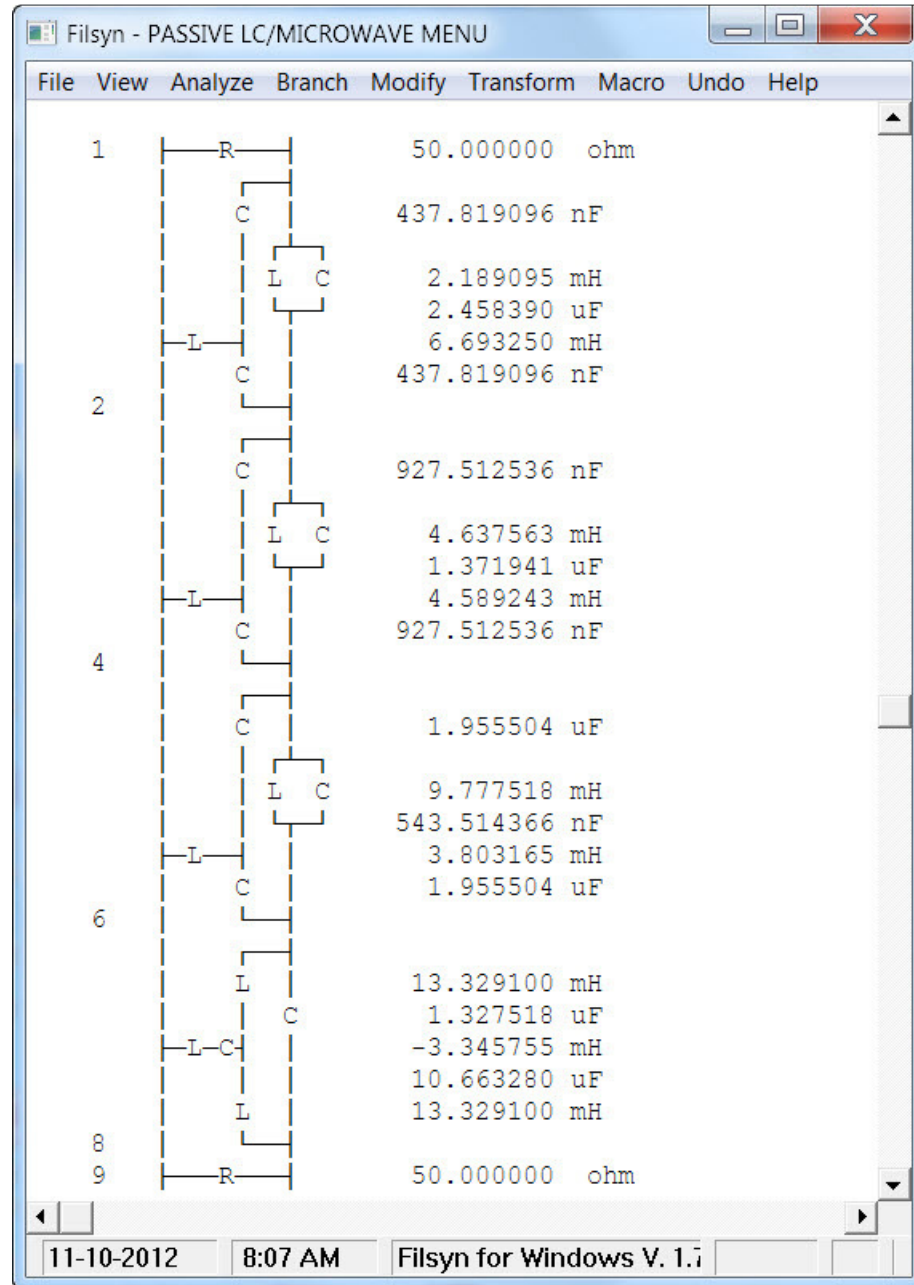
Clicking on the **OK** button, we immediately enter the delay equalizer segment that performs the optimization (see the appropriate part of chapter 4) we decline the tabulation of the complete set of data:



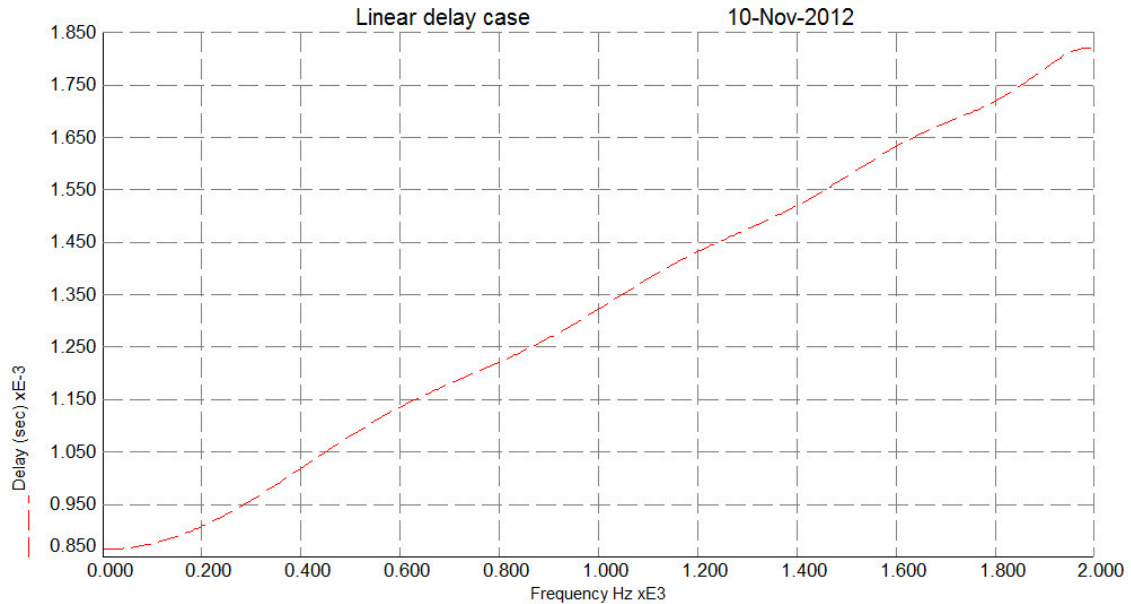
Next we see the final numerical results and are offered several options, our selection being the implementation, as shown below:



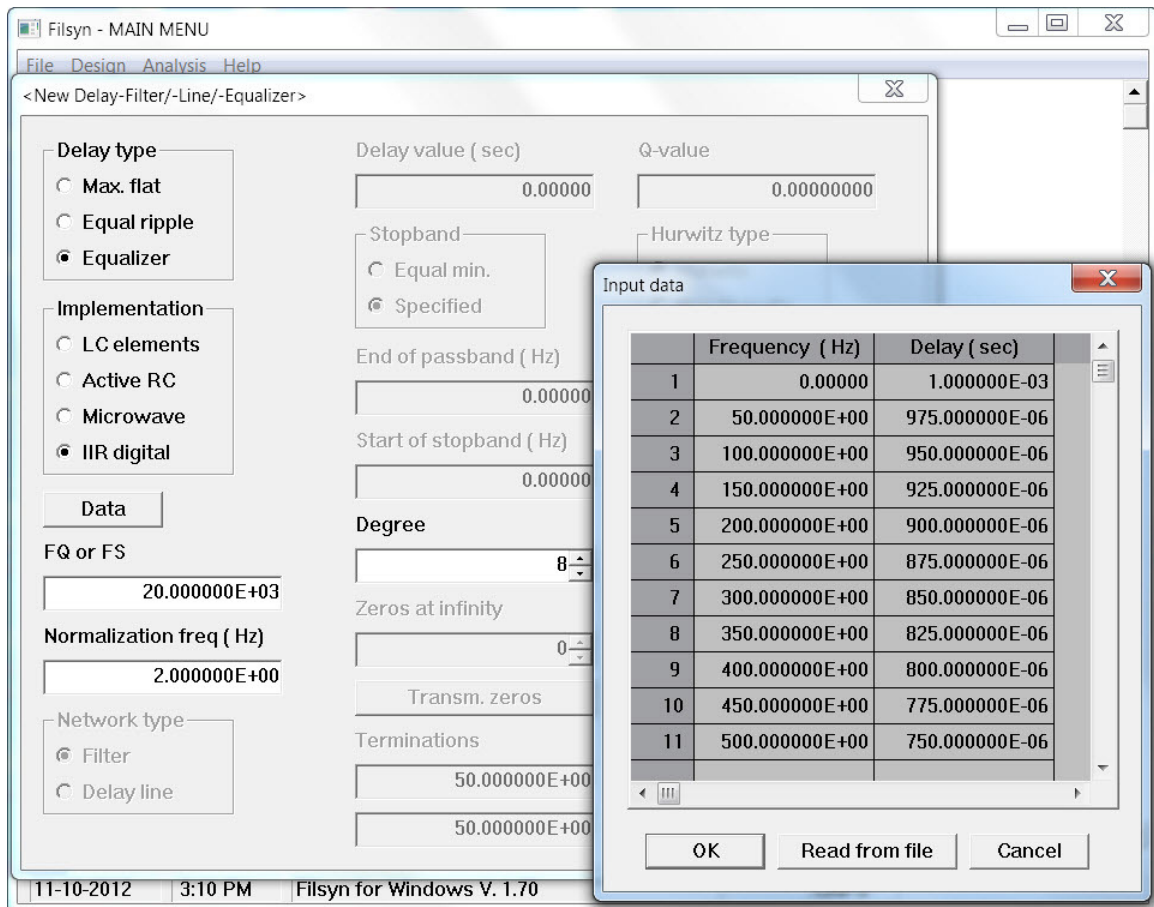
Yielding the equalizer implemented as four bridged-T sections:



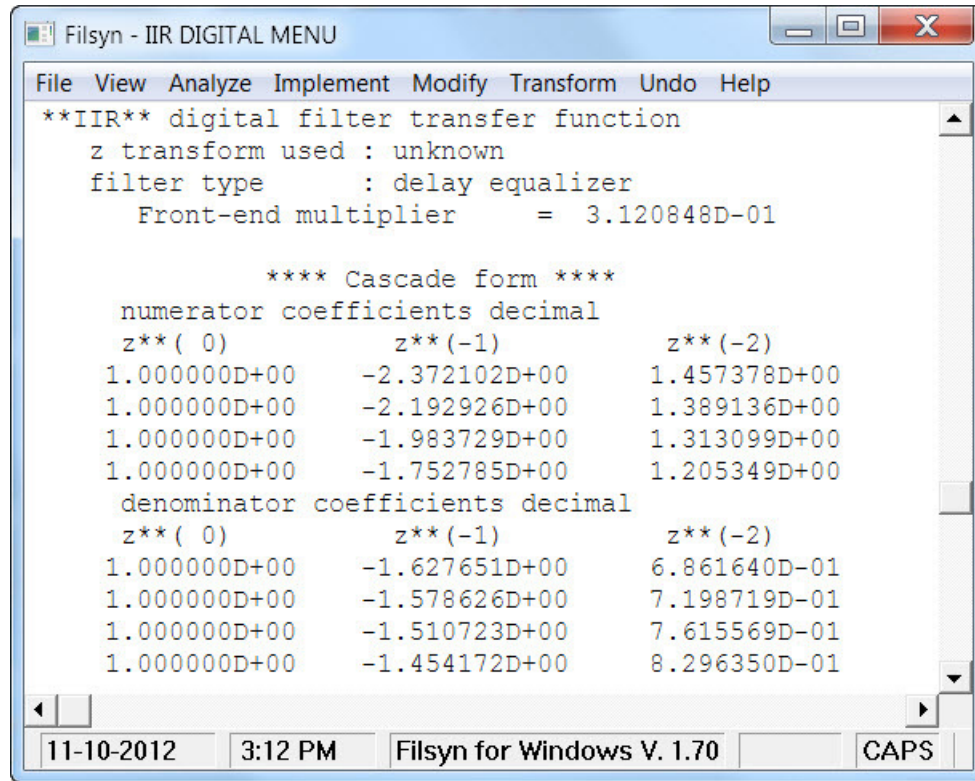
Since the overall degree was specified as 8, we have four second order equalizer sections here. Computing the performance of this network, we get what we expected, a very close-to-linear delay:



As another example we shall equalize the same delay with an IIR digital equalizer, that is driven by a 20 KHz sampling frequency and since the delay equalization band ends at 2 KHz, we specify a 2 KHz normalization frequency:



After optimizing the delay and declining the tabulation offer we selected the 8th order equalizer using the **Incorporate** option:



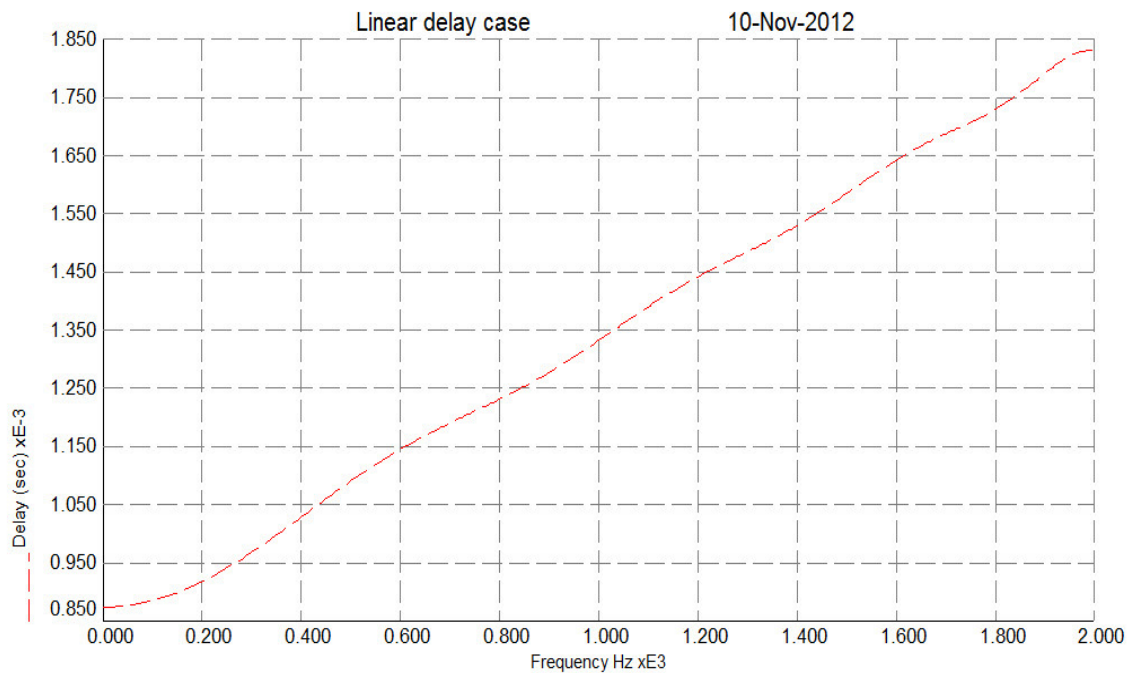
```

Filsyn - IIR DIGITAL MENU
File View Analyze Implement Modify Transform Undo Help
**IIR** digital filter transfer function
z transform used : unknown
filter type      : delay equalizer
Front-end multiplier = 3.120848D-01

**** Cascade form ****
numerator coefficients decimal
z**( 0)      z**(-1)      z**(-2)
1.000000D+00 -2.372102D+00  1.457378D+00
1.000000D+00 -2.192926D+00  1.389136D+00
1.000000D+00 -1.983729D+00  1.313099D+00
1.000000D+00 -1.752785D+00  1.205349D+00
denominator coefficients decimal
z**( 0)      z**(-1)      z**(-2)
1.000000D+00 -1.627651D+00  6.861640D-01
1.000000D+00 -1.578626D+00  7.198719D-01
1.000000D+00 -1.510723D+00  7.615569D-01
1.000000D+00 -1.454172D+00  8.296350D-01
11-10-2012  3:12 PM  Filsyn for Windows V. 1.70  CAPS

```

The computed delay is very similar to that of the other implementations:



Active RC or microwave equalizers may be designed just as easily.

Comment:

Specifying the natural modes of delay lines is not implemented in this segment, because it is available in the **Analysis** segment of the program. See chapter 11 (*Existing filters*) for details.

13. COMBLIN POSTPROCESSOR

13.1 Introduction

This stand-alone program takes its main input from a file generated by the **Filscript** -> **Filsyn** program pair. That file will contain the electrical design of a microwave filter that may be implemented in a comblin form and this postprocessor will then generate the dimensions of a possible physical implementation. This implementation may be either in the form of round rod resonators of equal lengths and equal diameters, or rectangular rods of equal lengths and equal thicknesses.

These rods may be tuned either with lumped capacitors or with distributed capacitors of shorter lengths. The program then prints the equivalent electrical circuit as well as the mechanical dimensions and placements of the resonator rods.

Because the conversion to physical dimensions is only approximate, the passband performance of the resulting structure will not be perfectly equal ripple any longer. This situation may be corrected by an additional touch-up optimization step as explained below.

13.2 Usage

We start by using the **Filscript** script processor and select the **MW Bandpass filters** -> **Comblin filter** option. Once we specify the filter, the **Filsyn** program will generate the filter. This design will then be saved automatically using the **File->Save** option and the program will terminate. Next this postprocessor **comb.exe** will be called and the **File->New design** option will select the file containing our saved filter. At this point control returned to the user and we need to enter some physical dimensions for the implementation. We then have the actual mechanical design which we can further analyze, optimize and otherwise refine.

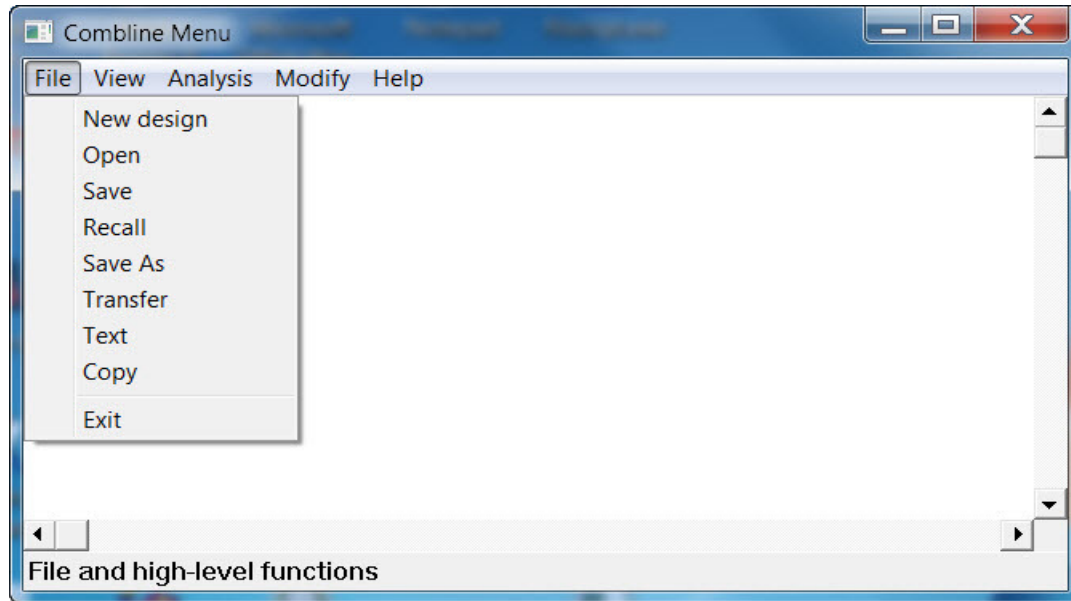
If the **Filscript.exe** script processor detects the presence of this **comb.exe** postprocessor, then all of the above steps leading up to the necessary mechanical dimensions, will be performed automatically by the script processor, as described above. The file used to save and transfer the data from **Filsyn** to **Comb** is called “_comb.qfp”. In case we wish to look at the final electrical form of the filter before proceeding to the mechanical design, we can call the **Filsyn** program again, select the **File->Recall** menu option and specify the **_comb.qfp** file to recall. This will then display the filter and we may analyze it if we wish.

Proceeding to the mechanical design the physical dimensions needed are as follows:

First we need to select either the metric system (mm) or the English one (inches). Next we must decide whether we will use lumped tuning capacitors or distributed ones. The lumped case is the default as the most often used. Next we need to specify the (common) diameter of the round resonator rods, the distances of the end rods from the enclosure walls (preferably the same at both ends) and finally the diameter of the coupling pins.

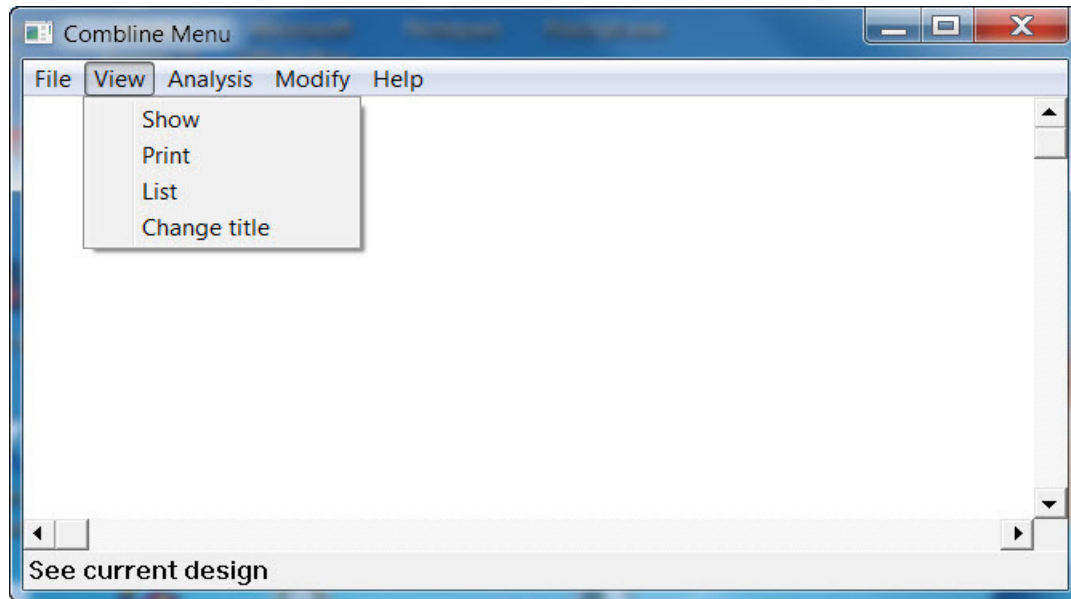
The program then generates the dimensions of the complete structure and its equivalent electrical circuit and prints them side by side.

At this stage, the full capability of the program is as follows:



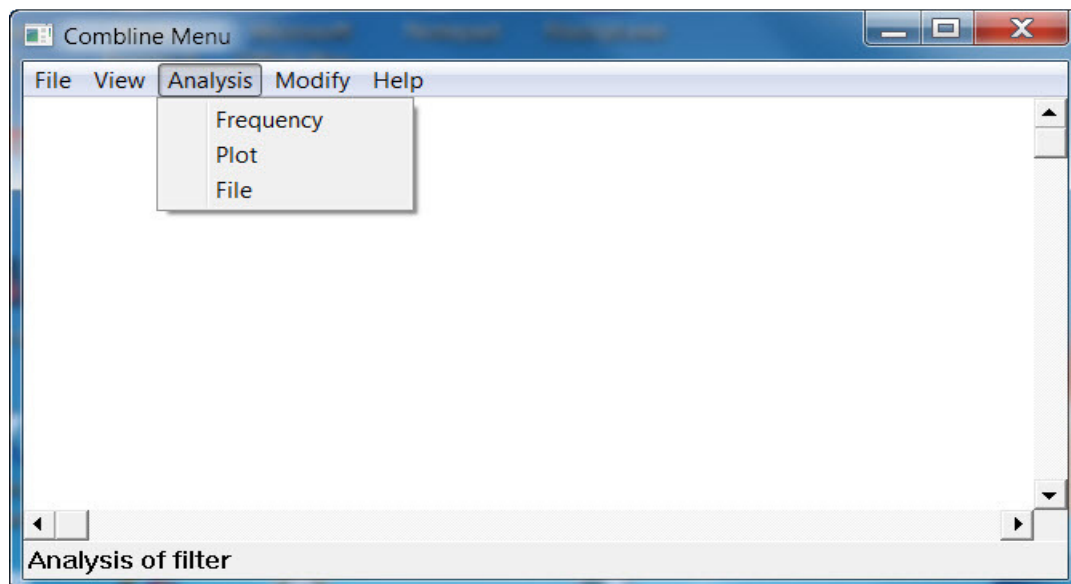
Under the **File** menu option we have the following submenu items:

1. **New design** This is used to start the design process by reading a stored filter in.
2. **Open** Opens any text file for observation (but does not edit).
3. **Save** This saves the current design in a disk file.
4. **Recall** Recalls the data stored in a file.
5. **Save as** Same as **Save** above but saves it in a new file.
6. **Transfer** Generates files to transfer data to the *Scompact* or *Touchstone* programs.
7. **Text** Writes the current design in a text file for documentation purposes.
8. **Copy** Copies the part of the window selected by the mouse to the clipboard.
9. **Exit** Terminates program execution.



Under the **View** menu option there are the following submenu items:

1. **Show** Displays the current design in the window.
2. **Print** Turns the print option on. This makes everything that shows up subsequently in the window also sent to the printer or a text file.
3. **List** For debugging purposes only.
4. **Change title** Just what it says.

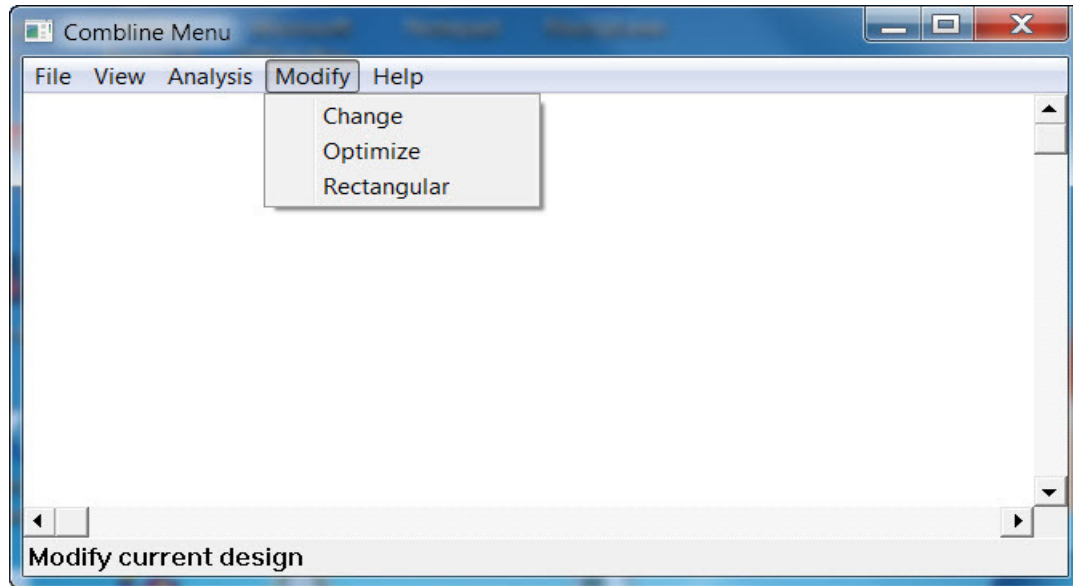


Next comes the **Analysis** menu option with submenu items:

1. **Frequency** Performs a standard frequency domain analysis on the equivalent electrical circuit.
2. **Plot** Plots any of the computed characteristics. The plot(s) may be

printed.

3. **File** Writes the analysis data to a text file for documentation purposes.



Under the **Modify** menu option we can:

1. **Change** Change the value of any of the electrical elements.
2. **Optimize** Perform a touch-up optimization. Changes both the physical dimensions and the corresponding electrical parameters.
3. **Rectangular** Converts the design from round rods to rectangular rods.

Finally the **Help** menu item contains the usual submenu items.

13.3 Example

Consider the 7 section comblne filter with the following specification. First we call the **Filscript** program and select the **MW Bandpass filters->Comblne filter** option. The corresponding data entry screen specifies a filter with a passband from 8 GHz to 10 GHz, an electrical length of 40° at the center frequency and a 0.05 dB passband ripple:

Combine microwave bandpass

Lower passband edge (GHz): 8.000000E+00

Upper passband edge (GHz): 10.000000E+00

Passband loss ripple (dB): 0.500000000

Electrical length in degrees: 40.0000000

Number of sections: 7

Capacitors: ☒ Lumped, ☐ Commensurate

Junction coupling: ☒ Inductive, ☐ Unit element

OK Cancel

Letting the program run, it completes its run in **Filsyn**, saves the design in a file called *_comb.qfp* and quits the **Filsyn** program. It then calls the **Comb.exe** postprocessor, and using the **File-> New design** menu option it reads the filter data in from the above storage file and shows the following display:

Combine input

Select unit: ☐ Metric (mm), ☒ English (in)

Convert to: ☒ Lumped C, ☐ Short stub C, ☐ Commensurate

Rod diameter: 0.940000000E-01

Gaps from end-rods to walls: 0.780000000E-01

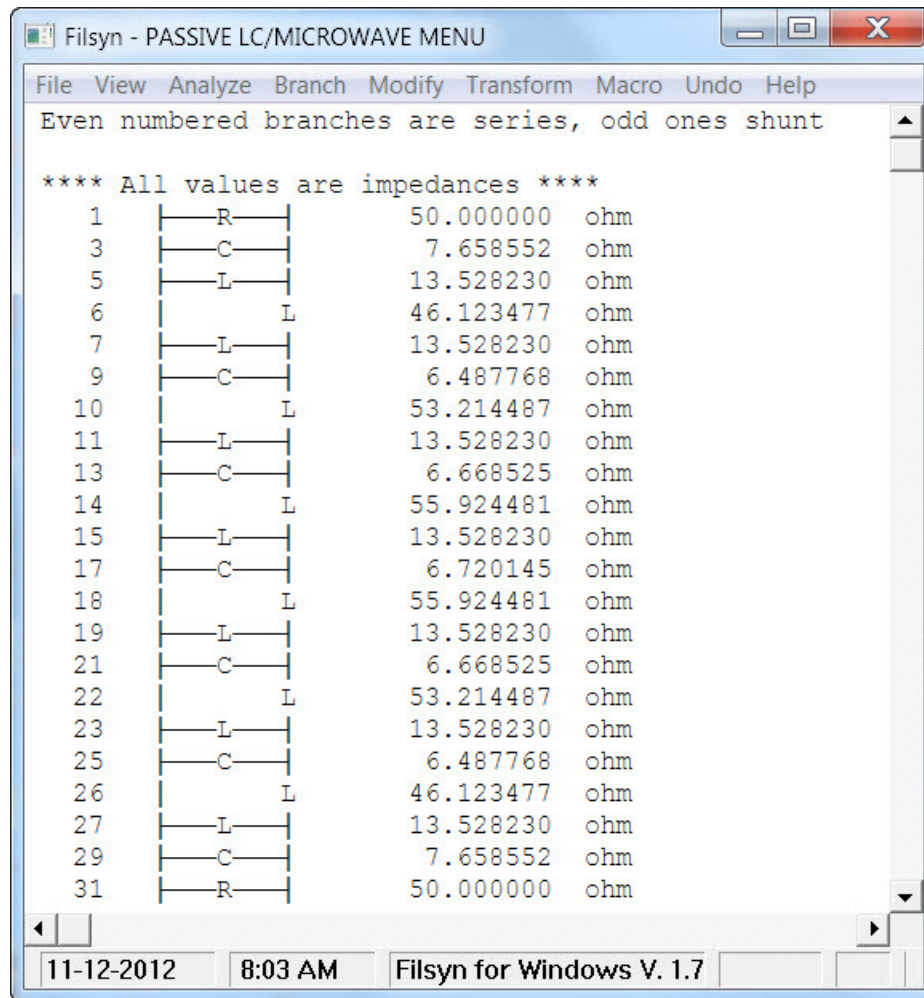
Coupling pin diameter: 0.500000000E-01

Ground space placing: 0.250000000

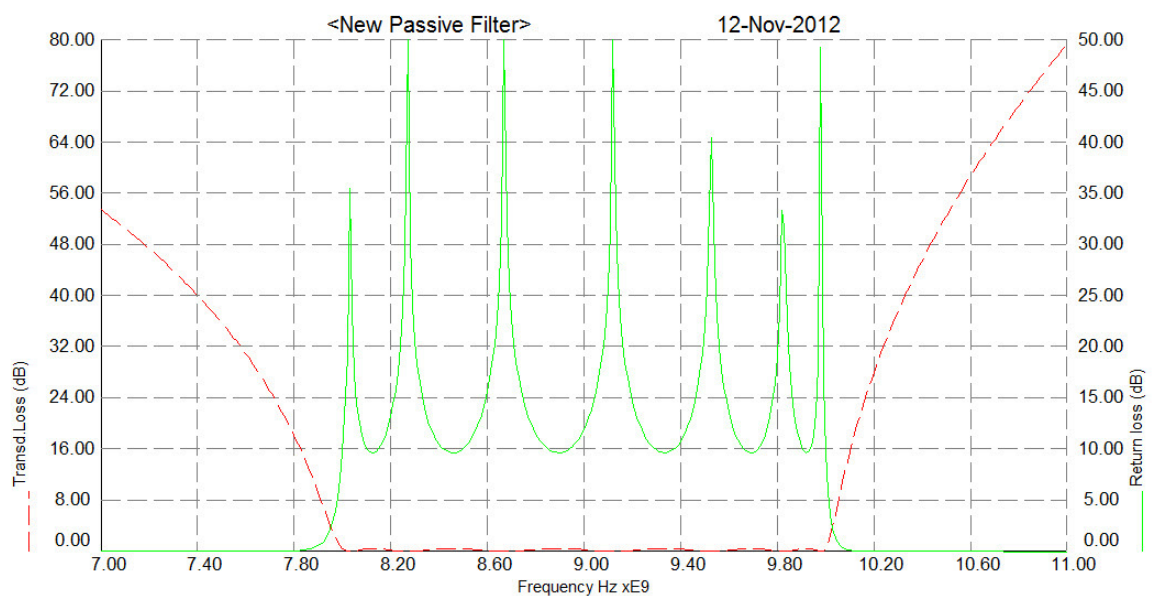
OK Cancel

Note at this point that in order to minimize displays we did not show the filter structure and element values as **Filsyn** has completed that part of the process. However if we wish we can look at that by starting **Filsyn** again and using the **File->Recall** menu item we can recall the *_comb.qfp* file and inspect the filter that is automatically displayed as shown:

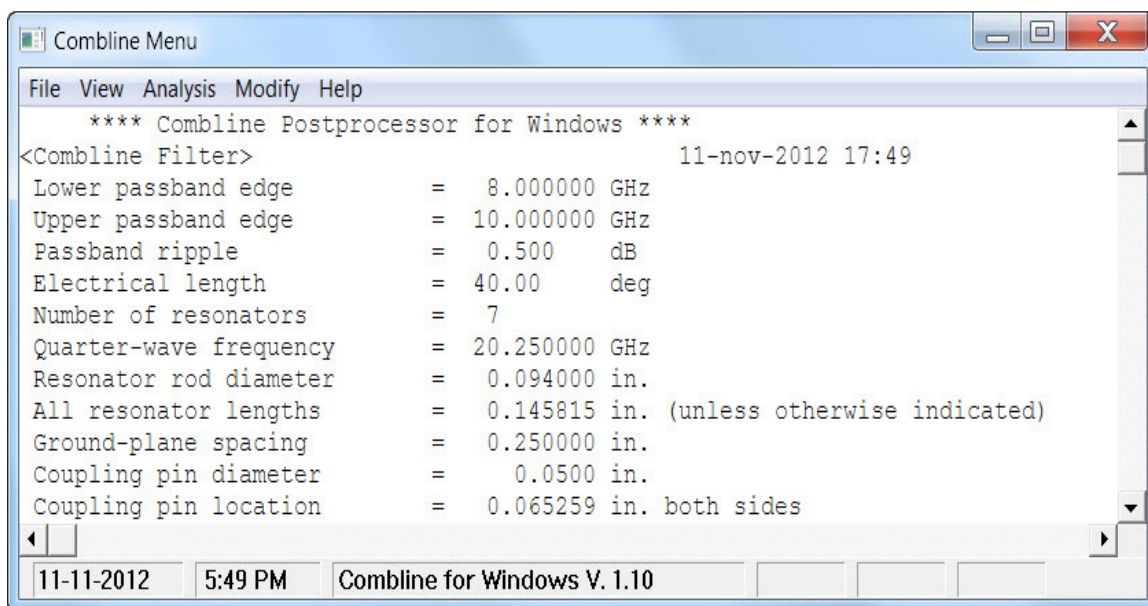
Comblne filters



The frequency analysis shows the perfect ideal response:



Going back to the **Comb** postprocessor shown above with the selected mechanical dimensions using English units and clicking the OK button, we immediately have the design summary:



followed by the circuit and dimensions:

Combine Menu					
File View Analysis Modify Help					
		Ohms	Capacitance in pF	Location	Length all in INCHES
1	—R—	5.00000D+01			
3	* * *				
	* U E *	7.65211D+01			0.0780 0.0780
	* * *				
5	—L—	7.54360D+01			0.0528
7	* * *				
	* U E *	7.54360D+01			0.0931
	* * *				
9	—L—	-5.42972D+02		0.1250	
11	—C—		3.17786D-01		
12	—L—	4.25133D+02			0.1034
13	—L—	1.17838D+02		0.3224	
15	—C—		2.57318D-01		
16	—L—	5.56577D+02			0.1272
17	—L—	1.13174D+02		0.5436	
19	—C—		2.53009D-01		
20	—L—	5.80145D+02			0.1310
21	—L—	1.12618D+02		0.7687	
23	—C—		2.52527D-01		
24	—L—	5.80145D+02			0.1310
25	—L—	1.13174D+02		0.9937	
27	—C—		2.53009D-01		
28	—L—	5.56577D+02			0.1272
29	—L—	1.17838D+02		1.2149	
31	—C—		2.57318D-01		
32	—L—	4.25133D+02			0.1034
33	—L—	-5.42972D+02		1.4123	
35	—C—		3.17786D-01		
37	* * *				
	* U E *	7.54360D+01			0.0931
	* * *				
39	—L—	7.54360D+01			0.0528
41	* * *				
	* U E *	7.65211D+01			0.0780 0.0780
	* * *				
43	—R—	5.00000D+01		1.5373	
11-11-2012 5:50 PM Comblne for Windows V. 1.10					

At this stage we have two things of concern. One is the fact that the electrical circuit on the left is an (approximate) equivalent of the physical structure on the right. We have no way to ascertain how accurate that equivalence is, because that needs the detailed analysis of the electromagnetic field inside the filter. However past experience shows the representation to be quite accurate.

The other question is: how close is the performance of this electrical circuit to the original filter we started out with? This we can answer by performing a frequency domain analysis on this electrical circuit, using lossless elements:

<Combine Filter>

Start and end freqs in GHz

7.000000E+00

11.000000E+00

Increment:

0.000000

or no. of frequencies:

401

Loss in dB/wavelength

0.00000000

Type of loss

☒ Const./length

☐ Const./wavelength

Capacitor Q:

0.00000000

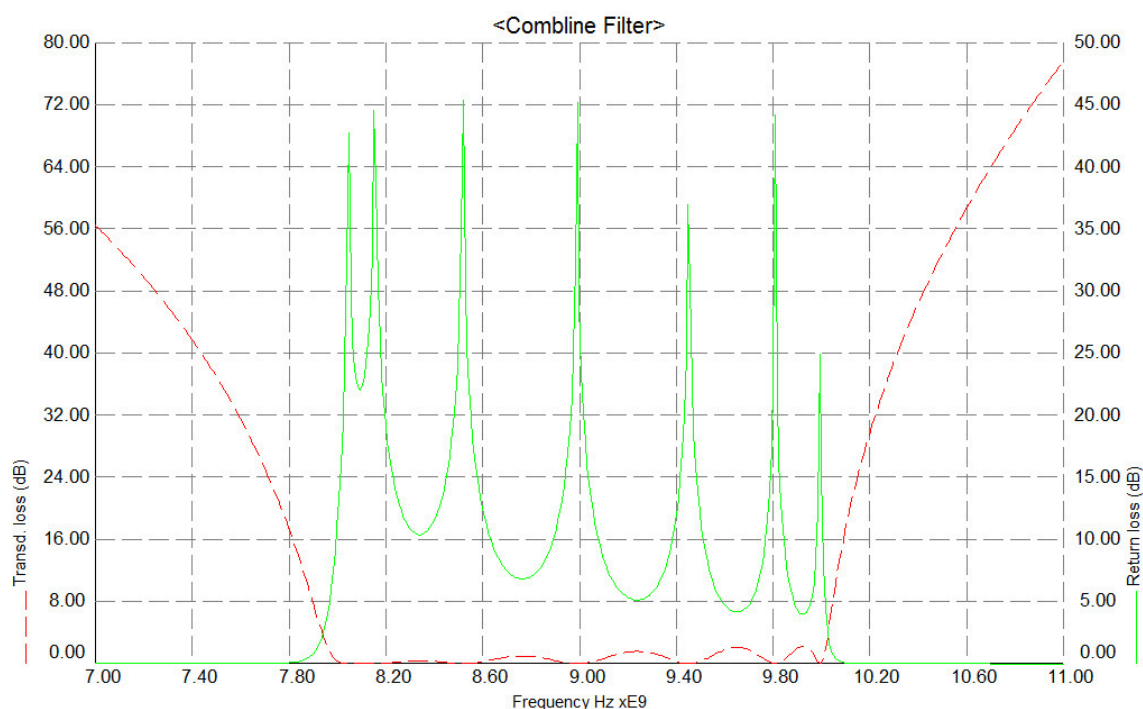
Type of Q

☒ Const. Q

☐ Const. dissipation

OK Cancel

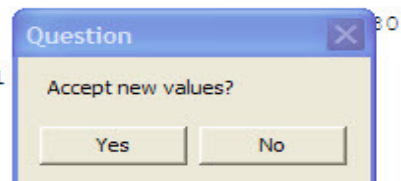
The results show a decent filter but not a very nice passband:



This is where the touch-up optimization under the **Modify->Optimize** menu option comes in. Invoking that, we get the next part display:

Combine filters

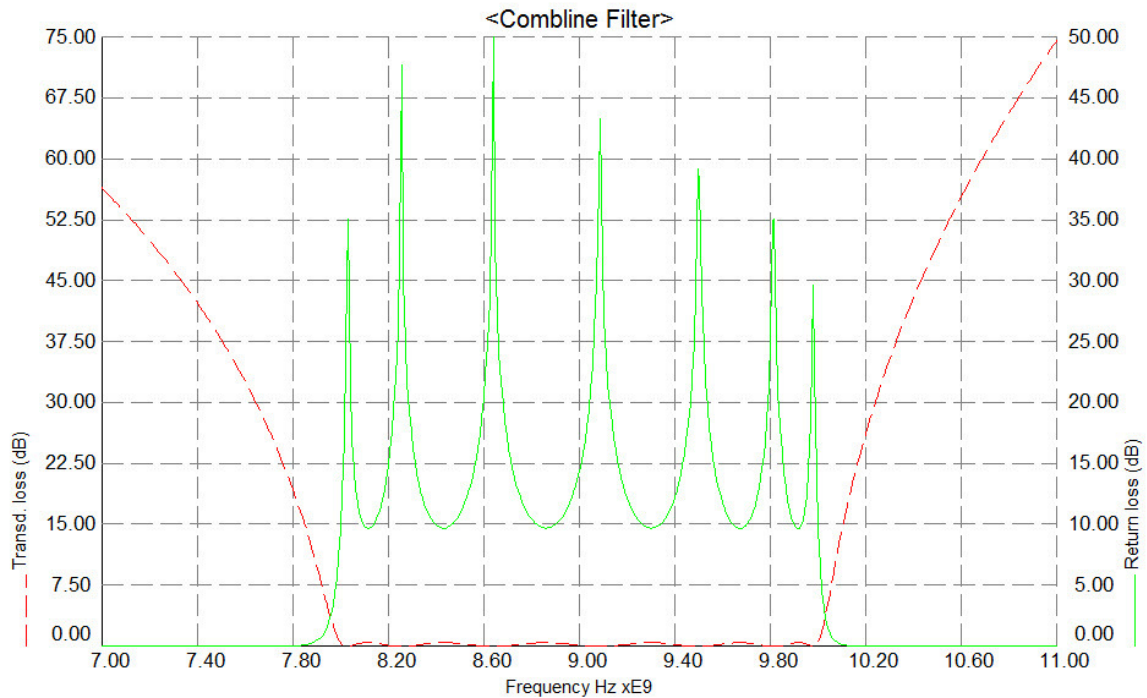
28		L	5.06336D+02			0.1187
29		L	1.23033D+02	1.1693		
31		C		2.62894D-01		
32		L	3.45435D+02			0.0861
33		L	-4.68468D+02	1.3494		
35		C		3.29254D-01		
37	*	U E *	7.52207D+01		0.0768	
39		L	7.52207D+01		0.0690	
41	*	U E *	7.63106D+01			
43		R	5.00000D+01	1		
Iteration 1 Step size = 1.00000						
Iteration 2 Step size = 1.00000						
Iteration 3 Step size = 1.00000						



Apparently the optimization needed only three rounds and if we accept the new values that will update both the mechanical dimensions as well as the values in the equivalent electrical circuit. The new values are only slightly different, since the design was fairly close to ideal to start with:

Combine Menu						
File View Analysis Modify Help						
		Ohms	Capacitance in pF	Location	Length all in INCHES	Gap
1		R	5.00000D+01			
3	*	U E *	7.72193D+01		0.0780	0.0780
5		L	7.54270D+01		0.0573	
7	*	U E *	7.54270D+01		0.0885	
9		L	-5.40044D+02	0.1250		
11		C		3.07826D-01		
12		L	4.22057D+02			0.1028
13		L	1.17986D+02	0.3218		
15		C		2.57816D-01		
16		L	5.55576D+02			0.1271
17		L	1.13206D+02	0.5429		
19		C		2.53114D-01		
20		L	5.79839D+02			0.1310
21		L	1.12633D+02	0.7678		
23		C		2.52588D-01		
24		L	5.79839D+02			0.1310
25		L	1.13206D+02	0.9928		
27		C		2.53114D-01		
28		L	5.55576D+02			0.1271
29		L	1.17986D+02	1.2139		
31		C		2.57816D-01		
32		L	4.22057D+02			0.1028
33		L	-5.40044D+02	1.4107		
35		C		3.07826D-01		
37	*	U E *	7.54270D+01		0.0885	
39		L	7.54270D+01		0.0573	
41	*	U E *	7.72193D+01		0.0780	0.0780
43		R	5.00000D+01	1.5357		

Analyzing the new design again, we get a very close to ideal performance already:



At this stage we may repeat the optimization (clearly unnecessary) and would get a performance indistinguishable from the ideal.

Instead, we call the **Modify->Rectangular** option to see what the corresponding physical implementation would be:

Combine Menu			
File View Analysis Modify Help			
<Combine Filter> 11-nov-2012 17:49			
**** Rectangular resonator dimensions ****			
All resonator lengths =		0.14581 inches	
All thicknesses =		0.07952 inches	
Coupling pin location =		0.04485 inches	
Resonator No.	Width	Gap	Leading edge
		All in inches	
		0.07800	
1	0.07549	0.11315	0.07800
2	0.08070	0.13782	0.26664
3	0.07975	0.14173	0.48516
4	0.07966	0.14173	0.70665
5	0.07975	0.13782	0.92804
6	0.08070	0.11315	1.14561
7	0.07549	0.07800	1.33947
Overall length =		1.49296 inches	
11-11-2012 5:59 PM Combine for Windows V. 1.10			

As a final result, we show the generated netfile to transfer the electrical equivalent circuit to *Scompact*:

```

* <Comblne Filter>
F1 : 2.0249999E+10
EL : 90.00
BLK
* RES 1 0 R = 5.0000000E+01
  TRL 2 1 Z = 7.6310616E+01 E = 48.14 F=F1
  SST 2 0 Z = 7.5220741E+01 E = 42.57 F=F1
  TRL 3 2 Z = 7.5220741E+01 E = 47.43 F=F1
  SST 3 0 Z = -4.6846780E+02 E=EL F=F1
  CAP 3 0 C = 3.2925390E-13
  SST 4 3 Z = 3.4543509E+02 E=EL F=F1
  SST 4 0 Z = 1.2303271E+02 E=EL F=F1
  CAP 4 0 C = 2.6289376E-13
  SST 5 4 Z = 5.0633600E+02 E=EL F=F1
  SST 5 0 Z = 1.1538541E+02 E=EL F=F1
  CAP 5 0 C = 2.5513107E-13
  SST 6 5 Z = 5.4501019E+02 E=EL F=F1
  SST 6 0 Z = 1.1432079E+02 E=EL F=F1
  CAP 6 0 C = 2.5416545E-13
  SST 7 6 Z = 5.4501019E+02 E=EL F=F1
  SST 7 0 Z = 1.1538541E+02 E=EL F=F1
  CAP 7 0 C = 2.5513107E-13
  SST 8 7 Z = 5.0633600E+02 E=EL F=F1
  SST 8 0 Z = 1.2303271E+02 E=EL F=F1
  CAP 8 0 C = 2.6289376E-13
  SST 9 8 Z = 3.4543509E+02 E=EL F=F1
  SST 9 0 Z = -4.6846780E+02 E=EL F=F1
  CAP 9 0 C = 3.2925390E-13
  TRL 10 9 Z = 7.5220741E+01 E = 47.43 F=F1
  SST 10 0 Z = 7.5220741E+01 E = 42.57 F=F1
  TRL 11 10 Z = 7.6310616E+01 E = 48.14 F=F1
* RES 11 0 R = 5.0000000E+01
A: 2POR 1 11
  END
  FREQ
* PUT FREQUENCIES HERE
  STEP *F1 F2 DF*
  END
*
  OUT
  PRI A R1 = 5.0000000E+01 R2 = 5.0000000E+01 S
  END

```

The text file generated for documentation purposes is too large for inclusion here.

14. EDGE-COUPLED MW FILTER POSTPROCESSOR

14.1 Introduction

This stand-alone program takes its main input from a file generated by the **Filscript** -> **Filsyn** program pair. That file will contain the electrical design of a microwave filter that may be implemented in an edge-coupled stripline or microstrip form and this postprocessor will then generate the dimensions of a possible physical implementation.

First note that the design is based on the following equivalence:

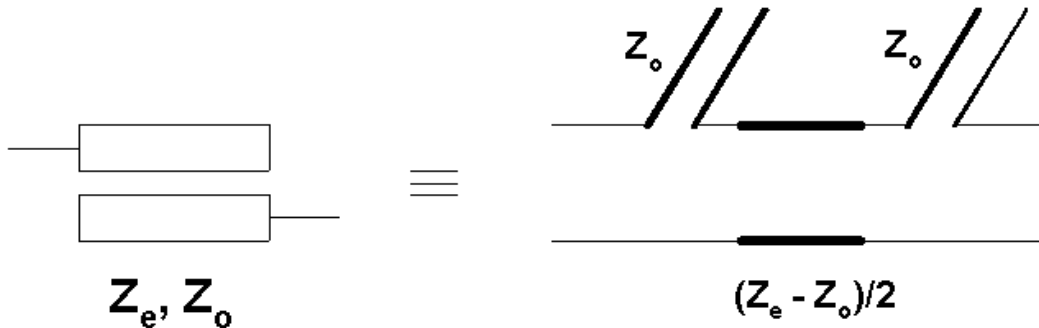


Fig. 21 Equivalence

In order to use this equivalence, the starting electrical design must contain a sequence of sections that look like the right side of this equivalence. The starting point is therefore a sequence of unit elements with a single series open stub at one end. Then a series of transformations need to be applied to this circuit to distribute the series open stub across the circuit such that the end stubs have an impedance exactly half of the ones in the middle. This is facilitated by a special form of the **Modify**->**Norton**->**All** menu option that is the **End values halved** submenu item. All this is initially performed by the script file that is built into the **Filscript.exe** program under the **MW bandpass filters**->**Edge-coupled microwave filters** menu item. This postprocessor picks up that design and proceeds to convert these electrical design data to physical dimensions.

Because the conversion to physical dimensions is only approximate, the passband performance of the resulting structure will not be perfectly equal ripple any longer.

14.2 Usage

We start by using the **Filscript** script processor and select the **MW Bandpass filters** -> **Edge-coupled microwave filter** option. Once we specify the filter, the **Filsyn** program will generate the electrical filter circuit. The script program then saves this design using the **File**->**Save** option and exits. Next **Filscript** calls this postprocessor **Ecm.exe** and uses the **File**->**New design** option and selects the file containing our saved filter. At this point we need to enter some physical dimensions for the implementation, after which we have the actual mechanical design, which we can analyze.

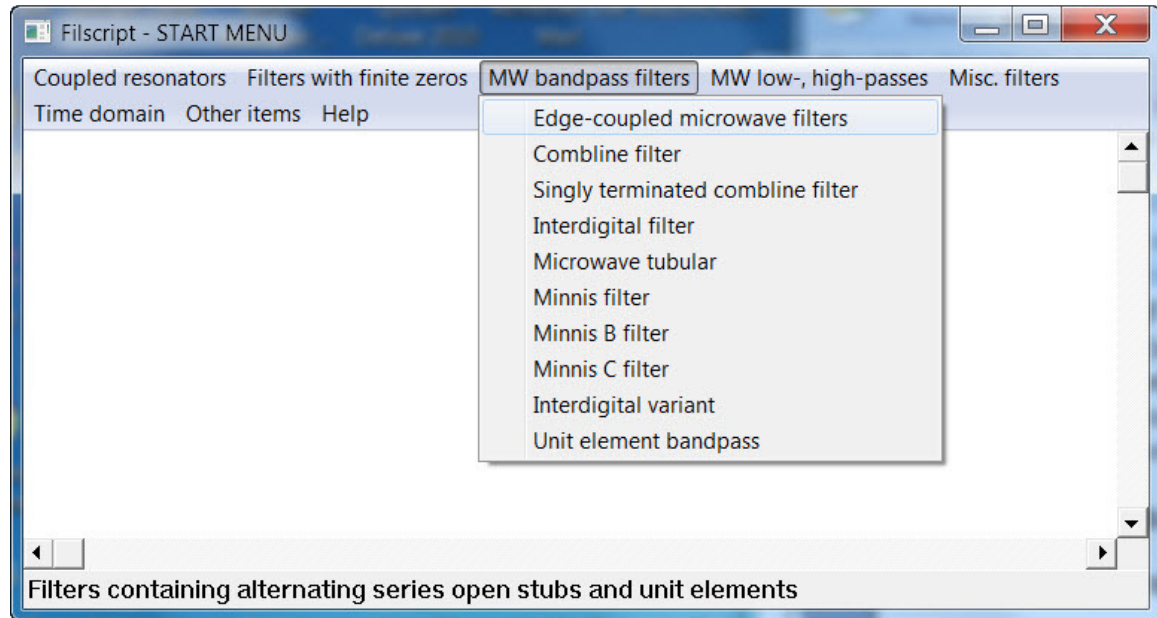
If the script processor detects the presence of this *ecm.exe* postprocessor, then all of the above steps leading up to the necessary mechanical dimensions, will be performed automatically by the **Filscript.exe** script processor. The file name used to save the data in is “_ecm.qfp”.

The initial physical dimensions needed are as follows:

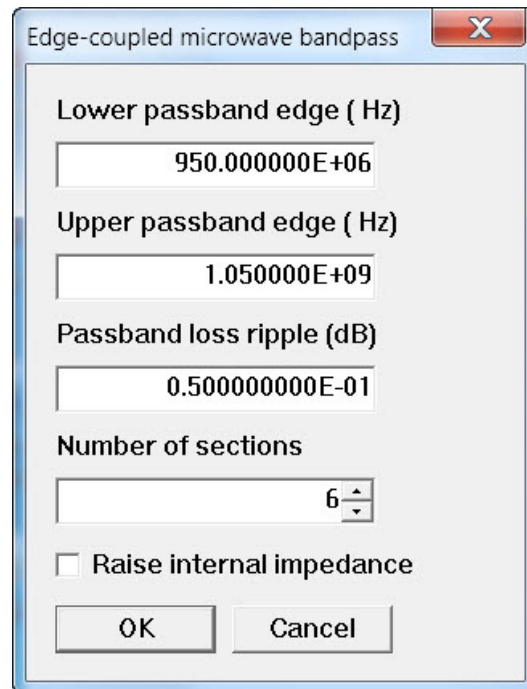
First we need to decide whether we will use the metric system (mm) or the English one (mil). Next we must decide whether we wish to implement the filter as a microstrip or a stripline structure. Finally we have to specify either the substrate height in the microstrip case or the groundplane spacing in the stripline one, as well as the relative dielectric constant of the insulating material.

14.3 Example

We start out by calling the **Filscript** script processor and select the proper option:

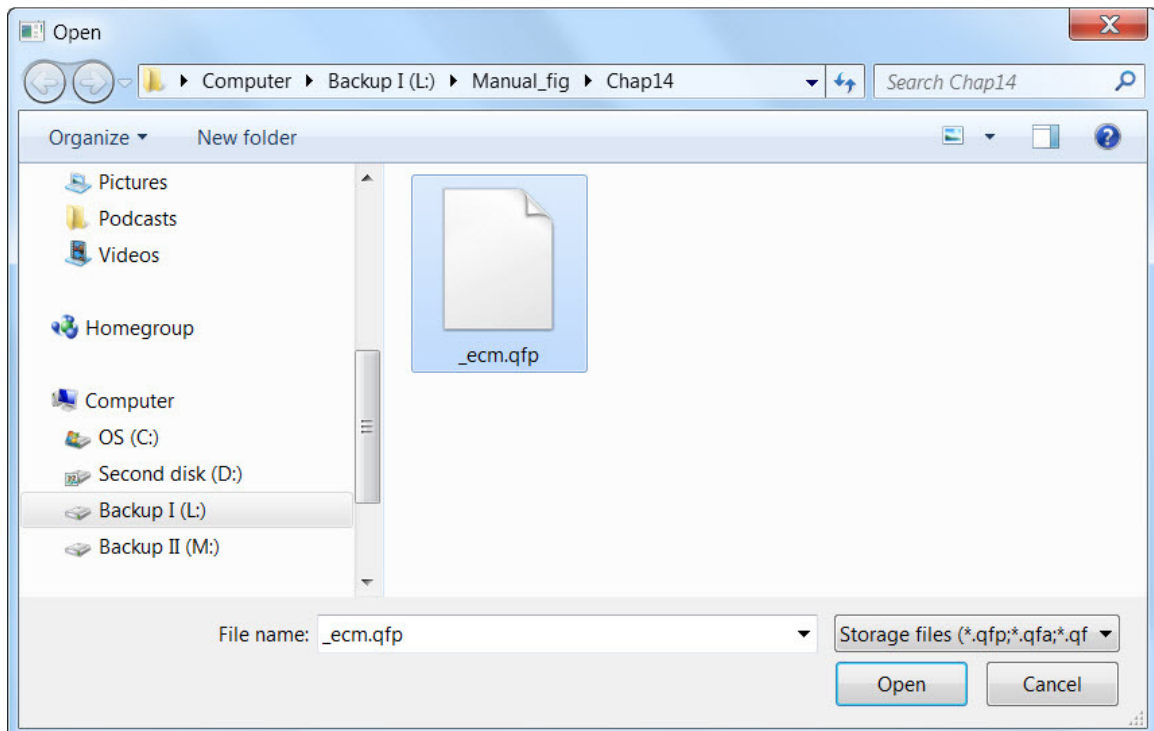


The filter will be a six resonator structure of 10% bandwidth centered at 1 GHz as shown:

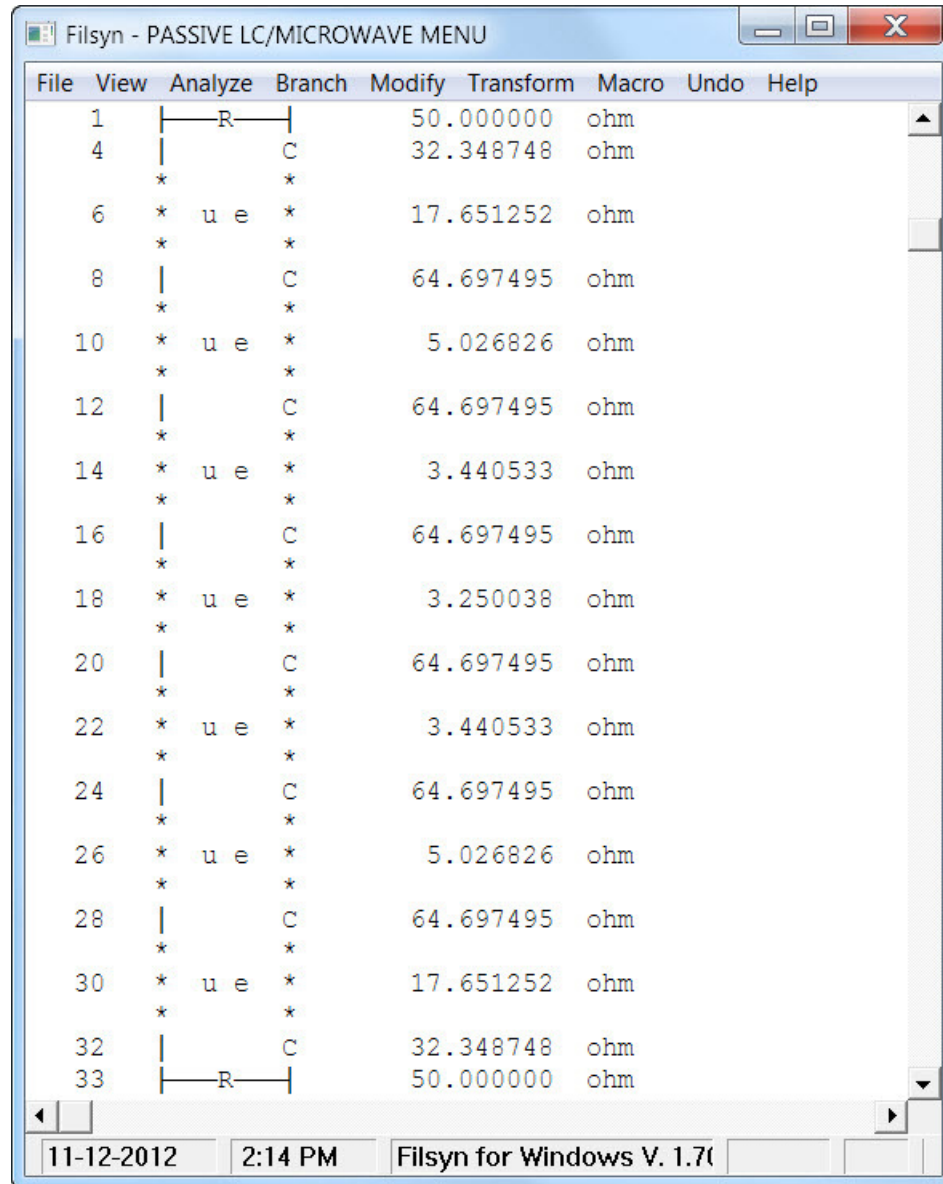


The processor calls the **Filsyn** main program, transfers the data and when it stops, we have the electrical design completed and at this point we enter the **Ecm** postprocessor.

Before proceeding, we demonstrate how to display the filter that is transferred, i.e. the one in *_ecm.qfp*. We call **Filsyn** and select the **File->Recall** menu item and specify this file:



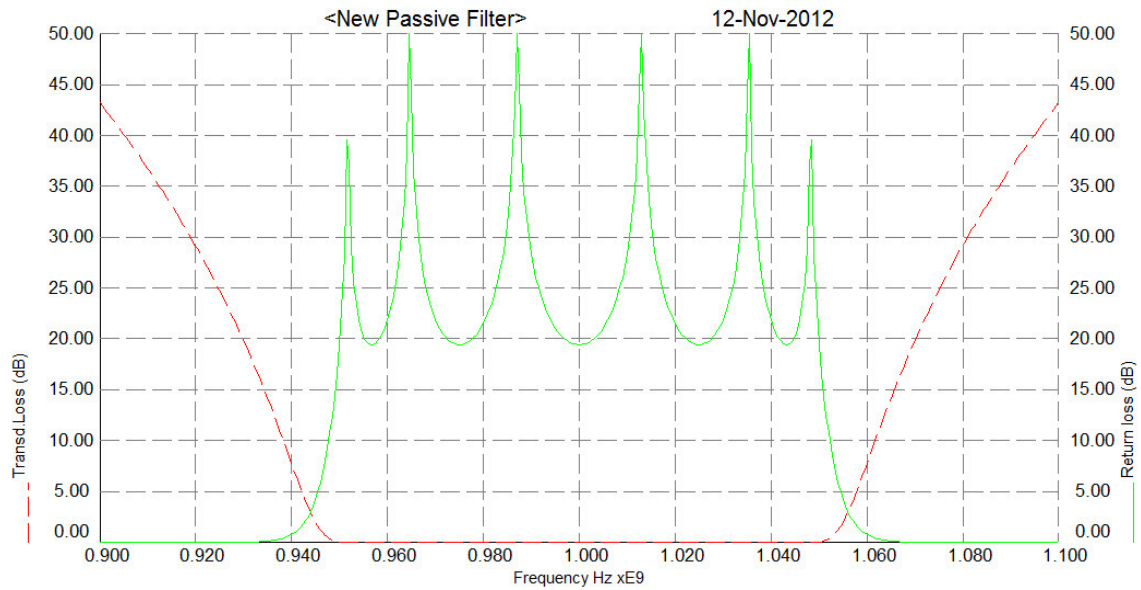
The result is the display of the filter data:



File	View	Analyze	Branch	Modify	Transform	Macro	Undo	Help
1		R		50.000000	ohm			
4			C	32.348748	ohm			
	*		*					
6	*	u e	*	17.651252	ohm			
	*		*					
8			C	64.697495	ohm			
	*		*					
10	*	u e	*	5.026826	ohm			
	*		*					
12			C	64.697495	ohm			
	*		*					
14	*	u e	*	3.440533	ohm			
	*		*					
16			C	64.697495	ohm			
	*		*					
18	*	u e	*	3.250038	ohm			
	*		*					
20			C	64.697495	ohm			
	*		*					
22	*	u e	*	3.440533	ohm			
	*		*					
24			C	64.697495	ohm			
	*		*					
26	*	u e	*	5.026826	ohm			
	*		*					
28			C	64.697495	ohm			
	*		*					
30	*	u e	*	17.651252	ohm			
	*		*					
32			C	32.348748	ohm			
33		R		50.000000	ohm			

11-12-2012 2:14 PM Filsyn for Windows V. 1.70

We can see that all series capacitors are equal, except the end ones that have half the value. Analyzing this filter we have a perfect response:

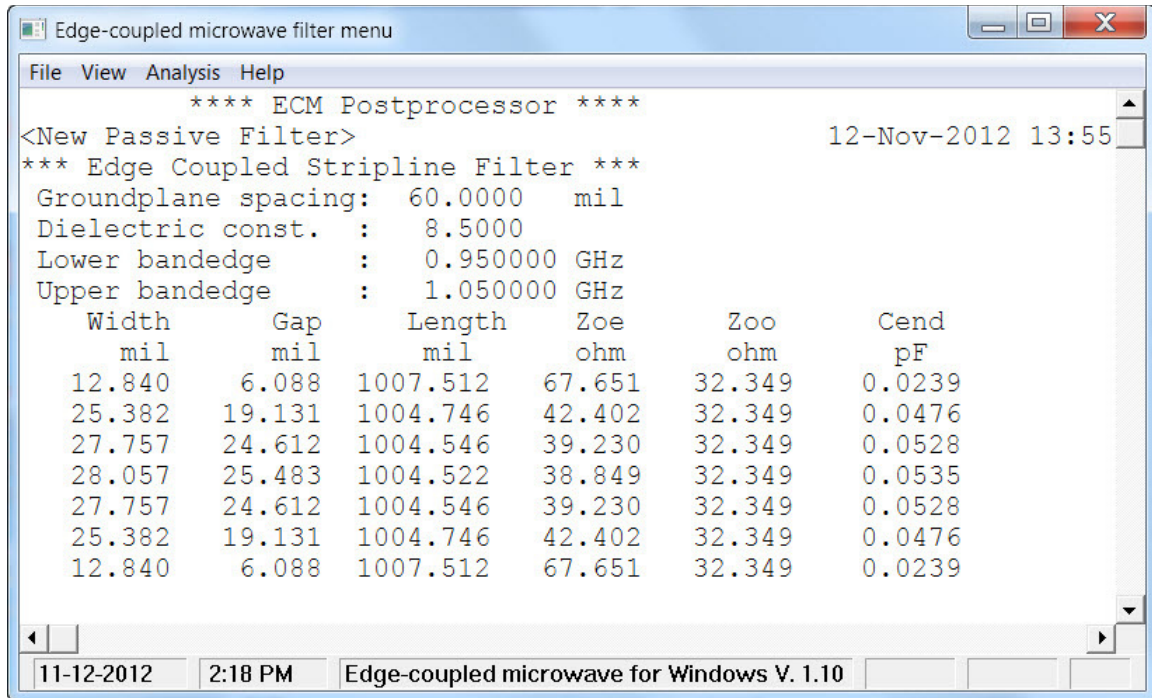


Returning to the postprocessor we enter the data:

The ECM input dialog box contains the following settings:

- Select unit:**
 - ☒ Metric (mm)
 - ☐ English (mil)
- Implementation:**
 - ☐ Microstrip
 - ☒ Stripline
- Substrate height:** (Label only)
- Ground plane spacing:** 60.0000000
- Relative dielectric const.:** 8.5
- Buttons:** OK, Cancel

Here we selected English units and specified ground plane spacing of 60 mils and a relative dielectric constant of 8.5. The results follow immediately:



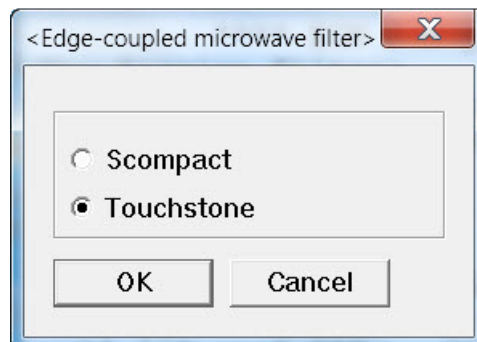
We have the following menu options at this point:

1) The **File** menu option has a number of sub options as shown above, including:

1. **New design**
2. **Open**
3. **Transfer**
4. **Text**
5. **Copy**
6. **Exit**

Most of these are familiar from the main **Filsyn** program, where they have the same functions. **New design** is for starting a new filter. **Open** is to open a text file for displaying it in a window, but not for editing it. **Transfer** is to transfer the data to another program, *Scompact* or *Touchstone* in this case. **Text** is to generate a text file for documentation and **Copy** is to copy a selected part of the output to the clipboard. Finally **Exit** is to terminate the program.

To demonstrate, let us generate a transfer file to the *Touchstone* program:



The file generated is listed here:

```

! EDGE COUPLED STRIPLINE FILTER
!   SYNTHESIZED BY FILSYN
DIM
  FREQ GHZ
  LNG MIL
  CAP PF

CKT
  SSUB ER= 8.50 B= 60.00 T=0.0 RHO=1.0
  SCLIN 1 2 3 4 W= 12.8 S= 6.1 L=1007.5
  SCLIN 3 5 6 7 W= 25.4 S= 19.1 L=1004.7
  SCLIN 6 8 9 10 W= 27.8 S= 24.6 L=1004.5
  SCLIN 9 11 12 13 W= 28.1 S= 25.5 L=1004.5
  SCLIN 12 14 15 16 W= 27.8 S= 24.6 L=1004.5
  SCLIN 15 17 18 19 W= 25.4 S= 19.1 L=1004.7
  SCLIN 18 20 21 22 W= 12.8 S= 6.1 L=1007.5
  CAP 2 0 C=.02393
  CAP 4 0 C=.02393
  CAP 5 0 C=.04759
  CAP 7 0 C=.04759
  CAP 8 0 C=.05284
  CAP 10 0 C=.05284
  CAP 11 0 C=.05353
  CAP 13 0 C=.05353
  CAP 14 0 C=.05284
  CAP 16 0 C=.05284
  CAP 17 0 C=.04759
  CAP 19 0 C=.04759
  CAP 20 0 C=.02393
  CAP 22 0 C=.02393
  DEF2P 1 21 FILTER

OUT
  FILTER DB[S21] GR1
  FILTER DB[S11] GR1

FREQ
  SWEEP F1 F2 F3

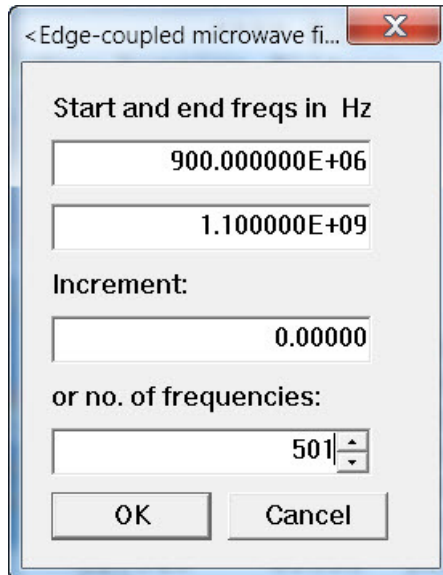
GRID
  RANGE
  GR1 -40 0 10

```

2) Under the **View** menu options we have the submenu items of **Show**, **Print** and **Change title**, again familiar from the other programs.

3) Under the **Analysis** option, we again have the **Frequency**, **Plot** and **File** submenu options, having the same role as in the other programs. The same is true of the submenu options of the **Help** menu item.

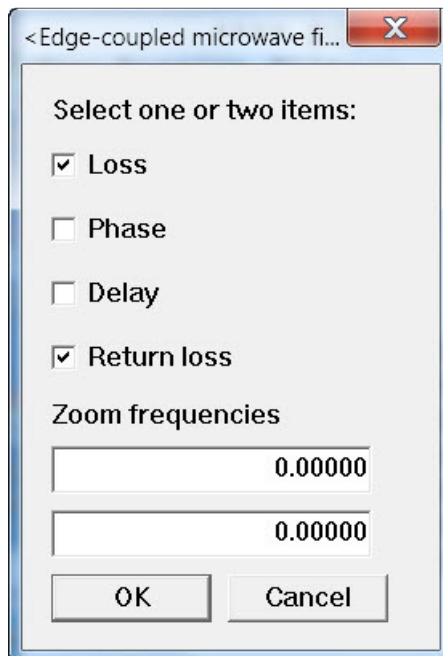
As the last step, let us analyze the performance of this filter. Calling the **Analysis->Frequency** option, we specify:



The dialog box titled "<Edge-coupled microwave fi..." contains the following fields and controls:

- Start and end freqs in Hz:**
 - Start frequency:
 - End frequency:
- Increment:**
 -
- or no. of frequencies:**
 - with up/down arrow buttons
- Buttons:** OK, Cancel

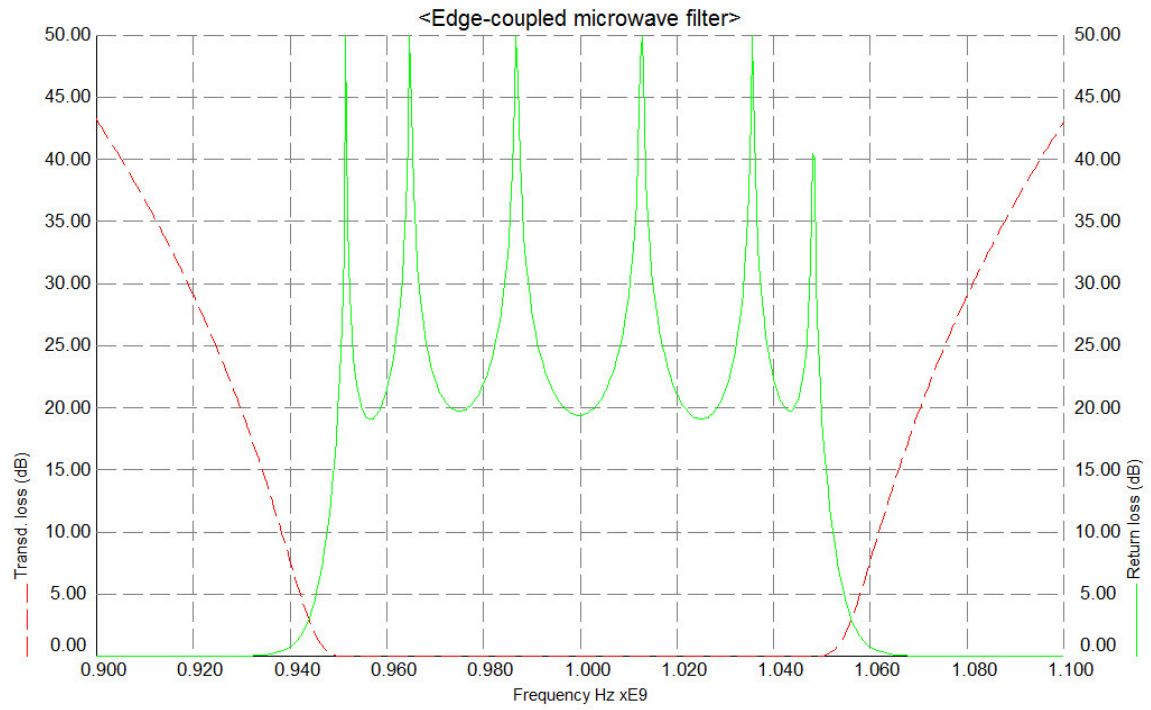
Declining the listing of the results, we opt for the plot of the loss and return loss:



The dialog box titled "<Edge-coupled microwave fi..." contains the following fields and controls:

- Select one or two items:**
 - ☒ Loss
 - ☐ Phase
 - ☐ Delay
 - ☒ Return loss
- Zoom frequencies:**
 -
 -
- Buttons:** OK, Cancel

The result is:



This is surprisingly close to the performance of the original electrical design.

15. OPTIMIZING PREPROCESSOR

15.1. Introduction

While the **Filsyn** program has always offered the most powerful capabilities of any known filter synthesis program, there has been one feature missing: the ability to design filters with pass bands of prescribed shape. The direct specification of filters in **Filsyn** was restricted to the flat passband option (either maximally-flat or equal-ripple), with the special exception of sloping passband for bandpass filters.

The functional input option alleviated this problem somewhat, but it was still necessary to find the poles and zeros of a transfer function with the correct loss and/or delay shape, before we could use **Filsyn** to synthesize the circuit.

Our **Optimizing preprocessor** has been developed to perform this function, thereby removing one of the few remaining gaps in our capabilities. It is able to adjust the poles and zeros of a transfer function to meet a given set of loss (and possibly delay) requirements.

In deciding to use the transfer function singularities as optimization parameters — as opposed to circuit component values — we have been guided by several considerations. One was the capability of **Filsyn** to go from transfer functions to circuit configurations and element values with ease. Another consideration was the existence of other programs with the capability of adjusting circuit element values to match required performance characteristics. The overwhelming reason however, was the difficulty we experienced in achieving optimal results using a fixed circuit configuration and working with the component values. We found that working with the transfer function parameters offers better optimization methods, leads to smoother and much faster convergence and better optima. We have also avoided the problem of how to select an initial circuit configuration that would be satisfactory for the purpose at hand, postponing this decision until after the procedure converged.

15.2 Data input

This program needs two sets of data, which it will read from disk files. The first contains the requirements in tabular form. This must be a standard text file that has the following structure:

A title line. This may be followed by any number of blank lines or lines starting with an “!”, followed by a line that contains the following keywords, separated by spaces and/or commas, in this order:

FREQ LOSS LW [DELAY DW]

Here LW stands for *Loss Weight*, DW for *Delay Weight* and the last two keywords may be left out simultaneously, if the only requirements are for the loss.

The next line should contain items indicating the units used. For example:

KHz dB usec

which refer to the frequency, loss and delay columns. Acceptable frequency and delay units are Hz, KHz, MHz, GHz, sec, msec, usec, nsec and psec respectively. The LW and DW columns need no unit indicators; these columns will contain numbers only. Actually, the dB item is also superfluous, since the loss will always be entered in dB.

After another blank line as a separator, we can have up to 501 data lines, each containing as many numbers as there are column keywords above. There should be no numbers missing. If a line is to contain, for example, a delay requirement only with no loss requirement, we still have to enter an item in the loss column and a zero in the loss weight (LW) column to indicate to the program, that the loss should be ignored at this frequency.

The actual numbers can be in free format, with or without decimal points, D or E exponents, and separated by spaces or commas or their combination. Blank lines or comment lines (beginning with an exclamation point) can also be used to make the file more readable. Comments may also be put on data lines following the actual data. Weights are positive, usually 1.0 or greater, except for zero values. As an example, here is the top section of a data file that we have been using to test the program:

This is the title line.

FREQ kHz	LOSS dB	LW	DELAY msec	DW
0.1	-0.007	1.	2.	1.
0.2	-0.029	1.	2.	1.
0.3	-0.065	1.	2.	1.
0.4	-0.116	1.	2.	1.
0.5	-0.182	1.	2.	1.
0.6	-0.264	1.	2.	1.
0.7	-0.361	1.	2.	1.
0.8	-0.474	1.	2.	1.
0.9	-0.603	1.	2.	1.
1.0	-0.750	1.	2.	1.
1.05	-0.830	1.	2.	1.
1.10	-0.914	1.	2.	1.
1.15	-1.003	1.	2.	1.
1.20	-1.097	1.	2.	1.
1.25	-1.197	1.	2.	1.
1.30	-1.301	1.	2.	1.
1.35	-1.410	1.	2.	1.
1.40	-1.525	1.	2.	1.
1.45	-1.645	1.	2.	1.
1.50	-1.772	2.	2.	1.
1.55	-1.904	2.	2.	2.
1.60	-2.043	5.	2.	2.
1.65	-2.188	5.	2.	2.
1.70	-2.341	5.	2.	2.
1.75	-2.500	5.	2.	2.
1.80	-2.668	5.	2.	2.

```

2.2      40.0      10.      1.      0.0
2.3      40.0      10.      1.      0.0
2.4      40.0      10.      1.      0.0
2.5      40.0      10.      1.      0.0
2.6      40.0      5.       1.      0.0
2.7      40.0      5.       1.      0.0
2.8      40.0      2.       1.      0.0
2.9      40.0      2.       1.      0.0
3.0      40.0      2.       1.      0.0
. . .

```

This file can have any name, but with an extension of “.req” (for *requirements*).

The other required file is the one that contains the starting point of the approximation, i.e. the initial values of the poles and zeros of the transfer function. This is also a simple text file and its structure needs no detailed description, because **Filsyn** writes it by itself. As described later, our best initial approximation will most likely come from **Filsyn** and can be generated when the filter was designed, using the **File->Pole-zero** menu option. This file will automatically have a “.pz” extension.

As an example, here is the file **Filsyn** has written from a 7th order elliptic lowpass with 0.2 dB passband ripple up to 2 kHz. The stop band begins at 2.2 kHz and yields about 42 dB stopband suppression.

```

! Pole-zero data of:
! <New Passive Filter>                                28-Jan-2010 11:33
! Poles (in rad/sec):
-6.359084810E+03    0.000000000E+00
-4.059931126E+03    8.478242531E+03
-1.448897681E+03    1.182648729E+04
-3.207255958E+02    1.271248589E+04
! Zeros (in Hz):
0.000000000E+00    3.749543617E+03
0.000000000E+00    2.468962106E+03
0.000000000E+00    2.221826042E+03
! Transfer function polynomials (normalized)
! Numerator      Denominator
2.606862218E-01    2.606862218E-01
0.000000000E+00    8.960365450E-01
0.000000000E+00    1.858642087E+00
0.000000000E+00    3.040641518E+00
0.000000000E+00    3.044181691E+00
0.000000000E+00    3.145878093E+00
0.000000000E+00    1.433842290E+00
0.000000000E+00    1.000000000E+00

```

This file is called “15_lowpass.pz”. We are only interested here in the poles and zeros; the actual file contains additional data but that will be ignored.

The exclamation point character signals the program to ignore everything that follows it. Therefore when ! is the first symbol, the whole line is ignored. Incidentally, the data is given in rad/sec units for the poles, while the zeros are specified in Hz, which is what we need (except in the case of digital or microwave filters, where we need normalized values for the poles). The reason for this particular structure is that this is the same structure that is needed to feed the pole-zero information back into **Filsyn** for the synthesis step.

Note that complex conjugate pairs need to be entered only once, and even complex zero quadruplets are entered as one pair. For the circuit to be stable, all the poles must have negative real parts. During the optimization process the numbers representing the zeros will most likely remain stable, but sometimes some of the real parts of the poles may become positive, representing a nonrealizable result. We will point out ways of dealing with this situation later.

All other data needed for the operation of this preprocessor are simple decisions and a few numerical data items.

15.3 Initial value selection

One of the most important features of any iterative optimization procedure is its sensitivity to the initial values. The closer the initial values are to the final ones, the faster the procedure is going to converge and the more likely it will yield the best possible solution.

We have found that the best initial approximation is the one we can obtain from **Filsyn** in a manner that all stopband requirements are met. If there are delay requirements as well, we can simply add complex zeros and twice the number of complex poles to such a transfer function. The initial values of these additional complex quadruplets of zeros can be obtained by delay equalizing the initial filter and looking at the zeros these delay sections would add to the overall function. The corresponding poles can be obtained by picking two pairs for each complex zero, one on either side of the zeros. Do *not* select identical pairs of poles, optimization routines cannot handle multiple singularities.

We will illustrate the procedure by examples later in this chapter.

15.4 Program output

We will see later, that the program keeps us informed of the progress of the optimization and will yield tabulated or graphical display of results on demand, so that we can make an intelligent decision as to the next step. Once we decide to accept the final results, we can write the data into another data file in the same format with the same extension, which **Filsyn** will be able to read back.

15.5 Definition of error

In order to understand the operation of the program, we must say a few words about error definition. We use, by and large, the standard sum of squared and weighted errors over the whole frequency range. In the case of the stopband, i.e. where the requirements are greater than 20 dB, if the loss is more than the value required, the error is set to zero.

For the delay characteristics, we do two additional things. First, we subtract an additional flat delay from the requirements. This value is initially determined by the program, but later it is offered as an additional parameter to the user. Second, we use an additional overall multiplier for the delay error to enable us to control the relative weighting of the

loss and the delay. This also permits us to ignore the delay, even if the requirements list it, by setting this overall weight to zero. Both of these parameters are useful in running the program, but their values can best be determined during the run by observing the results.

15.6 Optimization strategies

The program uses three optimization strategies that the user can freely use in any sequence, any number of times. Before describing these strategies, we must point out that due to the error definition used, no exact gradient information is available, and we are restricted to the use of algorithms, that either do not need the gradient, or approximate it by themselves.

a) The first is the *simplex* method (ref. [42]) that is fast, starts from any starting point and is recommended for use at the start of any optimization. It is limited to 100 iterations, but can be restarted for additional iterations. It usually will stop after the 100 iterations with a comment stating that it did not converge, but will show a sizable improvement in the error. In order to limit the amount of printout, we print the error once every 10 iterations.

b) The second method is *Powell's* method (ref. [43]) and is another one that does not need derivative information. This one is somewhat less reliable; sometimes the error worsens instead of improving, but it can be used effectively in the middle of an iterative optimization run. The number of iterations is limited to 50, but we will usually see 6 to 12 iterations, hence the results are printed after each iteration.

c) The third and last method is a gradient-based method that approximates the derivatives (ref. [44]). This is a reliable method, sometimes a bit slow, but always improves the results. The number of iterations is limited to 50 and results are printed after every 5 iterations.

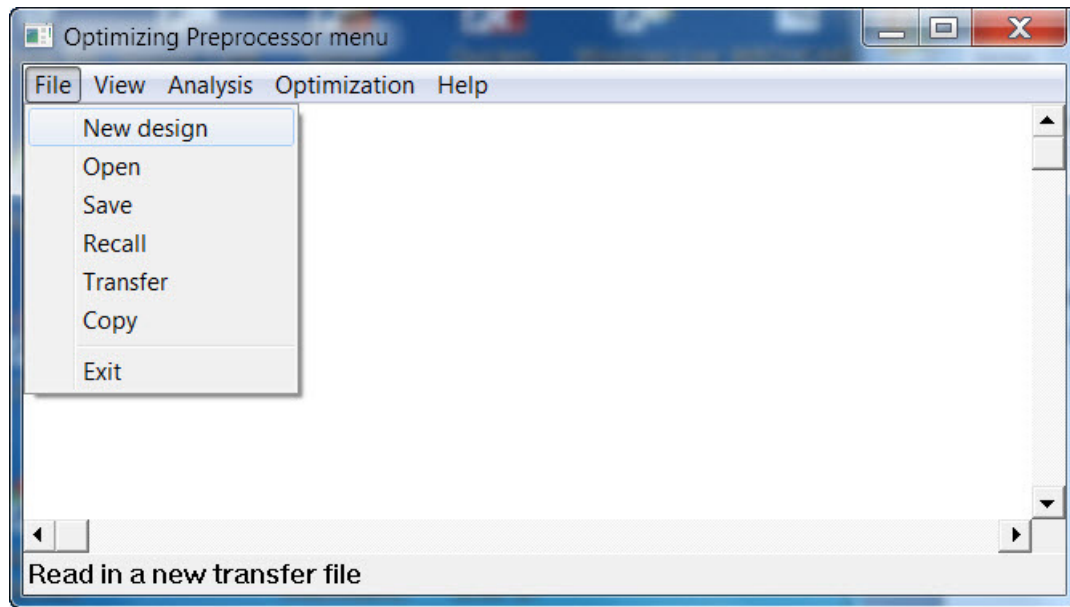
There are only a few guidelines as to the usage of these methods. One is to start any problem with two to three calls of method one and the other is to finish it with method 2 or 3. In between you may use any of them in any sequence any number of times.

15.7 Examples

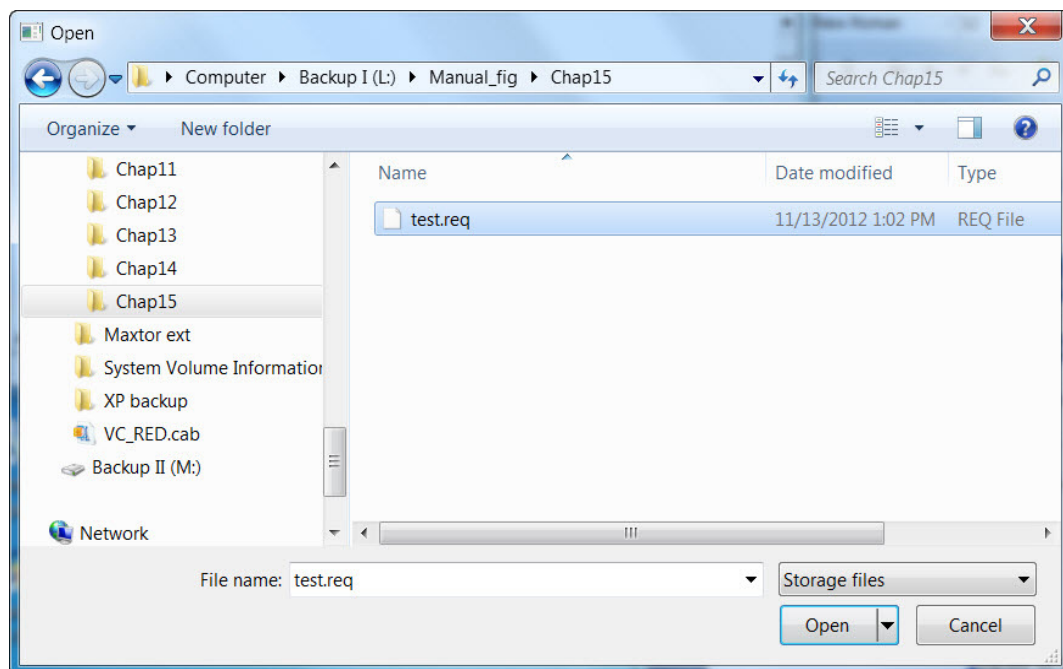
15.7.1 Example 1

As usual, the simplest way to introduce the program to new users is by way of a few examples. We will be using the requirement file illustrated above, which incidentally, specifies a loss shape that compensates for the $(\sin x)/x$ rolloff of a sample-and-hold circuit.

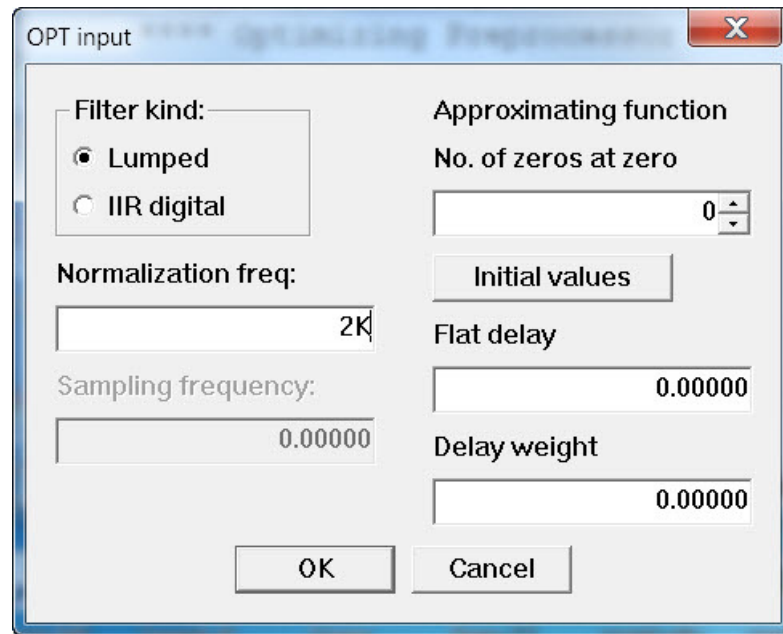
Calling the program is simple and we start by reading the file “test.req” which contains the requirements:



New design and **Exit** are self-explanatory; we shall explain the uses of the other menu items seen here later.



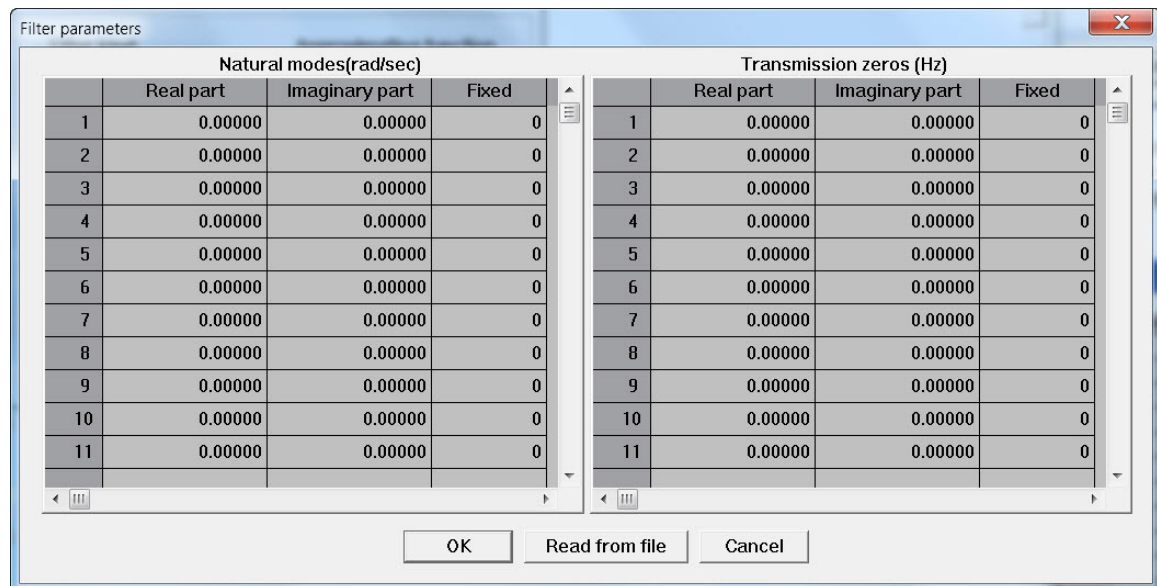
When the file is read, we get to the following data input screen:



The OPT input dialog box contains the following fields and controls:

- Filter kind:** Radio buttons for **Lumped** (selected) and **IIR digital**.
- Approximating function:** A section with a **No. of zeros at zero** spinner set to 0, an **Initial values** button, and a **Flat delay** field set to 0.00000.
- Normalization freq:** A text field containing 2K.
- Sampling frequency:** A text field containing 0.00000.
- Delay weight:** A text field containing 0.00000.
- Buttons:** **OK** and **Cancel** at the bottom.

The normalization frequency is the (upper) passband edge and is needed for just that purpose, normalization. Leaving the overall delay weight at zero tells the program to ignore the delay for this run. Next comes the specification of the approximating function, using the **Initial values** button:



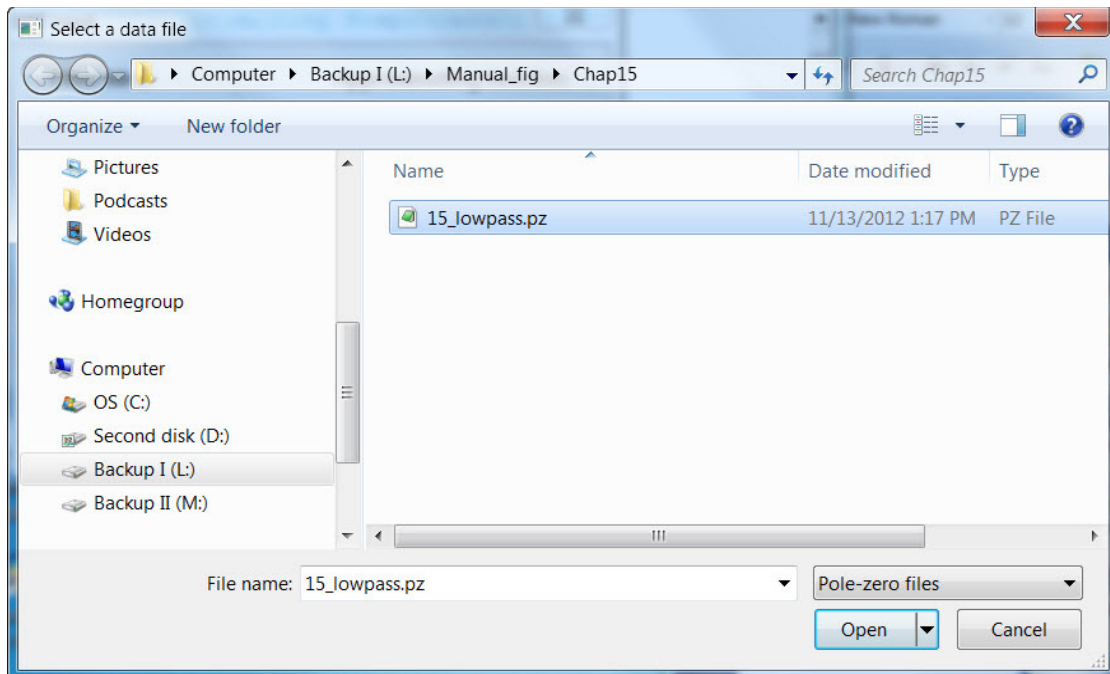
The Filter parameters dialog box displays two tables for manual data entry:

Natural modes(rad/sec)			
	Real part	Imaginary part	Fixed
1	0.00000	0.00000	0
2	0.00000	0.00000	0
3	0.00000	0.00000	0
4	0.00000	0.00000	0
5	0.00000	0.00000	0
6	0.00000	0.00000	0
7	0.00000	0.00000	0
8	0.00000	0.00000	0
9	0.00000	0.00000	0
10	0.00000	0.00000	0
11	0.00000	0.00000	0

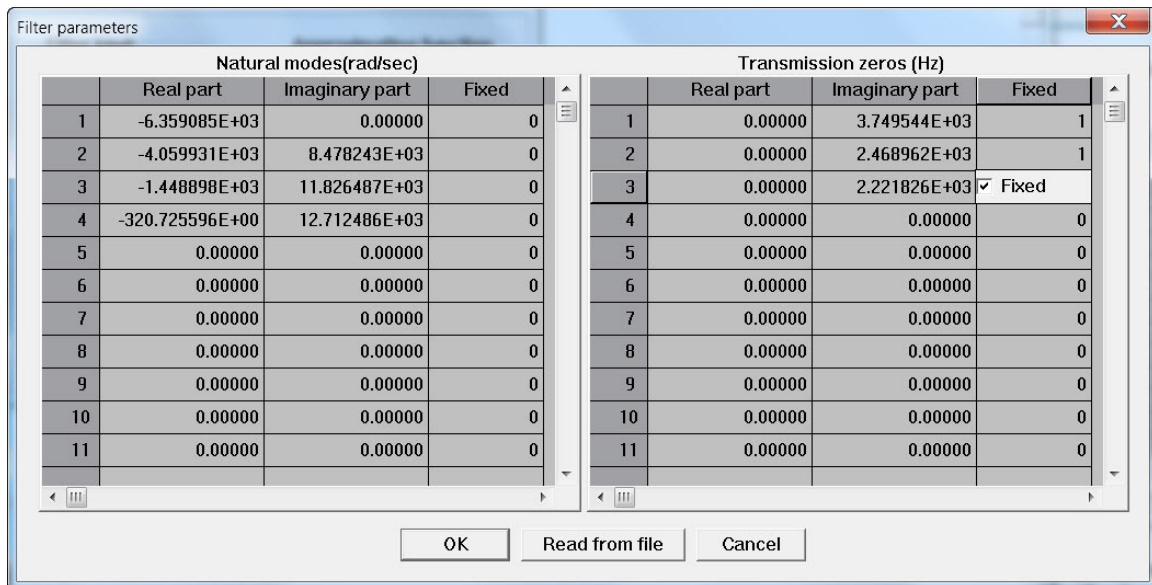
Transmission zeros (Hz)			
	Real part	Imaginary part	Fixed
1	0.00000	0.00000	0
2	0.00000	0.00000	0
3	0.00000	0.00000	0
4	0.00000	0.00000	0
5	0.00000	0.00000	0
6	0.00000	0.00000	0
7	0.00000	0.00000	0
8	0.00000	0.00000	0
9	0.00000	0.00000	0
10	0.00000	0.00000	0
11	0.00000	0.00000	0

At the bottom are **OK**, **Read from file**, and **Cancel** buttons.

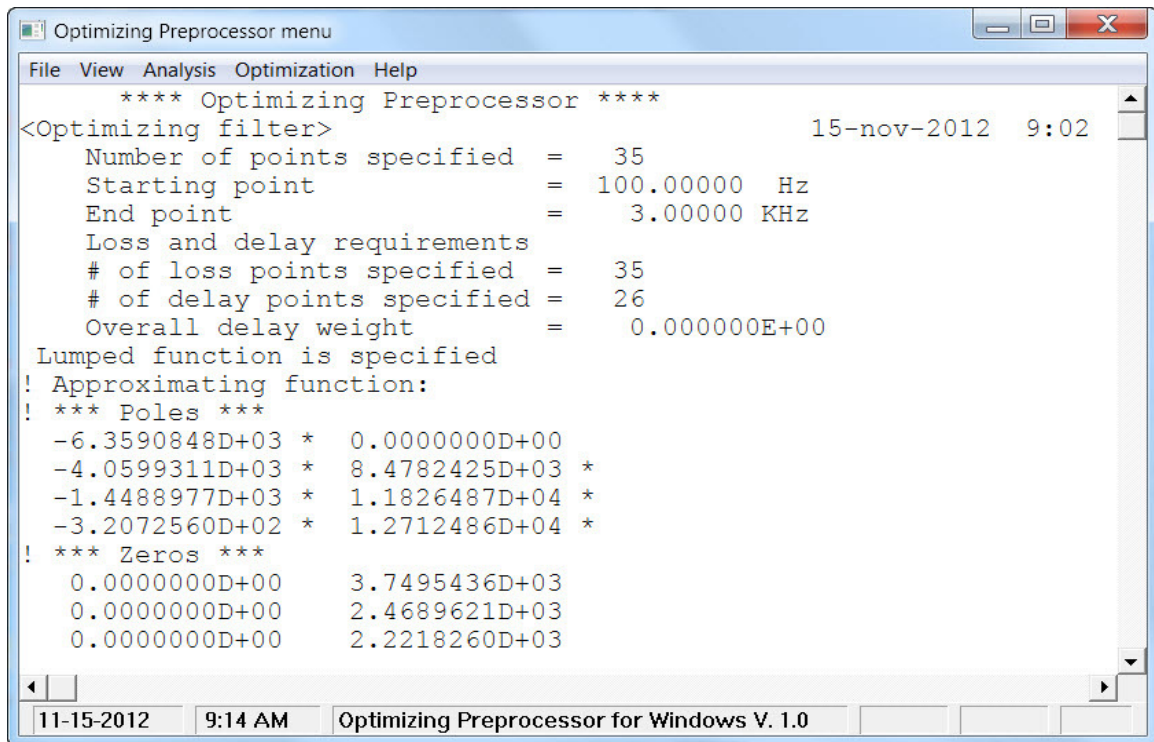
Here we can enter the data manually from the keyboard, or read it in from a text file. Using this option, we read the data from “15_lowpass.pz”:



The program indicates that the file was read and we can check that by clicking on the **Initial values** button again:

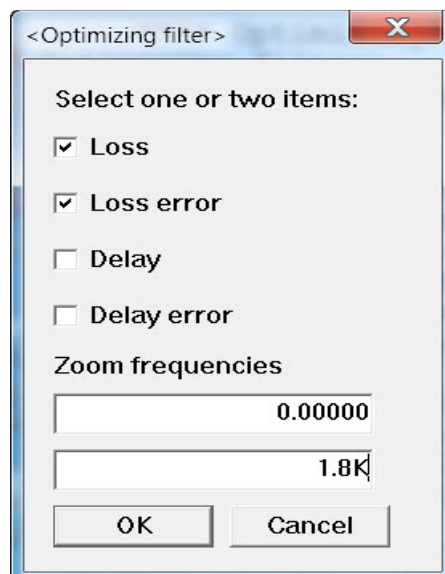


This also enables us to select the parameters we wish to keep unchanged. Since we wish to keep all the transmission zeros fixed, we double clicked on the first three lines of the last (Fixed) column and checked the checkboxes. Clicking on the **OK** buttons twice, we get to the start of the program:

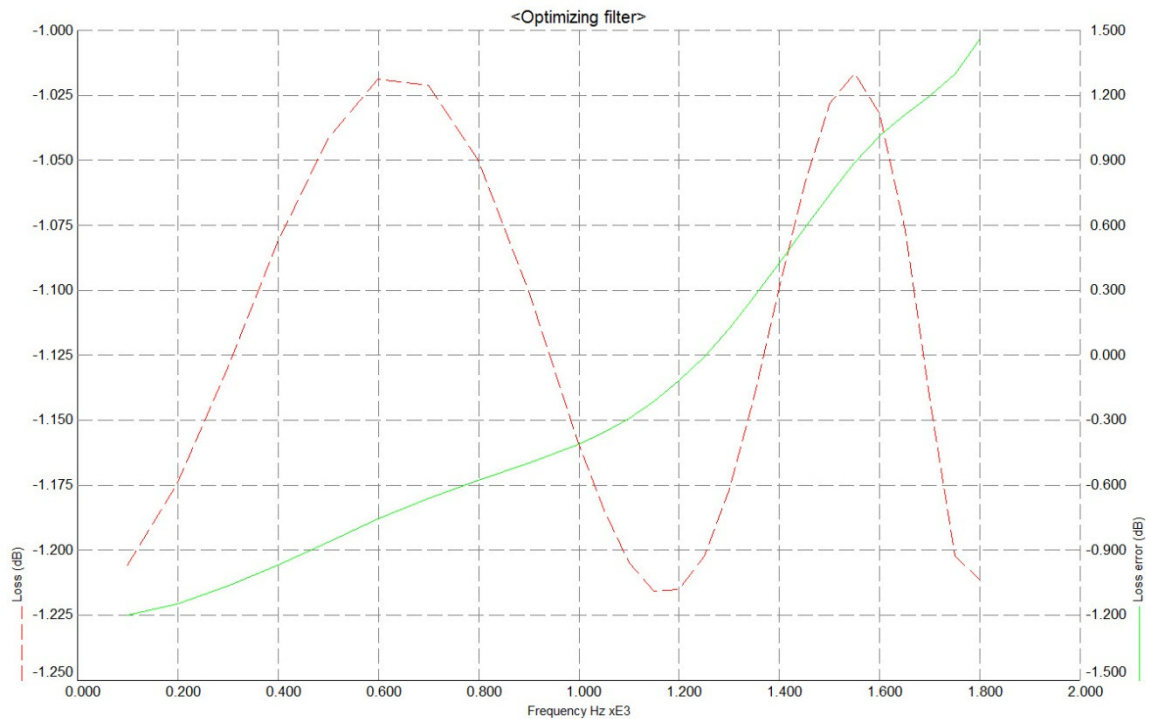


This is just a summary of the data entered and the asterisks indicate variable parameters while those without them are fixed. Incidentally, the maximum number of variables the optimizer can handle at this time is 30. If only the poles are allowed to vary, this permits us to handle functions of up to degree 30.

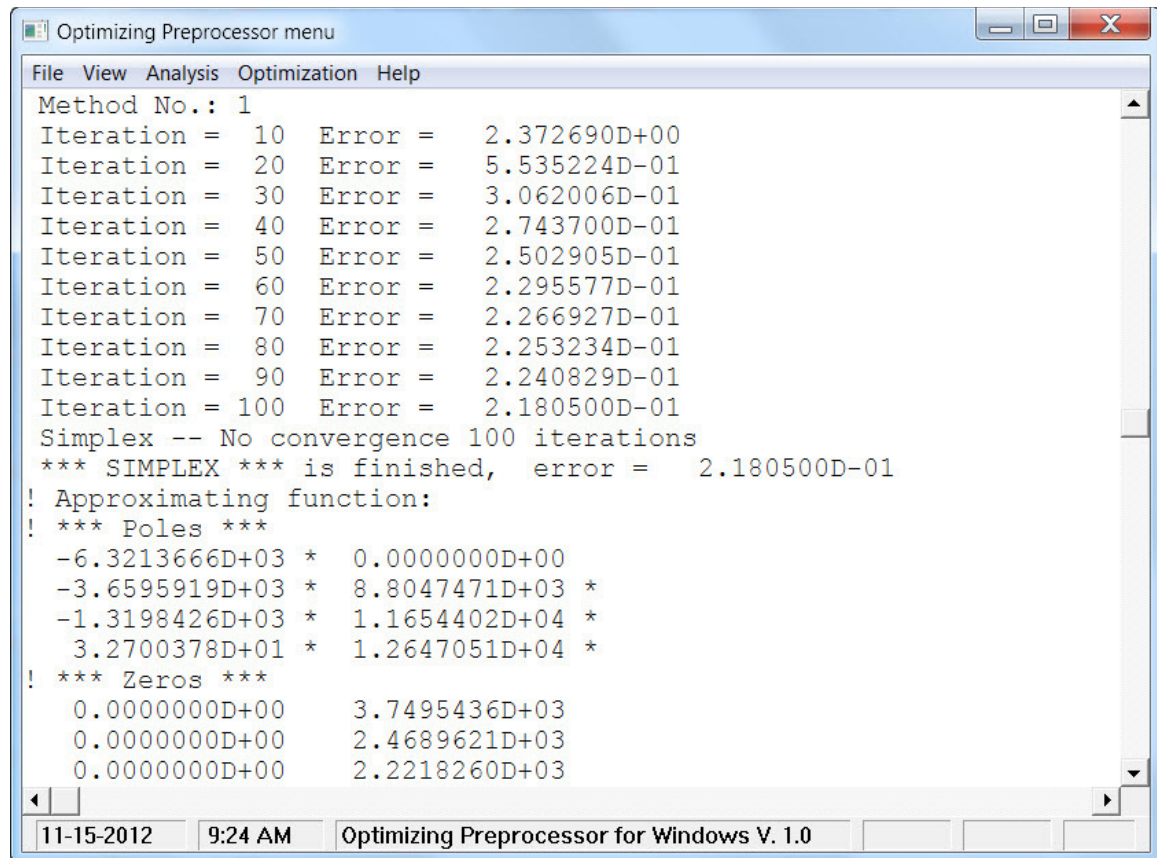
At this stage we have not done any approximation, but the function has already been evaluated and we could either tabulate the results or display them graphically. We elect to do the latter and show the loss and the loss error over the range of 0 to 1800 Hz by selecting **Analysis->Plot**:



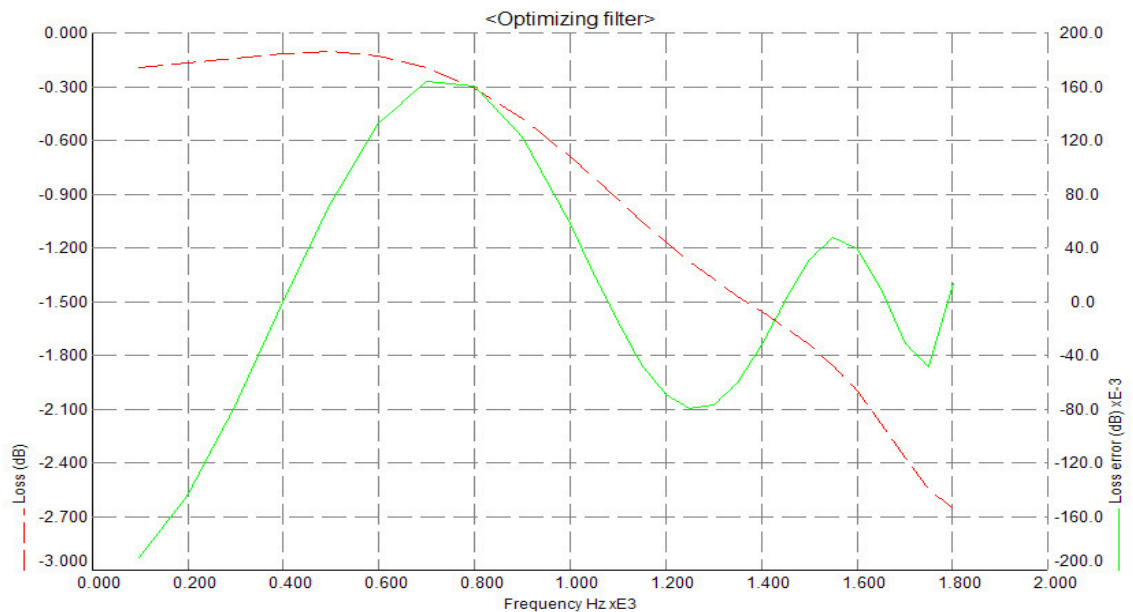
The result is shown next and shows that the loss is the usual equal ripple and the error, of course, is just about the full shape we need:



The delay could also be displayed together with the error even though it is not taken into account in the approximation. The tabulated data could also be written to a text file for documentation purposes. We are ready to start the optimization, using method no. 1 (**Optimization->Method 1**):



Plotting the loss and the loss error again, shows the improvement:



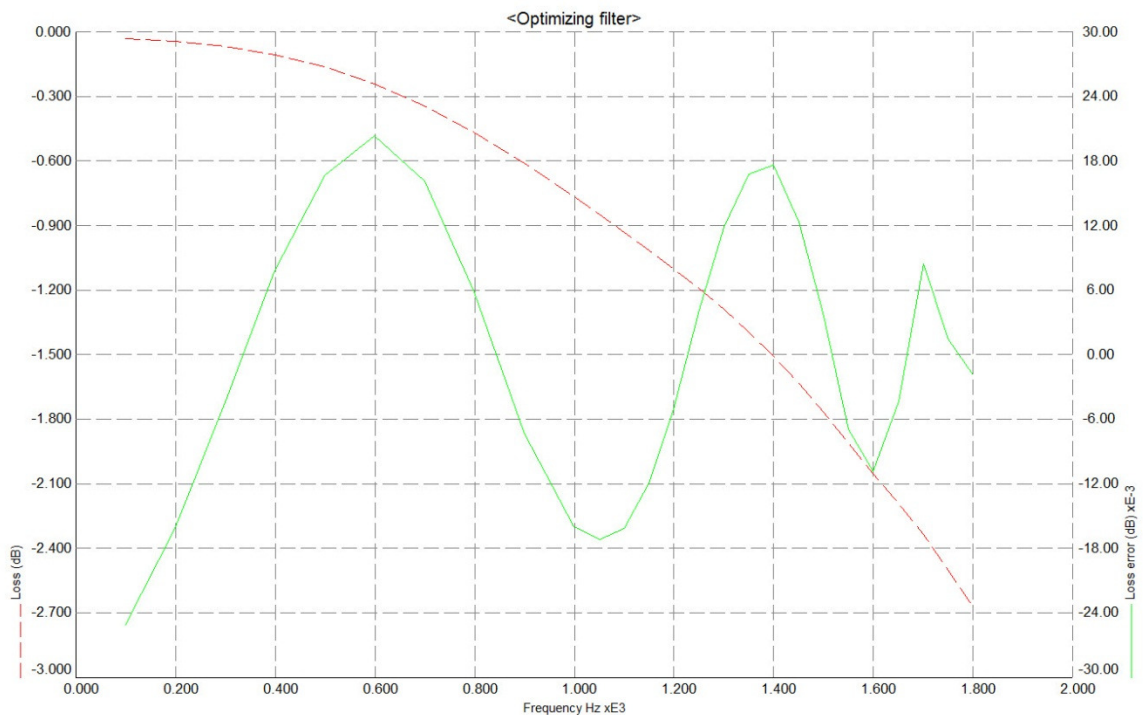
This method improved the error a bit more than an order of magnitude. **Method 2** did not do any good at this point, so we did a large number of all three methods in an arbitrary sequence, stopping at this point:

```

Optimizing Preprocessor menu
File View Analysis Optimization Help
Method No.: 3
Iteration = 5 error = 5.192119D-03
Iteration = 10 error = 5.177992D-03
Iteration = 15 error = 5.154064D-03
Iteration = 20 error = 5.150353D-03
Iteration = 25 error = 5.119860D-03
Iteration = 30 error = 5.064728D-03
Method 3 warning -- Info = 3: Cannot find lower point
*** GRADIENT *** is finished, error = 5.064410D-03
! Approximating function:
! *** Poles ***
-8.1132025D+03 * 0.0000000D+00
-4.8717448D+03 * 8.2772032D+03 *
-1.9582391D+03 * 1.1030634D+04 *
-5.0789469D+02 * 1.1830353D+04 *
! *** Zeros ***
0.0000000D+00 3.7495436D+03
0.0000000D+00 2.4689621D+03
0.0000000D+00 2.2218260D+03
11-15-2012 9:34 AM Optimizing Preprocessor for Windows V. 1.0

```

This improved things a bit less than two orders of magnitude and then slowed down. This may indicate that we are close to the optimum. At this point we elect to stop the iteration and we get the graphical results:



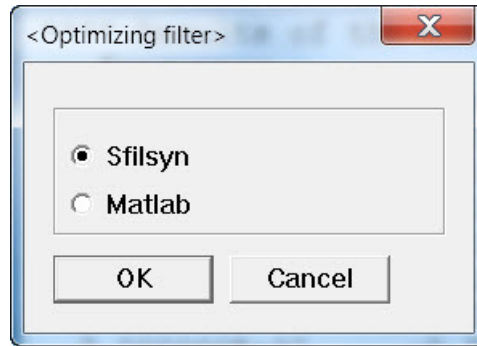
It is not easy to decide when to stop the iteration. Usually, we can look at the deviation function and count the number of zero crossings. If this number is about the same, or more, than the number of free parameters we have, we can be fairly confident, that we are indeed close to the optimum. The graphical results show a better than ± 0.03 dB error and the number of sign changes in the error does seem to indicate that this is now at or very near the global optimum.

Note that during the optimization, the real part of a pole sometime became positive (unstable). This can be corrected by using the **Optimization->Stabilize** menu option, which changes all real parts of the poles back to negative. This has no effect on the loss, only on the delay, which is of no concern at the moment. Other options in this column – **Modify** – permits us to change values of the parameters and – **Undo** – to go back a step in case the optimization step ended with a worse error than it started with.

We can also tabulate the analysis results using the **Analysis->Tabulate** menu item and the first few lines will show:

**** Results of the analysis ****				
Frequency in Hz	Loss values		Delay values	
	Actual	Deviation	Actual	Deviation
1.000000E+02	-0.032080	-0.025080	2.678331E-04	-1.732167E-03
2.000000E+02	-0.045096	-0.016096	2.694344E-04	-1.730566E-03
3.000000E+02	-0.069369	-0.004369	2.724015E-04	-1.727599E-03
4.000000E+02	-0.108233	0.007767	2.771479E-04	-1.722852E-03
5.000000E+02	-0.165320	0.016680	2.841913E-04	-1.715809E-03
6.000000E+02	-0.243678	0.020322	2.940990E-04	-1.705901E-03
7.000000E+02	-0.344870	0.016130	3.074259E-04	-1.692574E-03
8.000000E+02	-0.468180	0.005820	3.246482E-04	-1.675352E-03
9.000000E+02	-0.610297	-0.007297	3.461223E-04	-1.653878E-03
1.000000E+03	-0.765948	-0.015948	3.721412E-04	-1.627859E-03
1.050000E+03	-0.847178	-0.017178	3.869967E-04	-1.613003E-03
1.100000E+03	-0.930132	-0.016132	4.032446E-04	-1.596755E-03
1.150000E+03	-1.014876	-0.011876	4.211190E-04	-1.578881E-03
1.200000E+03	-1.102023	-0.005023	4.410016E-04	-1.558998E-03
1.250000E+03	-1.192828	0.004172	4.634812E-04	-1.536519E-03
1.300000E+03	-1.289156	0.011844	4.894155E-04	-1.510584E-03
1.350000E+03	-1.393275	0.016725	5.199873E-04	-1.480013E-03
1.400000E+03	-1.507357	0.017643	5.567358E-04	-1.443264E-03
1.450000E+03	-1.632655	0.012345	6.015361E-04	-1.398464E-03
1.500000E+03	-1.768408	0.003592	6.565037E-04	-1.343496E-03
1.550000E+03	-1.910891	-0.006891	7.238726E-04	-1.276128E-03
1.600000E+03	-2.053812	-0.010813	8.062227E-04	-1.193777E-03
1.650000E+03	-2.192320	-0.004320	9.084010E-04	-1.091599E-03
1.700000E+03	-2.332579	0.008421	1.044775E-03	-9.552246E-04
1.750000E+03	-2.498557	0.001443	1.259718E-03	-7.402826E-04
1.800000E+03	-2.669882	-0.001882	1.667218E-03	-3.327825E-04

At this stage we can still go back to perform more iterations, but decided that these results were acceptable and wrote the data in a file, using the **File->Transfer** menu option:

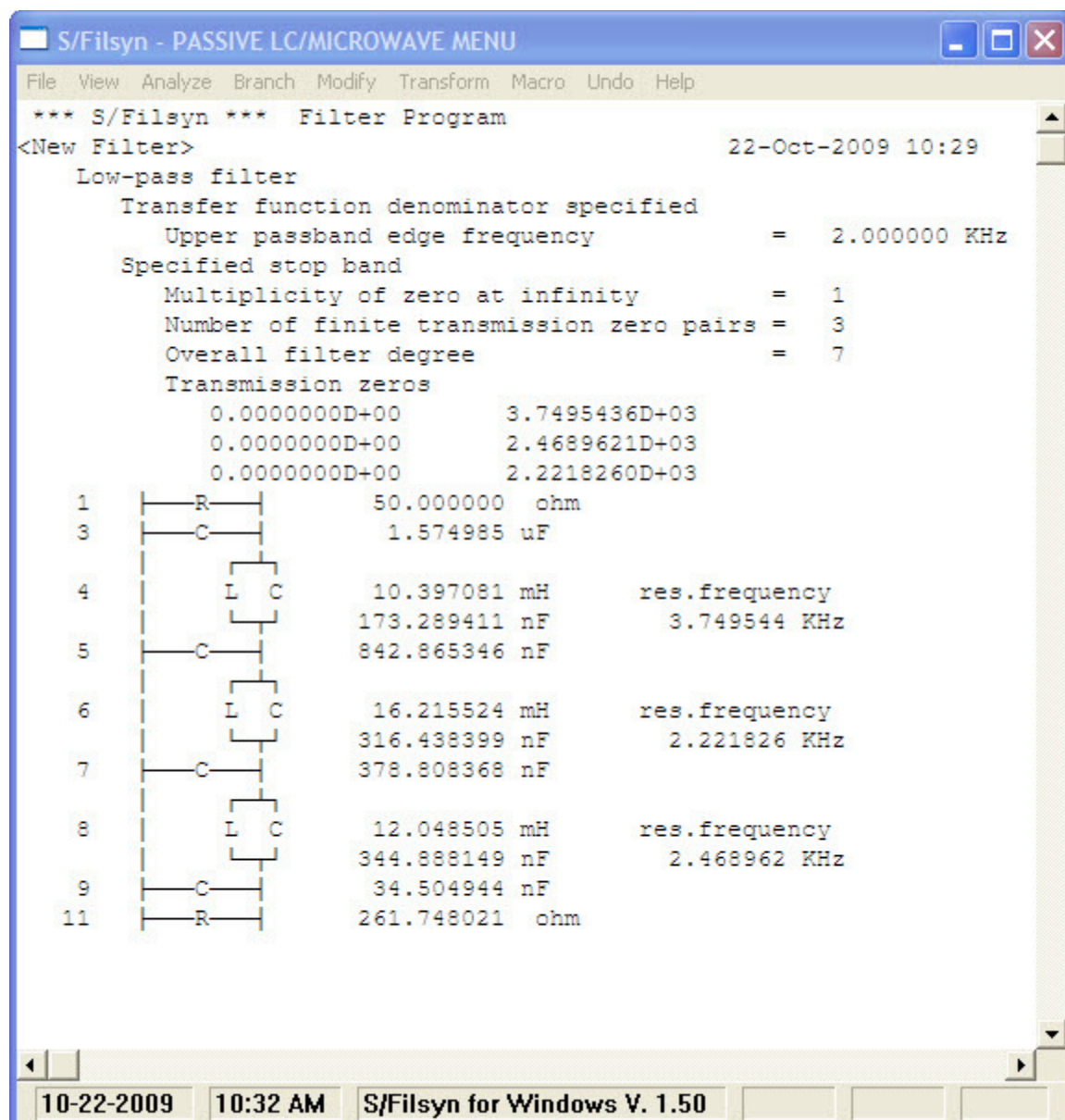


The generated pole-zero file is then saved as “*results.pz*” and is displayed below using a simple text editor, but can in fact be also displayed inside the program using the **File->Open** menu option and specifying the file:

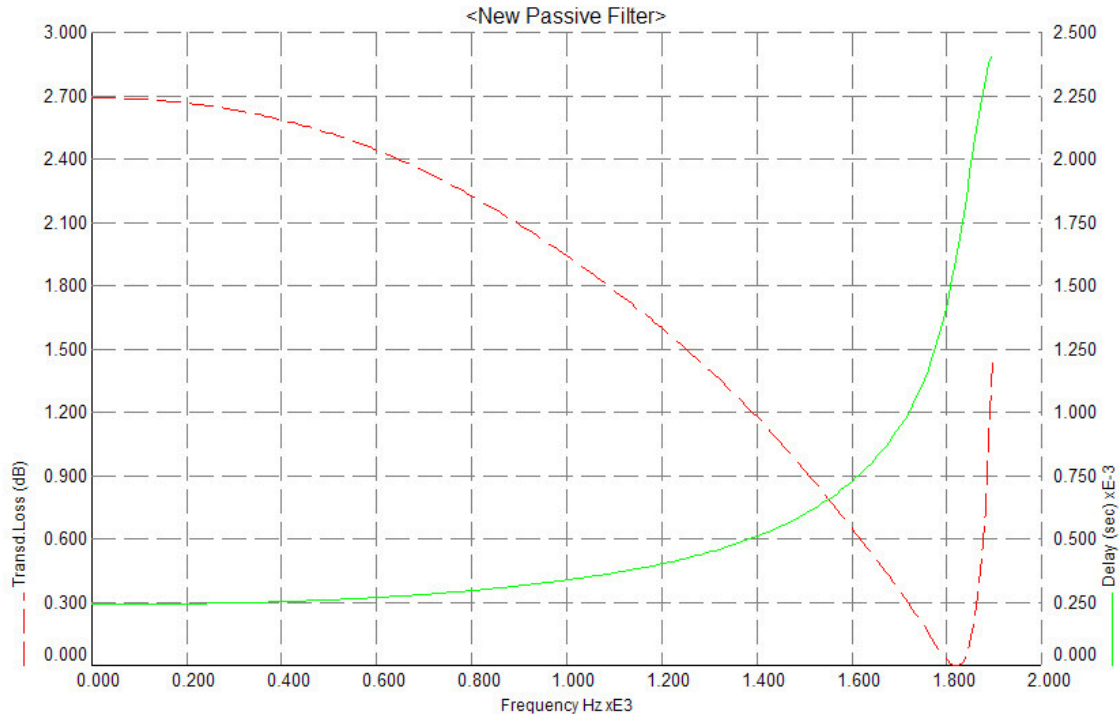
```

!<Optimizing filter>
! Approximating function:
! *** Poles ***
-8.1132025D+03      0.0000000D+00
-4.8717448D+03      8.2772032D+03
-1.9582391D+03      1.1030634D+04
-5.0789469D+02      1.1830353D+04
! *** Zeros ***
0.0000000D+00      3.7495436D+03
0.0000000D+00      2.4689621D+03
0.0000000D+00      2.2218260D+03
  
```

which leads to the LC implementation by the use of the functional input synthesis method of the **Filsyn** program:



Analyzing this circuit at the same frequencies, yields identical results to those obtained in the optimizer save for a flat loss:



15.7.2 Example 2

We shall use the same set of requirements but this time we will take the delay requirements into account as well. We will set the overall delay weight to 0.02 and the flat delay to -1. This setting indicates to the program that it should use the flat delay as a free parameter and set it to its optimal value. Later on we shall have the ability to change both of these values.

OPT input

Filter kind:

- ☒ Lumped
- ☐ IIR digital

Approximating function

No. of zeros at zero: 0

Initial values

Normalization freq: 2.000000E+03

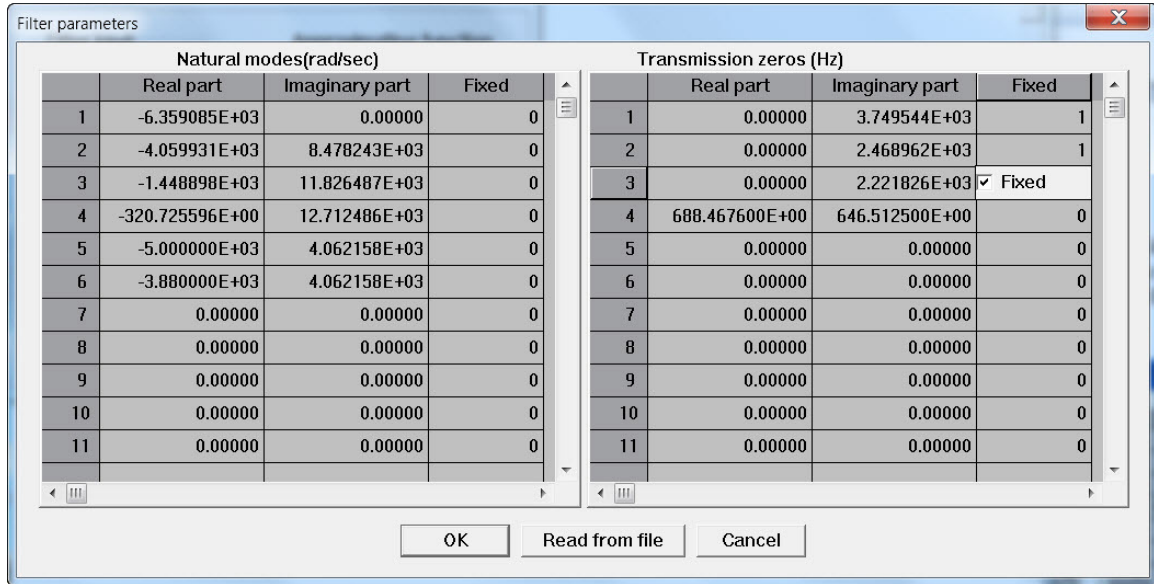
Sampling frequency: 0.00000

Flat delay: -1.000000E+00

Delay weight: 0.02

OK Cancel

The starting values are also the same except we added a complex transmission zero (actually a quadruplet) and two complex poles near the new zeros. The zero was selected by doing a delay equalization step of the original elliptic lowpass filter in the range from 0 to 1800 Hz using one second order delay section.

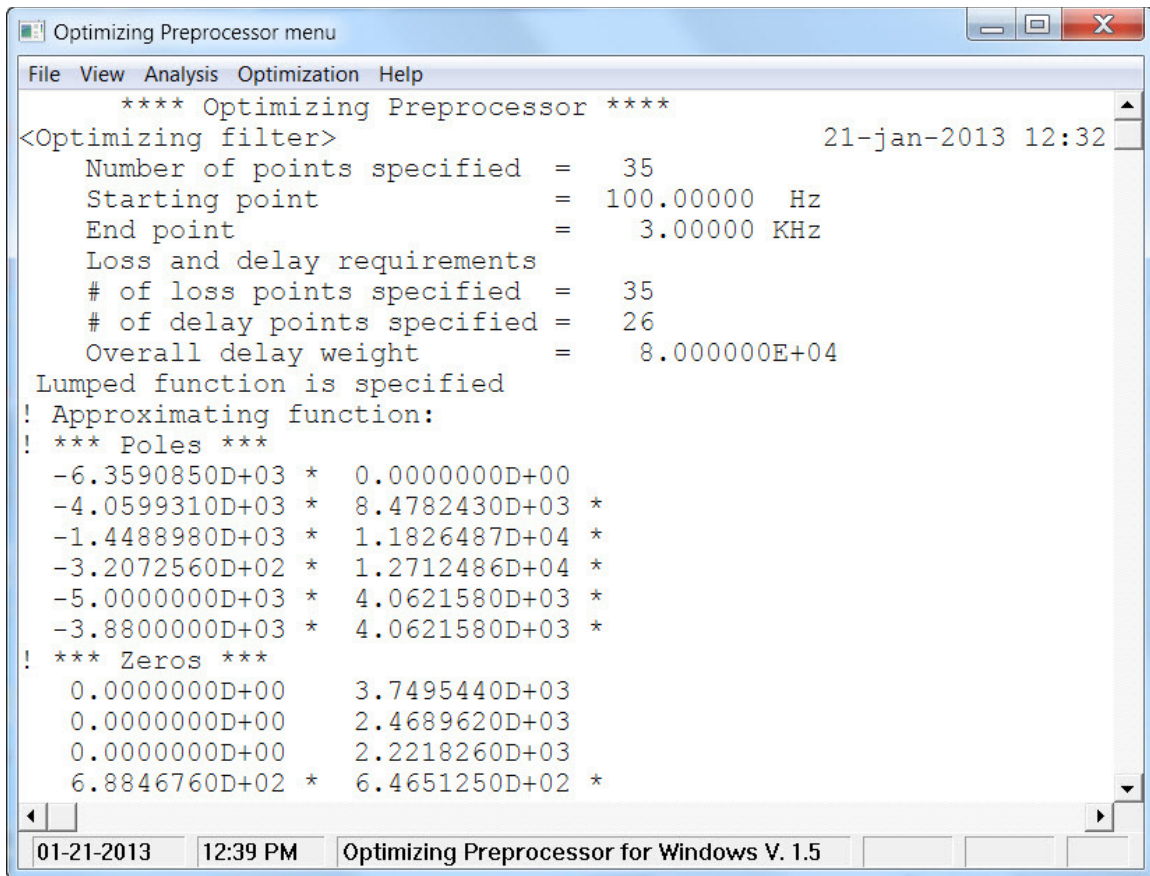


The dialog box titled "Filter parameters" contains two tables. The left table, "Natural modes(rad/sec)", has 11 rows with columns for Real part, Imaginary part, and Fixed. The right table, "Transmission zeros (Hz)", also has 11 rows with the same columns. In the right table, the third row is highlighted, and the "Fixed" column contains a checkmark and the word "Fixed". At the bottom of the dialog are three buttons: "OK", "Read from file", and "Cancel".

Natural modes(rad/sec)			
	Real part	Imaginary part	Fixed
1	-6.359085E+03	0.00000	0
2	-4.059931E+03	8.478243E+03	0
3	-1.448898E+03	11.826487E+03	0
4	-320.725596E+00	12.712486E+03	0
5	-5.000000E+03	4.062158E+03	0
6	-3.880000E+03	4.062158E+03	0
7	0.00000	0.00000	0
8	0.00000	0.00000	0
9	0.00000	0.00000	0
10	0.00000	0.00000	0
11	0.00000	0.00000	0

Transmission zeros (Hz)			
	Real part	Imaginary part	Fixed
1	0.00000	3.749544E+03	1
2	0.00000	2.468962E+03	1
3	0.00000	2.221826E+03	✓ Fixed
4	688.467600E+00	646.512500E+00	0
5	0.00000	0.00000	0
6	0.00000	0.00000	0
7	0.00000	0.00000	0
8	0.00000	0.00000	0
9	0.00000	0.00000	0
10	0.00000	0.00000	0
11	0.00000	0.00000	0

Again we fixed all pure imaginary transmission zeros but will let the complex value change. Clicking on the **OK** button, we are ready to start the optimization.



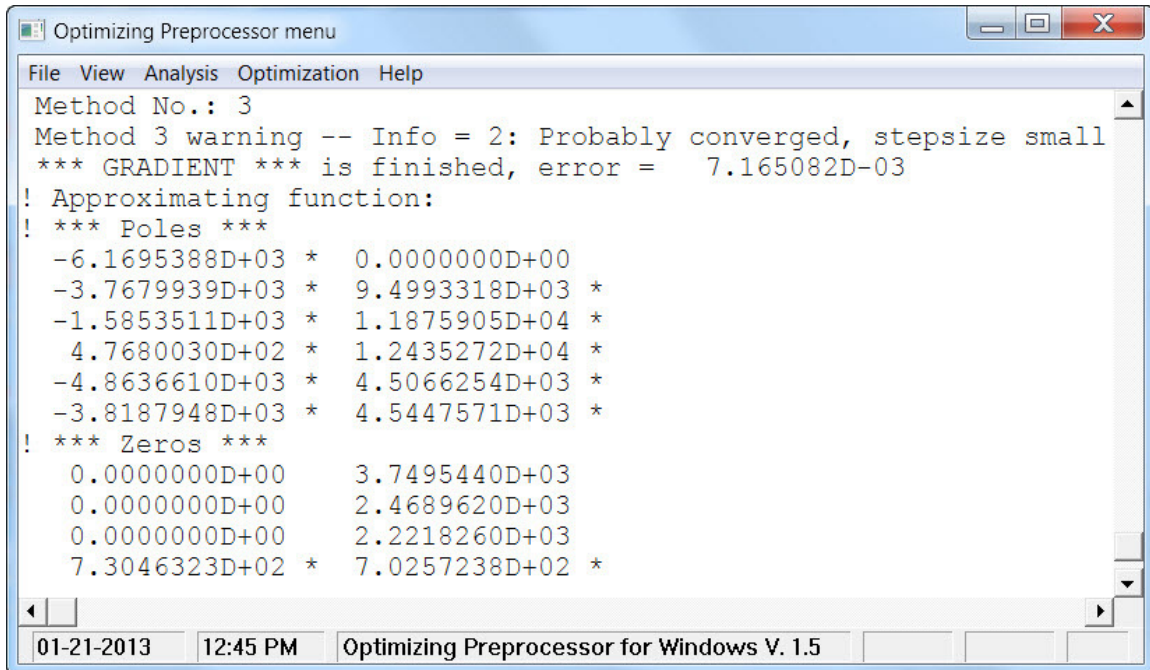
```

Optimizing Preprocessor menu
File View Analysis Optimization Help
**** Optimizing Preprocessor ****
<Optimizing filter> 21-jan-2013 12:32
Number of points specified = 35
Starting point = 100.00000 Hz
End point = 3.00000 KHz
Loss and delay requirements
# of loss points specified = 35
# of delay points specified = 26
Overall delay weight = 8.000000E+04
Lumped function is specified
! Approximating function:
! *** Poles ***
-6.3590850D+03 * 0.0000000D+00
-4.0599310D+03 * 8.4782430D+03 *
-1.4488980D+03 * 1.1826487D+04 *
-3.2072560D+02 * 1.2712486D+04 *
-5.0000000D+03 * 4.0621580D+03 *
-3.8800000D+03 * 4.0621580D+03 *
! *** Zeros ***
0.0000000D+00 3.7495440D+03
0.0000000D+00 2.4689620D+03
0.0000000D+00 2.2218260D+03
6.8846760D+02 * 6.4651250D+02 *
01-21-2013 12:39 PM Optimizing Preprocessor for Windows V. 1.5

```

Note that the delay weight has been changed, since the numerical values of the delay itself are much lower than the loss, hence a rescaling by the square of the normalization frequency is called for.

Using the three methods of optimization in sequence a few times, we reach the following results:



The screenshot shows a window titled "Optimizing Preprocessor menu" with a menu bar (File, View, Analysis, Optimization, Help) and a text area containing the following output:

```

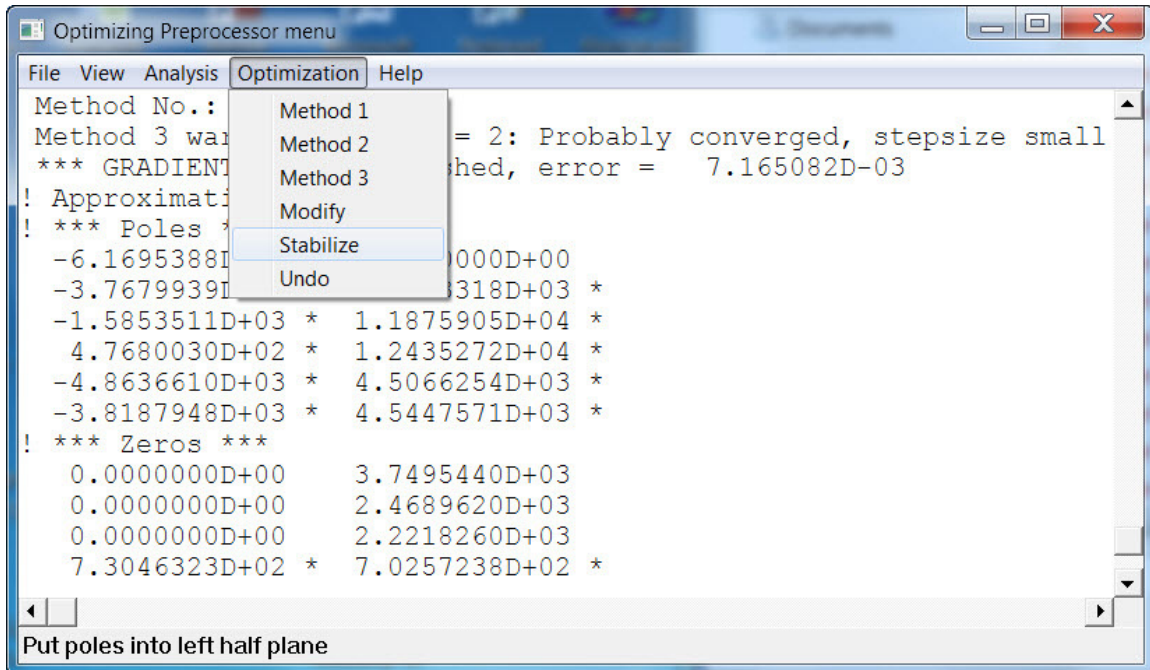
Method No.: 3
Method 3 warning -- Info = 2: Probably converged, stepsize small
*** GRADIENT *** is finished, error = 7.165082D-03
! Approximating function:
! *** Poles ***
-6.1695388D+03 * 0.0000000D+00
-3.7679939D+03 * 9.4993318D+03 *
-1.5853511D+03 * 1.1875905D+04 *
 4.7680030D+02 * 1.2435272D+04 *
-4.8636610D+03 * 4.5066254D+03 *
-3.8187948D+03 * 4.5447571D+03 *
! *** Zeros ***
 0.0000000D+00 3.7495440D+03
 0.0000000D+00 2.4689620D+03
 0.0000000D+00 2.2218260D+03
 7.3046323D+02 * 7.0257238D+02 *
    
```

The status bar at the bottom shows the date "01-21-2013", time "12:45 PM", and version "Optimizing Preprocessor for Windows V. 1.5".

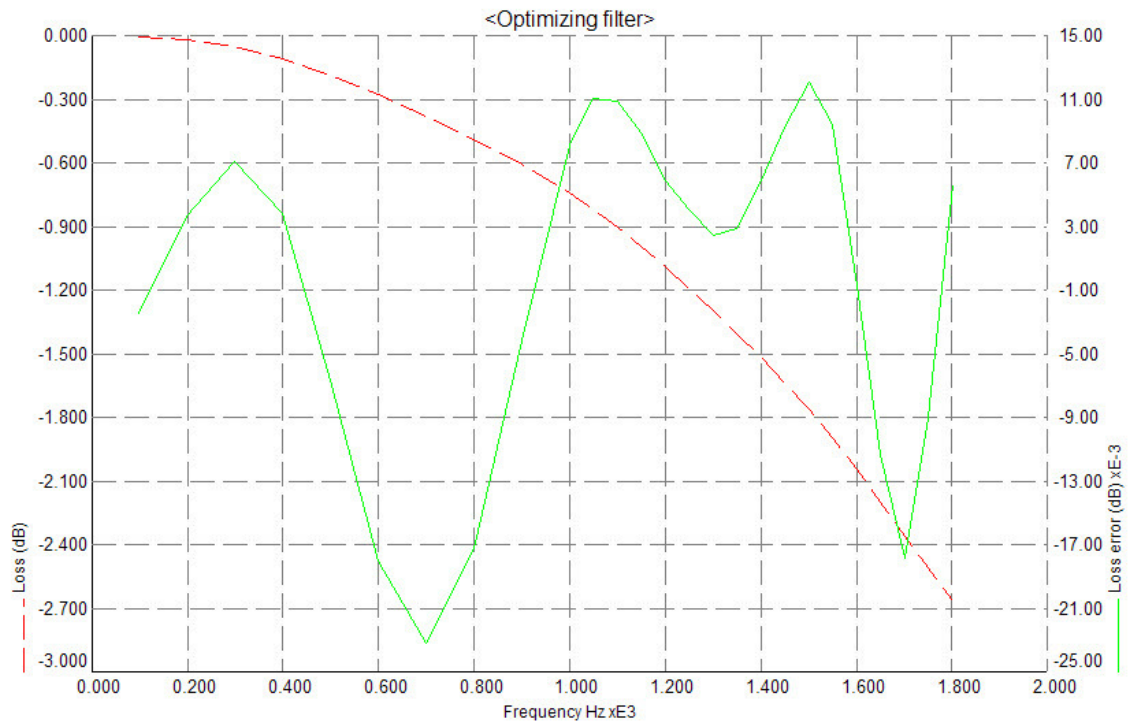
During this stage, no matter which optimization strategy we use, we end up by having at least one of the poles becoming unstable (having a positive real part). When that happens we have several options:

- We can simply back up using the **Optimization->Undo** menu option (which is what we have done here)
- We can use the **Optimization->Modify** option to change the delay weight and/or the flat delay
- We can just use the **Optimization->Stabilize** option to change the sign of the positive real part(s) back to negative

and in all of these cases continue to try to optimize the filter. Whether any of these methods would yield a better solution is left for the user to try and answer. We selected here the **Stabilize** option:



Going back to the results on the previous page if we need to look at the results, we can use the **View->Show** menu option to reprint the last set of data and repeat the analysis, then used the **Analysis->Plot** option to show the loss and the loss error:



Clearly, the loss approximation is not as good as it was with no concern for the delay, but it is quite acceptable and the delay (not shown here) is also reasonable. The stopband

performance is about the same as in the first example, i.e. quite good, therefore we only show the details of the passband above.

Proceeding to the actual realization of this transfer function we use the functional input option and read in the poles and zeros that the optimizer supplied:

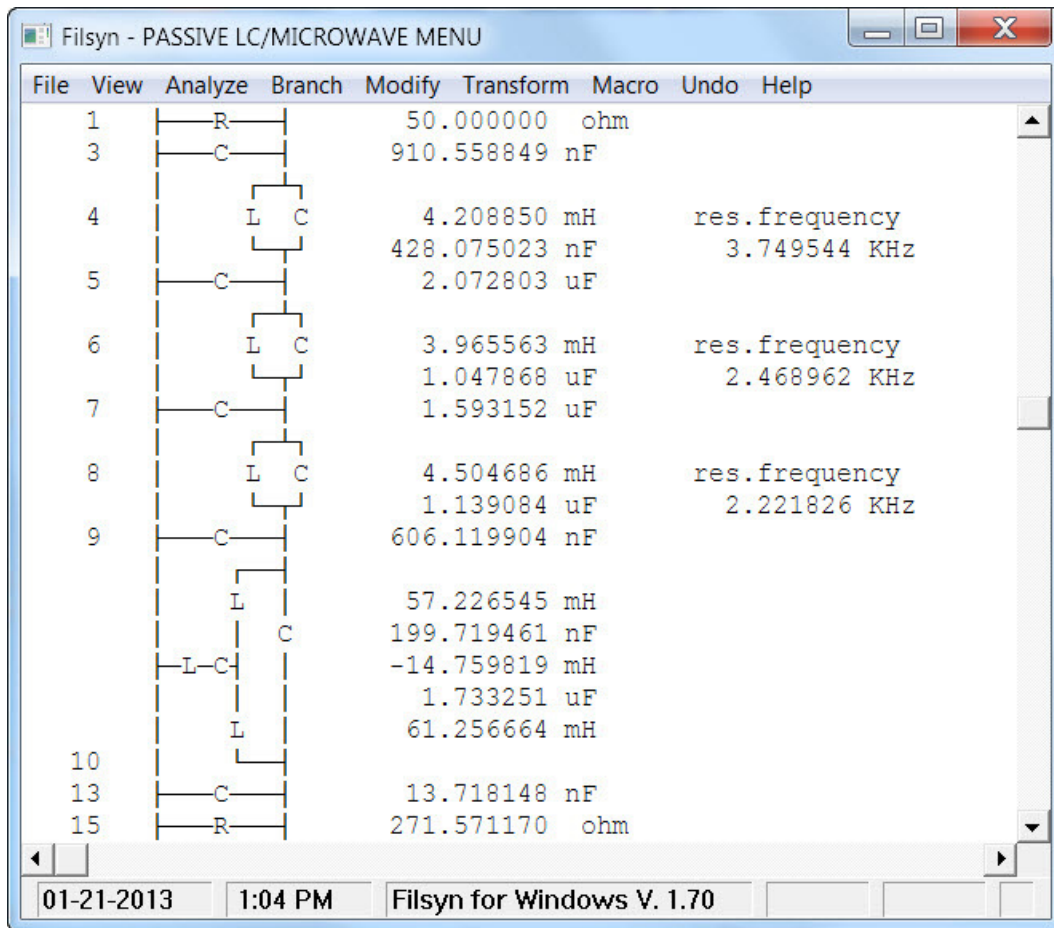
```
!<Optimizing filter> 21-jan-2013 12:32
! Approximating function:
! *** Poles ***
-6.1695388D+03 0.0000000D+00
-3.7679939D+03 9.4993318D+03
-1.5853511D+03 1.1875905D+04
-4.7680030D+02 1.2435272D+04
-4.8636610D+03 4.5066254D+03
-3.8187948D+03 4.5447571D+03
! *** Zeros ***
0.0000000D+00 3.7495440D+03
0.0000000D+00 2.4689620D+03
0.0000000D+00 2.2218260D+03
7.3046323D+02 7.0257238D+02
```

Reading this data into the program:

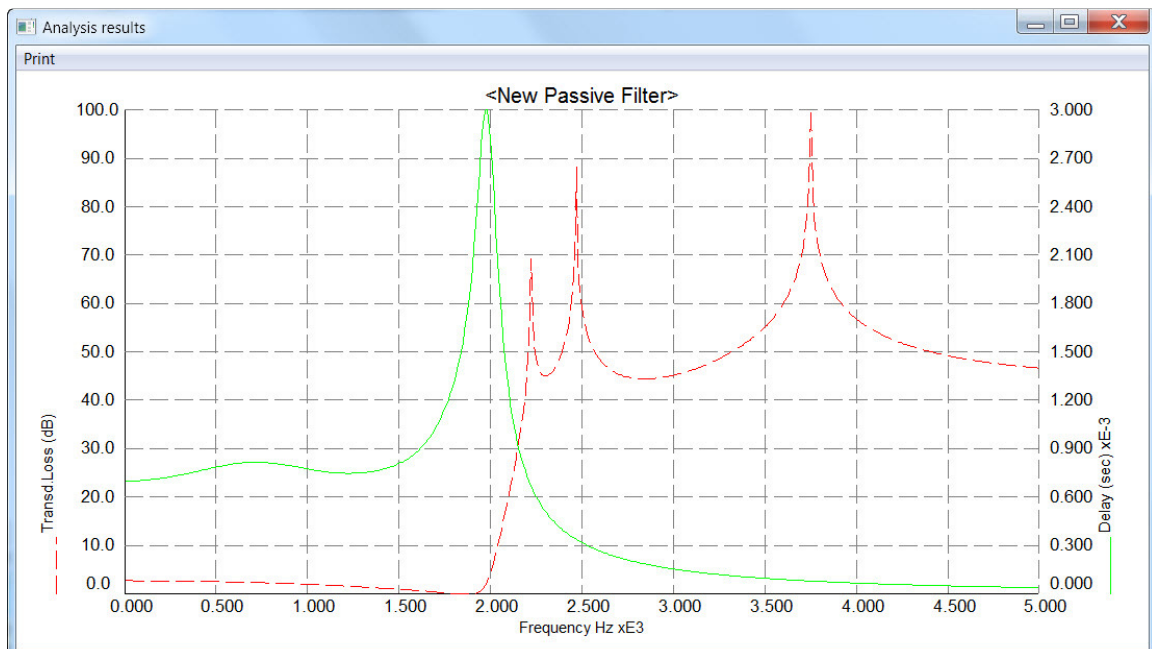
Function zeros (in rad/sec)			Transmission zeros (in Hz)		
	Real Part	Imaginary Part		Real Part	Imaginary Part
1	-6.169539E+03	0.00000	1	0.00000	3.749544E+03
2	-3.767994E+03	9.499332E+03	2	0.00000	2.468962E+03
3	-1.585351E+03	11.875905E+03	3	0.00000	2.221826E+03
4	-476.800300E+00	12.435272E+03	4	730.463230E+00	702.572380E+00
5	-4.863661E+03	4.506625E+03	5	0.00000	0.00000
6	-3.818795E+03	4.544757E+03	6	0.00000	0.00000
7	0.00000	0.00000	7	0.00000	0.00000
8	0.00000	0.00000	8	0.00000	0.00000
9	0.00000	0.00000	9	0.00000	0.00000
10	0.00000	0.00000	10	0.00000	0.00000
11	0.00000	0.00000	11	0.00000	0.00000

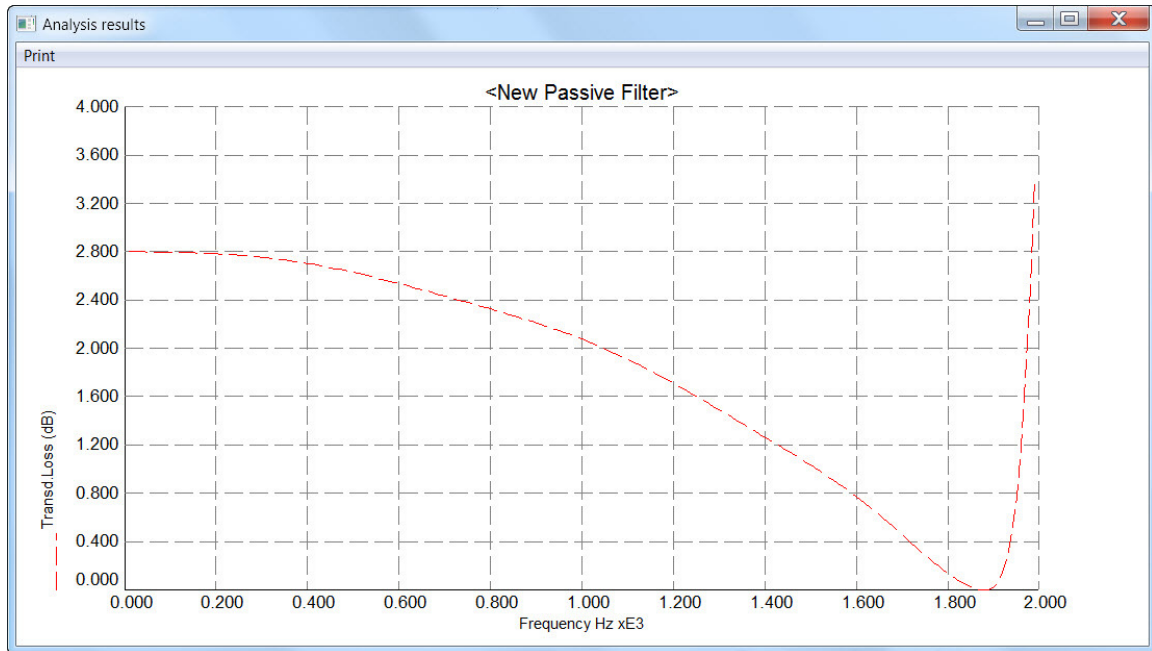
OK Read from file Cancel

The resulting filter is fine and realizable:



The analysis results show a very satisfactory response with perfect equal minima stopband, reasonable delay flatness and very good passband shown on the next page.





Along an entirely different line, we could simply take the circuit obtained in Example 1 above and equalize its delay. This yields an overall solution that is substantially better than this solution obtained by the simultaneous approximation described here.

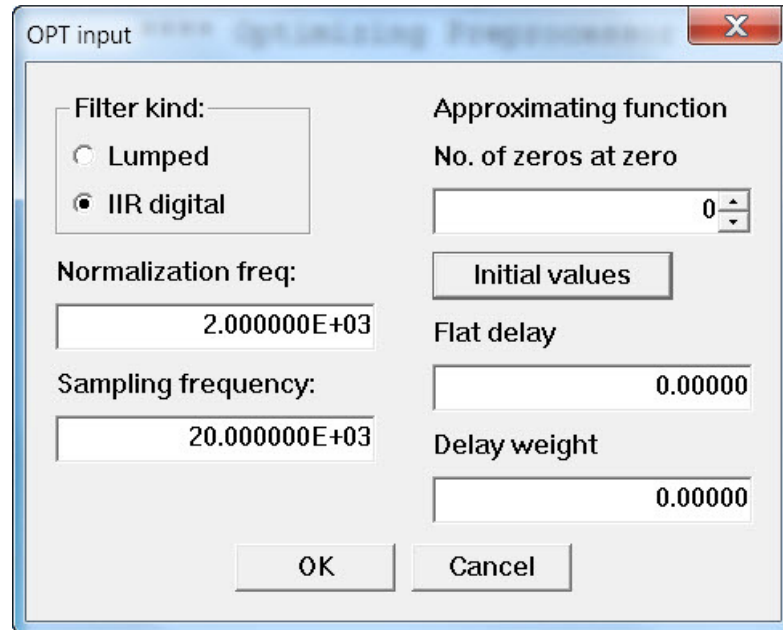
Naturally, this is only one example and we should not draw any major conclusions from it. The only thing we wish to point out is that one should keep a completely open mind regarding optimization in general, and simultaneous optimization of loss and delay, in particular.

15.7.3 Example 3

Our last example will be an IIR digital filter to follow the same set of requirements. The initial approximating function will be an 8th order elliptic design using the same specifications, with the exception of a 0.1 dB passband loss, and a 20 KHz sampling rate. **Filsyn** has written the data into a file where the poles are in normalized form but the zeros are still given in Hz (the standard for both **Filsyn** and **Opt**):

```
! Pole-zero data of:
! <New Digital Filter>
! Poles (normalized):
-7.236004788E-01    -1.648727387E-01
-7.520196777E-01    -4.155345549E-01
-7.767109164E-01    -5.386498370E-01
-7.932337509E-01    -5.864711324E-01
! Zeros (in Hz):
0.000000000E+00    2.215874324E+03
0.000000000E+00    2.378764046E+03
0.000000000E+00    2.996403218E+03
0.000000000E+00    5.797002418E+03
```

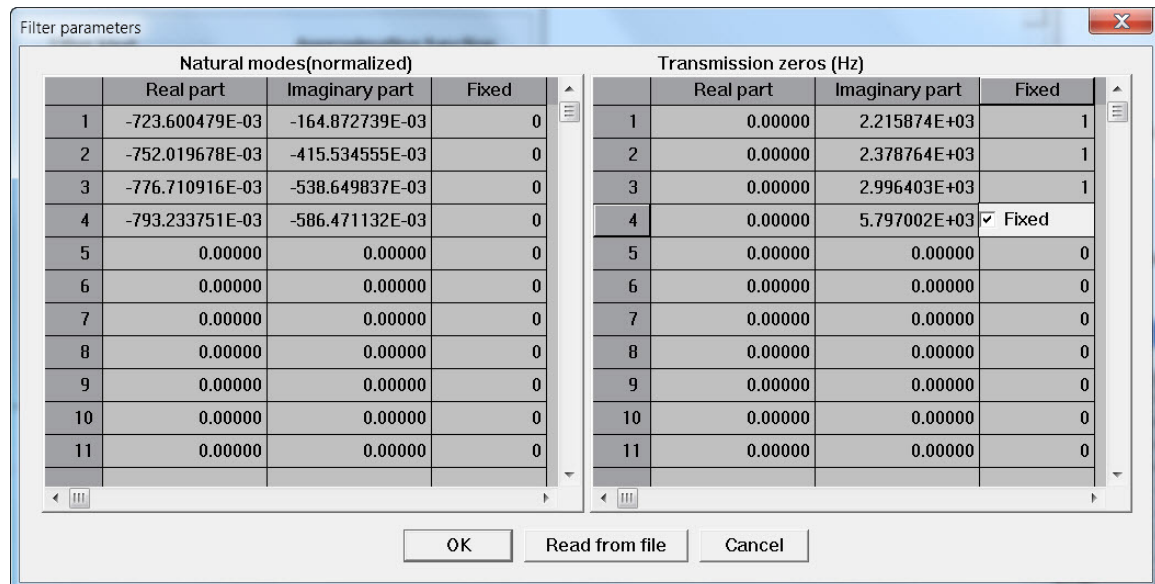
Using the same requirement table as before but ignoring the delay, our run is specified as follows:



The OPT input dialog box contains the following fields and controls:

- Filter kind:** Radio buttons for ☐ Lumped and ☒ IIR digital.
- Approximating function:** A dropdown menu showing '0'.
- No. of zeros at zero:** A numeric input field with '0'.
- Initial values:** A button.
- Normalization freq:** A numeric input field with '2.000000E+03'.
- Flat delay:** A numeric input field with '0.00000'.
- Sampling frequency:** A numeric input field with '20.000000E+03'.
- Delay weight:** A numeric input field with '0.00000'.
- Buttons:** OK and Cancel.

The **Initial values** window shows the values read in from the pole-zero table and the zeros are all fixed:



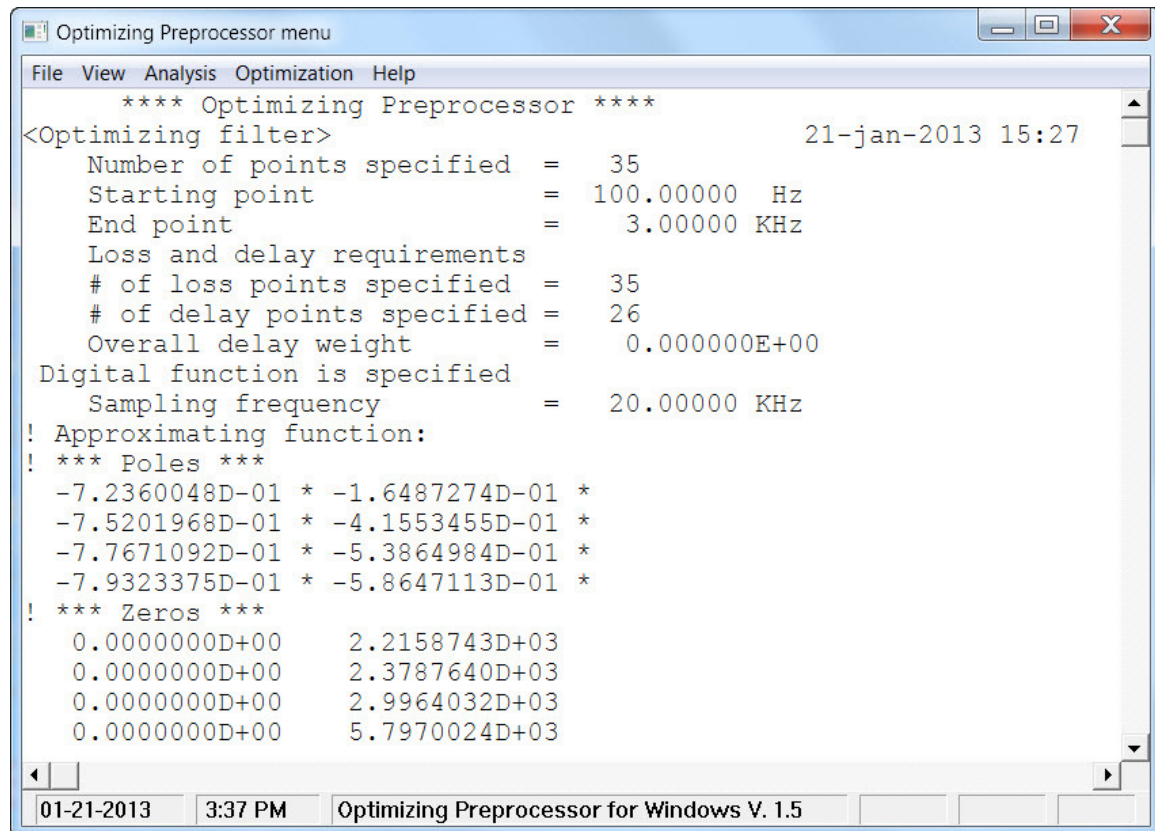
The Filter parameters dialog box displays two tables: Natural modes(normalized) and Transmission zeros (Hz).

Natural modes(normalized)			
	Real part	Imaginary part	Fixed
1	-723.600479E-03	-164.872739E-03	0
2	-752.019678E-03	-415.534555E-03	0
3	-776.710916E-03	-538.649837E-03	0
4	-793.233751E-03	-586.471132E-03	0
5	0.00000	0.00000	0
6	0.00000	0.00000	0
7	0.00000	0.00000	0
8	0.00000	0.00000	0
9	0.00000	0.00000	0
10	0.00000	0.00000	0
11	0.00000	0.00000	0

Transmission zeros (Hz)			
	Real part	Imaginary part	Fixed
1	0.00000	2.215874E+03	1
2	0.00000	2.378764E+03	1
3	0.00000	2.996403E+03	1
4	0.00000	5.797002E+03	Fixed
5	0.00000	0.00000	0
6	0.00000	0.00000	0
7	0.00000	0.00000	0
8	0.00000	0.00000	0
9	0.00000	0.00000	0
10	0.00000	0.00000	0
11	0.00000	0.00000	0

Buttons: OK, Read from file, Cancel.

The starting point of the program contains the usual summary:

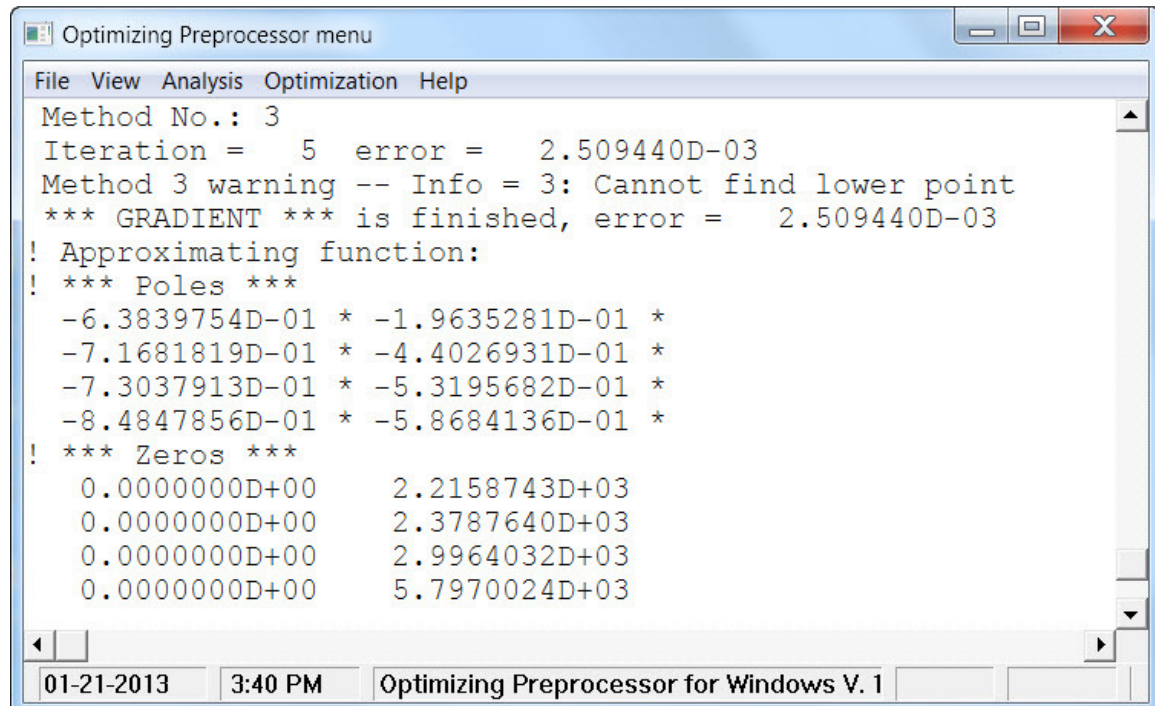


```

Optimizing Preprocessor menu
File View Analysis Optimization Help
**** Optimizing Preprocessor ****
21-jan-2013 15:27
<Optimizing filter>
  Number of points specified = 35
  Starting point = 100.00000 Hz
  End point = 3.00000 KHz
  Loss and delay requirements
  # of loss points specified = 35
  # of delay points specified = 26
  Overall delay weight = 0.000000E+00
  Digital function is specified
  Sampling frequency = 20.00000 KHz
! Approximating function:
! *** Poles ***
-7.2360048D-01 * -1.6487274D-01 *
-7.5201968D-01 * -4.1553455D-01 *
-7.7671092D-01 * -5.3864984D-01 *
-7.9323375D-01 * -5.8647113D-01 *
! *** Zeros ***
0.0000000D+00 2.2158743D+03
0.0000000D+00 2.3787640D+03
0.0000000D+00 2.9964032D+03
0.0000000D+00 5.7970024D+03
01-21-2013 3:37 PM Optimizing Preprocessor for Windows V. 1.5

```

Using the three optimization methods a few times we reach the following point, which is probably not the exact optimum, but is close to it:

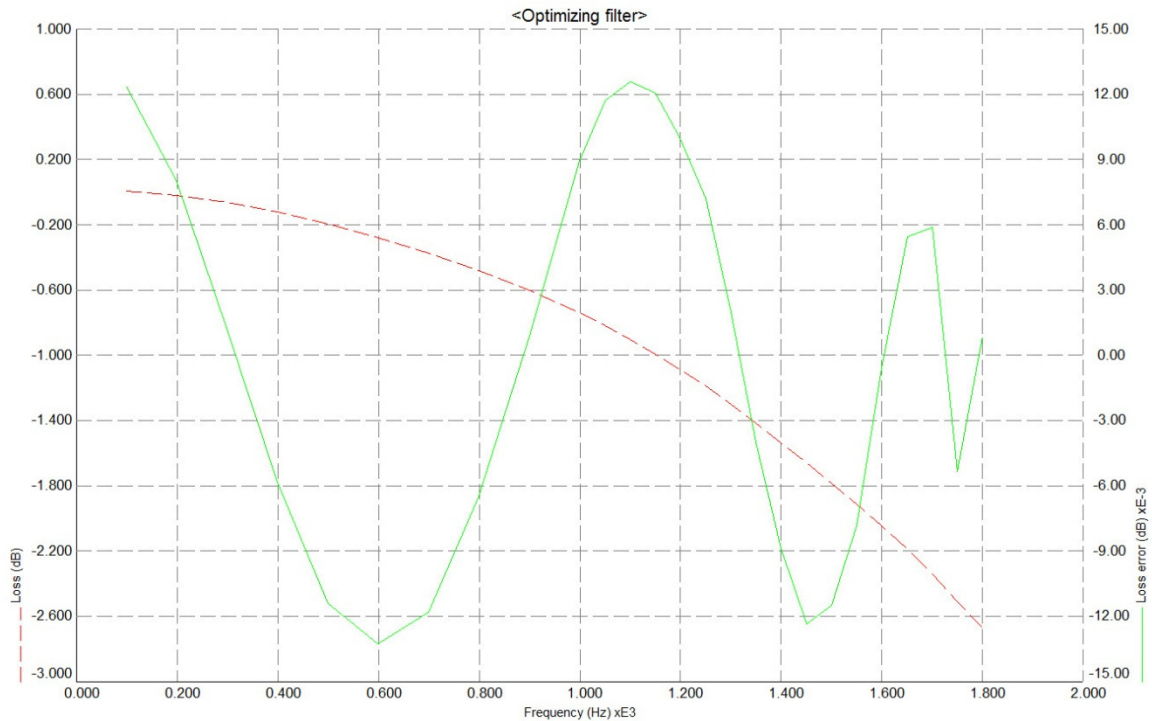


```

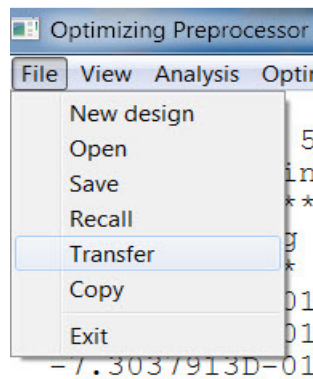
Optimizing Preprocessor menu
File View Analysis Optimization Help
Method No.: 3
Iteration = 5 error = 2.509440D-03
Method 3 warning -- Info = 3: Cannot find lower point
*** GRADIENT *** is finished, error = 2.509440D-03
! Approximating function:
! *** Poles ***
-6.3839754D-01 * -1.9635281D-01 *
-7.1681819D-01 * -4.4026931D-01 *
-7.3037913D-01 * -5.3195682D-01 *
-8.4847856D-01 * -5.8684136D-01 *
! *** Zeros ***
0.0000000D+00 2.2158743D+03
0.0000000D+00 2.3787640D+03
0.0000000D+00 2.9964032D+03
0.0000000D+00 5.7970024D+03
01-21-2013 3:40 PM Optimizing Preprocessor for Windows V. 1

```

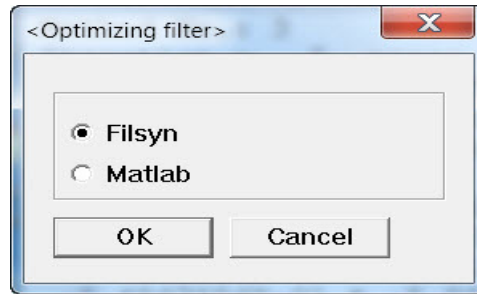
This seems to be about as good as we can expect, indicated by the plot of the loss and the loss error:



We stopped the process and wrote the results to a file called *result.pz*, as usual:



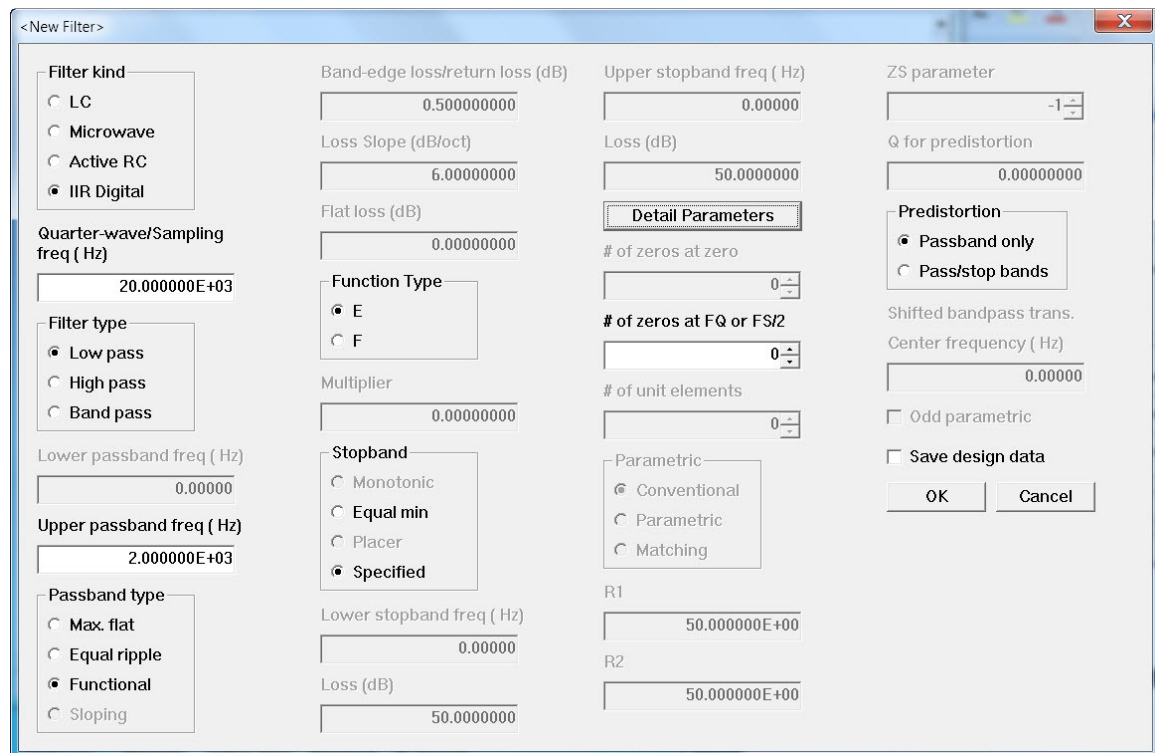
which leads to the selection of the output format:



Note that the preprocessor now writes the pole data in normalized units while the zeros are written in Hz as needed in the resulting file:

```
!<Optimizing filter>
! Approximating function:
! *** Poles ***
-6.3839754D-01    -1.9635281D-01
-7.1681819D-01    -4.4026931D-01
-7.3037913D-01    -5.3195682D-01
-8.4847856D-01    -5.8684136D-01
! *** Zeros ***
0.0000000D+00     2.2158743D+03
0.0000000D+00     2.3787640D+03
0.0000000D+00     2.9964032D+03
0.0000000D+00     5.7970024D+03
```

Finally, we proceed to implement the corresponding digital filter using the functional input option of the **Filsyn** program. The starting point is:

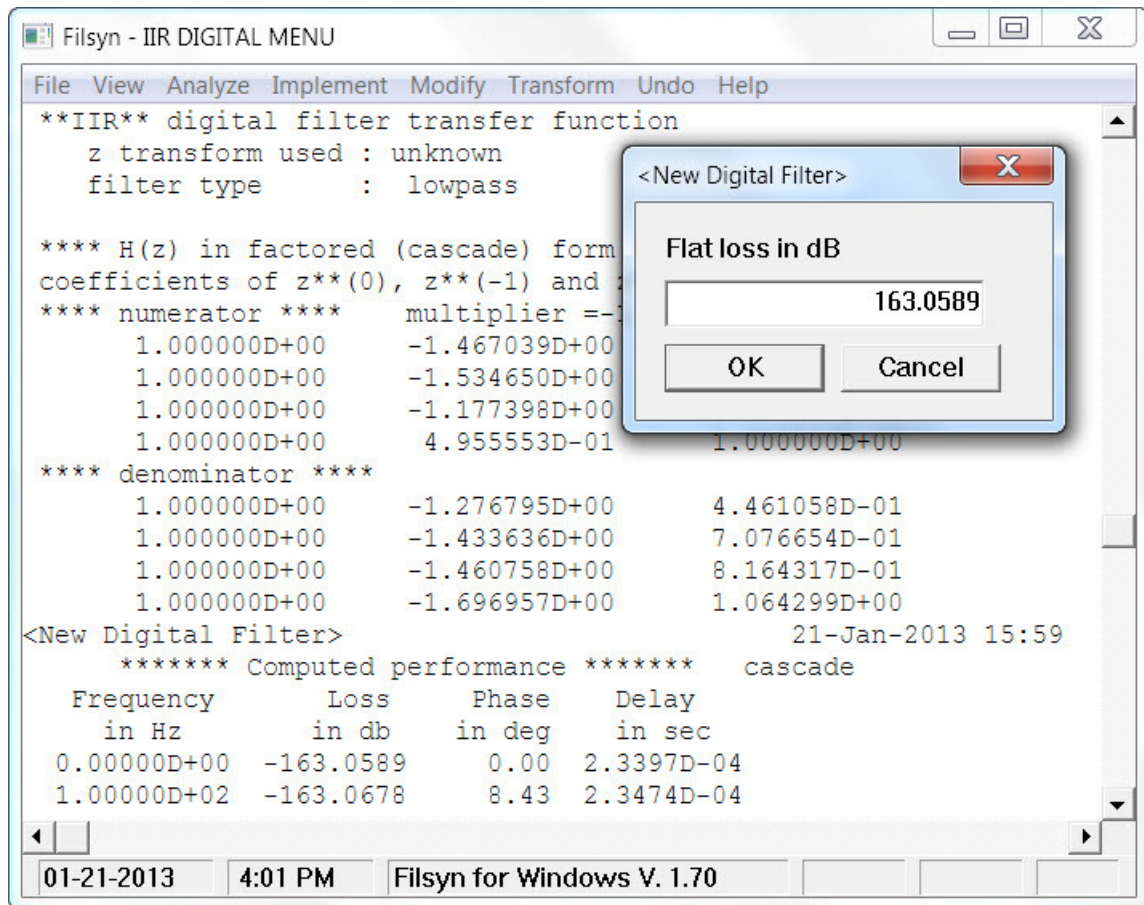


Clicking on the **Detail Parameters** button and specifying the file shown above, the pole-zero data have been read in correctly:

The dialog box titled "Filter Parameters" contains two tables. The left table, "Function zeros (normalized)", has columns for "Real Part" and "Imaginary Part". The right table, "Transmission zeros (in Hz)", also has columns for "Real Part" and "Imaginary Part". Both tables have 11 rows. The bottom of the dialog features three buttons: "OK", "Read from file", and "Cancel".

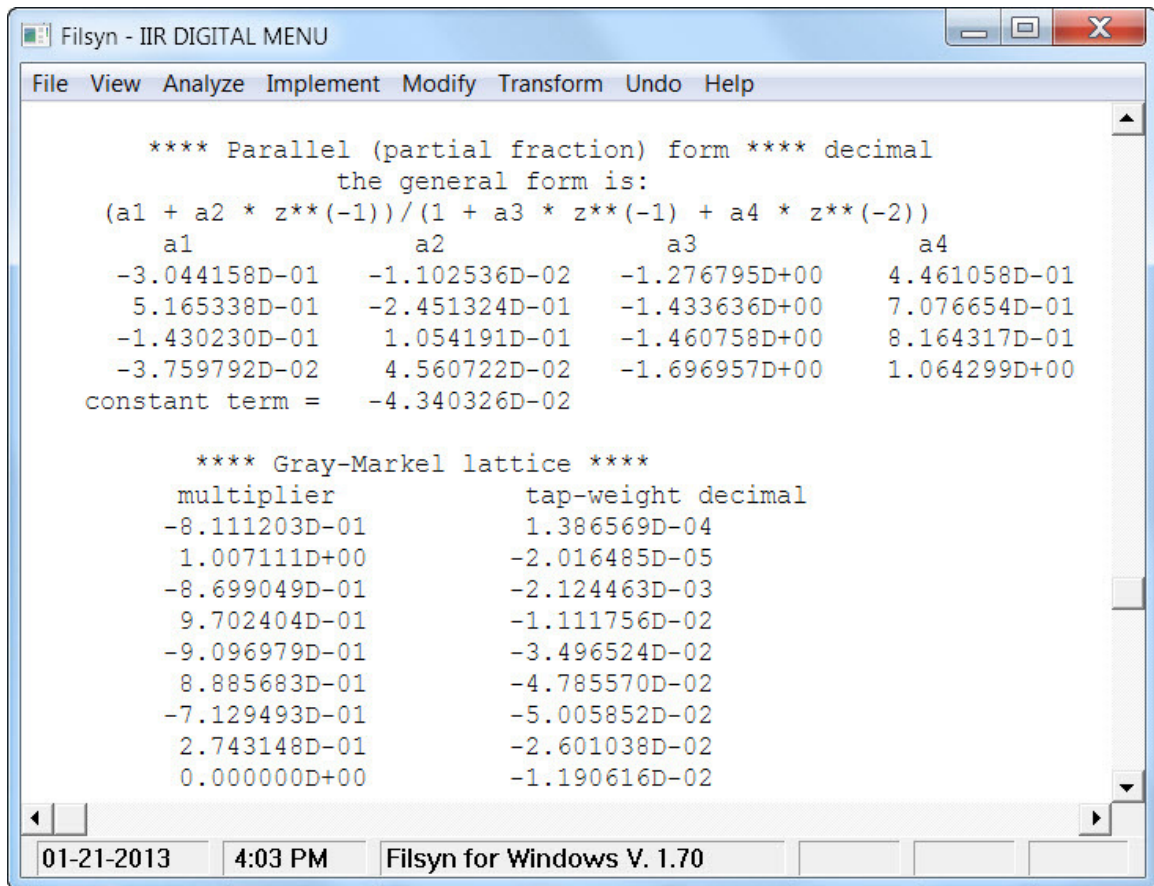
Function zeros (normalized)			Transmission zeros (in Hz)		
	Real Part	Imaginary Part		Real Part	Imaginary Part
1	-638.397540E-03	-196.352810E-03	1	0.00000	2.215874E+03
2	-716.818190E-03	-440.269310E-03	2	0.00000	2.378764E+03
3	-730.379130E-03	-531.956820E-03	3	0.00000	2.996403E+03
4	-848.478560E-03	-586.841360E-03	4	0.00000	5.797002E+03
5	0.00000	0.00000	5	0.00000	0.00000
6	0.00000	0.00000	6	0.00000	0.00000
7	0.00000	0.00000	7	0.00000	0.00000
8	0.00000	0.00000	8	0.00000	0.00000
9	0.00000	0.00000	9	0.00000	0.00000
10	0.00000	0.00000	10	0.00000	0.00000
11	0.00000	0.00000	11	0.00000	0.00000

The resulting filter is shown next that shows a missing multiplier, yielding a large flat loss – gain in this particular case:

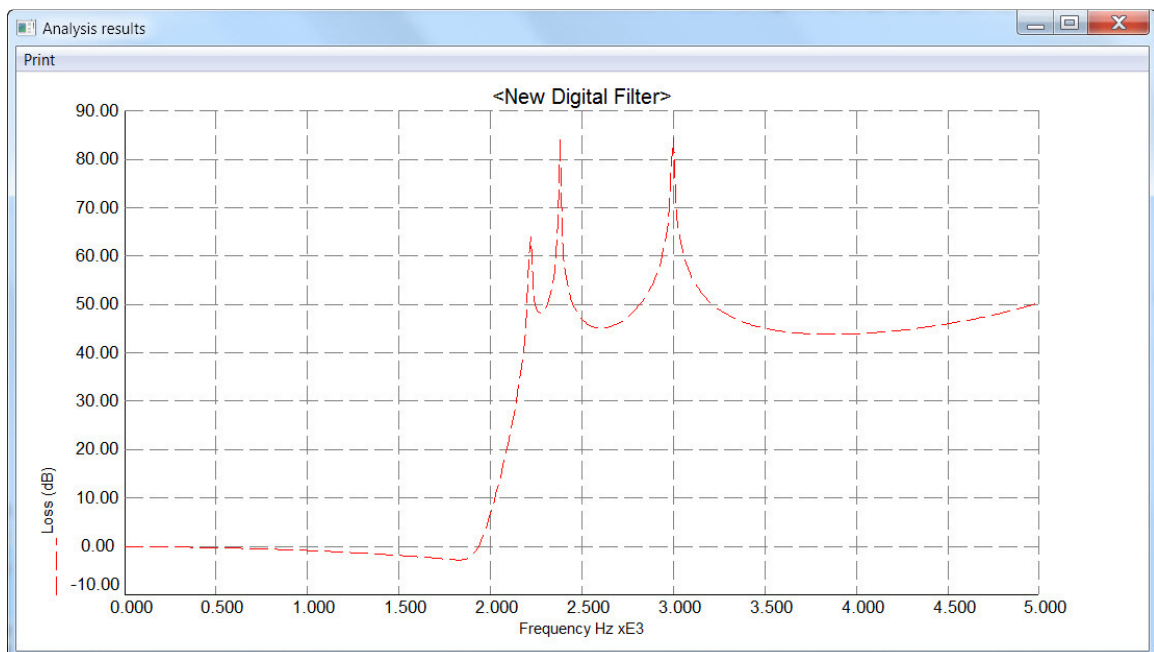


Before we do anything else, we must adjust this multiplier, by analyzing this function at a few frequencies in the passband. We see a gain of 163.0589 dB at zero frequency, so adding a flat loss using the **Transform->Flat loss** option will make the loss positive at the origin. If we need to set the gain factor such that the loss is always non-negative, we can do that using the **Transform->Scale** menu item.

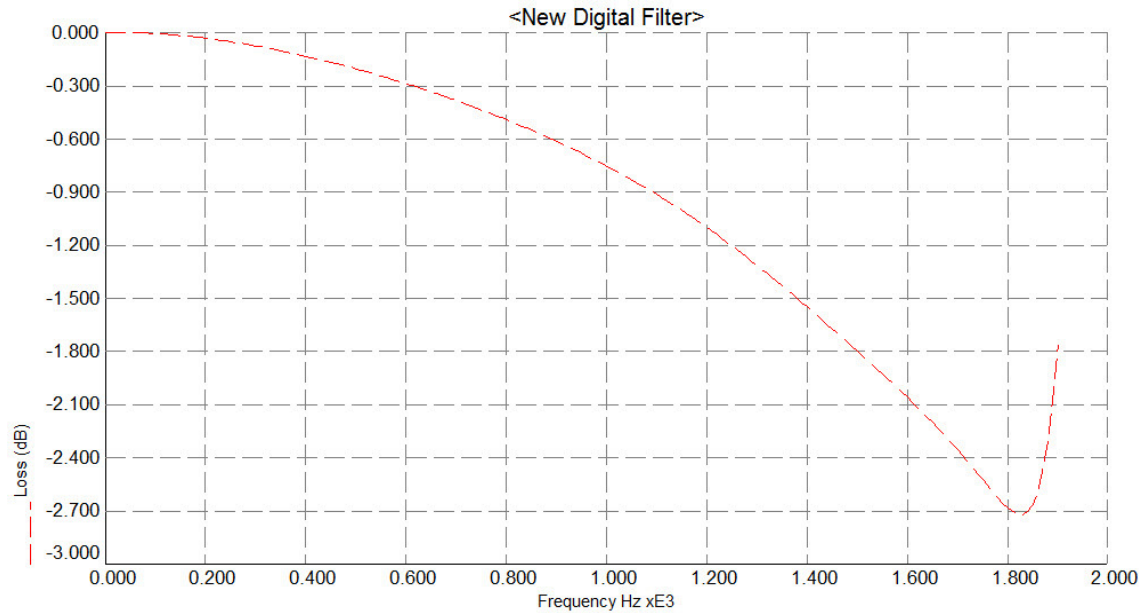
Having done that, we can implement the filter in parallel and/or Gray-Markel lattice if we desire:



Finally, analyzing the filter yields the following:



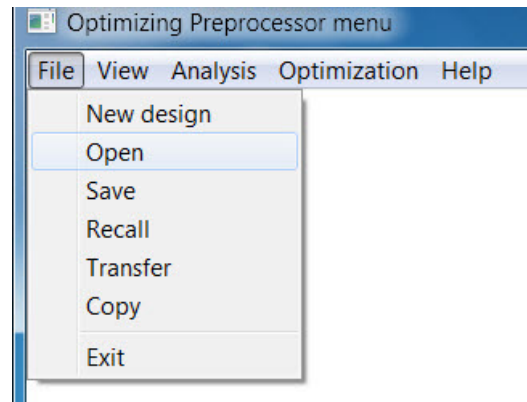
While the stopband performance could be a bit better, the passband details are indeed very good:



This analysis shows an excellent agreement. The stopband is close to equal minima and the passband approximation is very smooth, with no sharp spikes in the transition band.

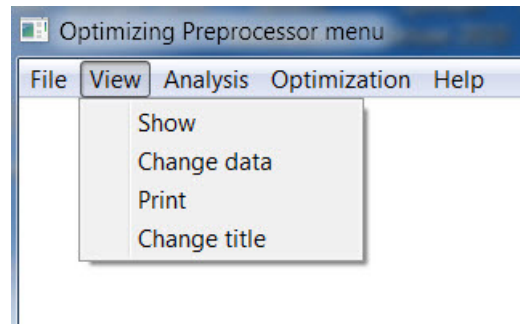
15.8 Additional features

Looking at the **File** category of menu items:



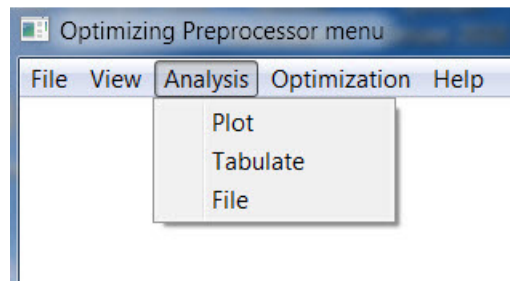
The **New design** and **Exit** are self explanatory. **Open** opens a text file for reading, but not editing. **Save** is used to save the design being performed in a file that can subsequently **Recalled** for further work. The file containing the design will have a ".qfo" extension. **Transfer** is used as we have seen above, is to transfer the resulting transfer function we have been working on to either the **Filsyn** program or a **Matlab** program. Finally, **Copy** copies what's on the screen to the clipboard.

Under the View menu we have the following submenu items:



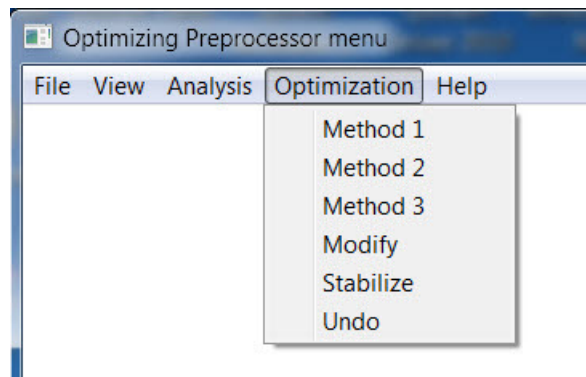
Show just prints the latest set of singularities to the screen again while **Change data** lets us change some of the pole-zero values of the function we are optimizing. **Print** can send whatever is printed to the screen also to the printer or to a text file and **Change title** is again self explanatory.

Under the **Analysis** header we have:



As you must have noticed before, the analysis request does not ask you to specify the frequencies. That is because the design is based on a set of tabulated data that specifies the frequencies of interest and is used for the analysis which is performed when we use the **Plot** submenu the plot being the most useful display of the data we are interested in. However we can display the tabulated result on the screen or send it to a text file using the other submenu items.

Finally the **Optimization** header gives us the most significant submenu items:



We have in fact used all of these submenu items except the **Modify**, which allows us to change the parameters used for the delay optimization.

15.9 Summary

This optimizing preprocessor now fills the gap of offering passband shapes other than the usual flat ones. Based on the transfer function poles and zeros, the iterative procedure is much more rugged, reliable and faster, than working with element values would have been.

However, as all iterative nonlinear optimization procedures, this one seems to have slight quirks and while we have always been able to obtain satisfactory results so far, no guarantees are offered that it will always work (as **Filsyn** will) or that if it does, it does indeed find the global optimum.

16. MACRO FILES

We have added macro capability to the passive LC/microwave analysis segment of the program. This is under the Macro menu option and has three submenu items: **Start**, **Stop** and **Play**. The **Start** option starts recording your actions in the macro file with the name you specified, while the **Stop** option will stop the recording and closes the macro file. The last **Play** option will replay the recorded steps.

The macro files are *WinBatch* files with ".wbt" extensions and need the *Winbatch.exe* program to run. These are in fact, batch (script) files, except that they are not compiled like the *.wbc files distributed with the program. They are simple text files which can be edited in any text editor or using the *Winbatch Studio* program. The *.wbc files are compiled, cannot be edited and do *not* need the *Winbatch.exe* program to run.

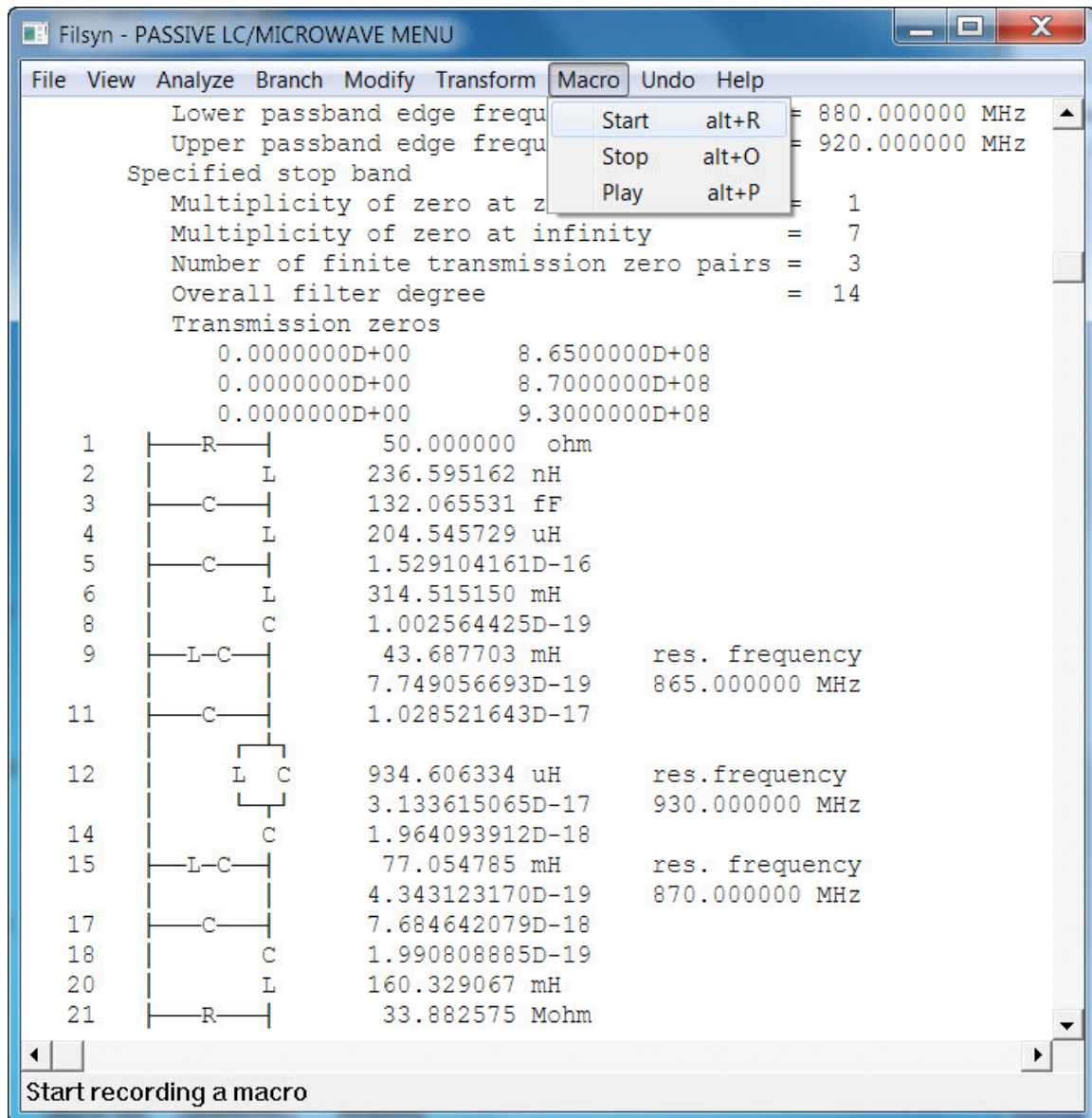
These files record all actions you are performing on a circuit, that change that circuit, except the delay equalizer (**Transform**->**Equalize**). Actions which do not change the circuit (i.e. analysis, etc.) are *not* recorded. Furthermore, if an action is not performed due to a mistake by the user, it is not recorded either in most cases. The structure of these files is fairly simple and once you record a few of them and look at what they contain, you will quickly learn how to construct them yourself. The only somewhat unusual fact is that the symbol "~" represent clicking the Enter key.

Since the operation of the macro **Play** menu option needs the *WinBatch* executable, this will also enable our users to generate their own batch files, which then can be used as macros as well. As an interesting aside, these *.wbt files can be used in two ways. One is to call them from inside **Filsyn** as macros, or alternatively, while **Filsyn** is running, you can find these macro files and double click on the one you need and it will also run like a macro.

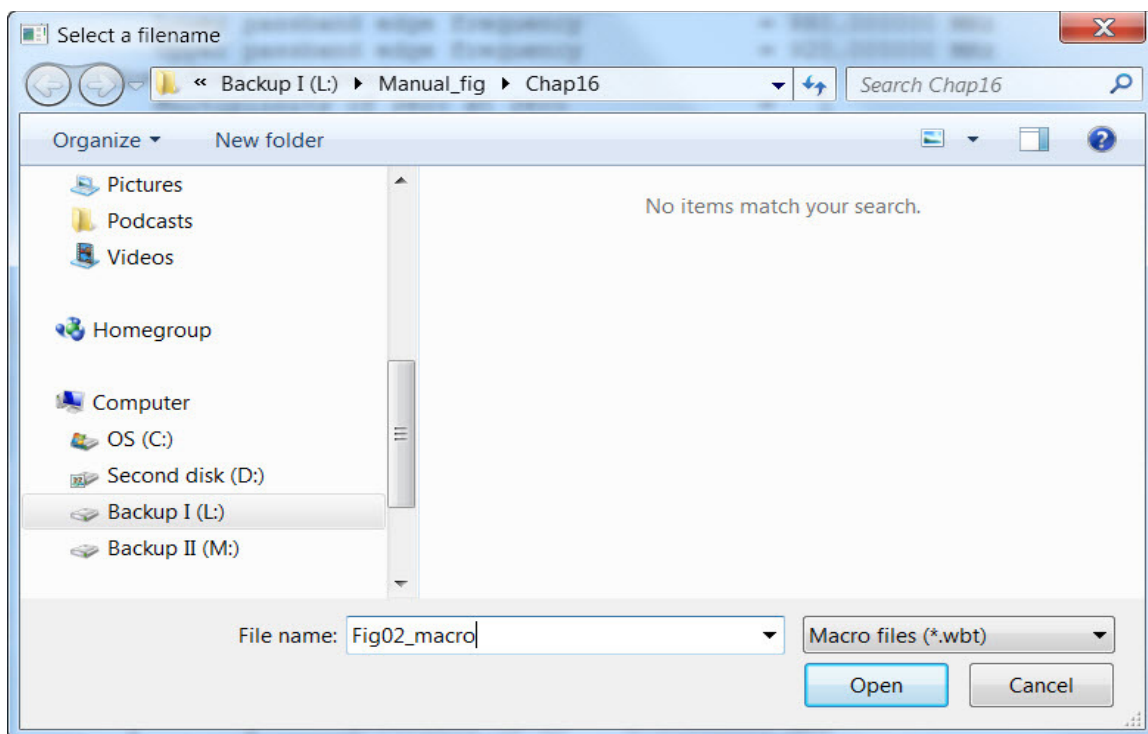
Please note however, that we do NOT provide a copy of the Winbatch executable, you will have to contact Wilson WindowWare, Inc., (<http://www.winbatch.com/>) and purchase a copy of the program.

Example

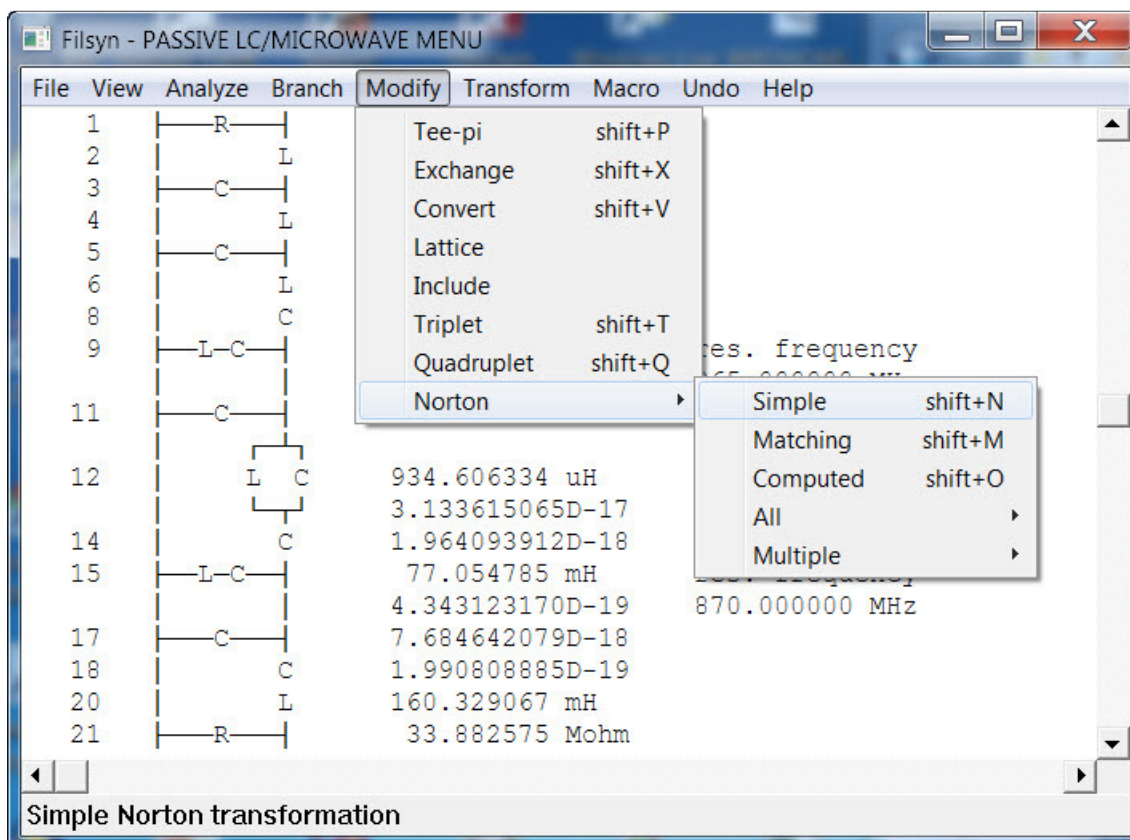
Consider a high frequency bandpass filter with the following computer-generated configuration:

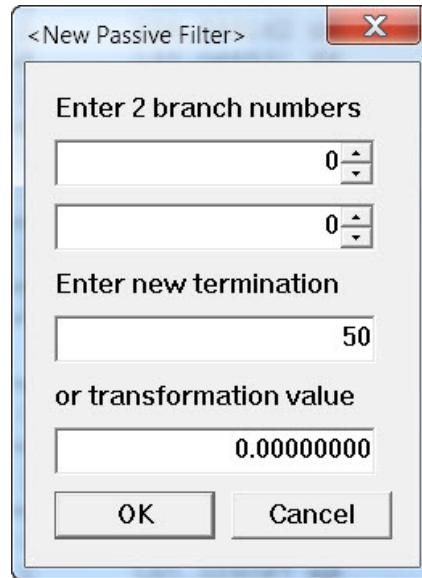


We shall record whatever we do next in a macro file called *test_bp* as shown here, using the **Macro->Start** menu option:

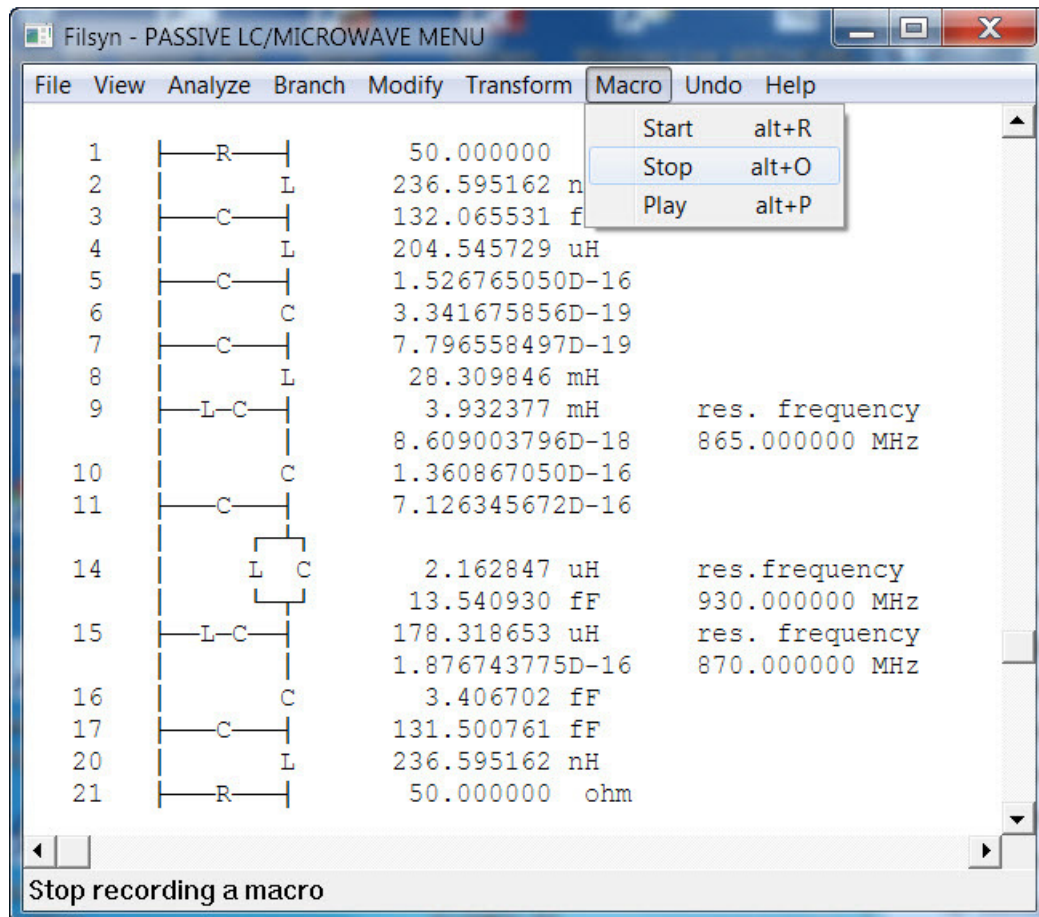


Next we perform a Norton transformation step (actually several steps as needed) to get 50 ohm output terminations:

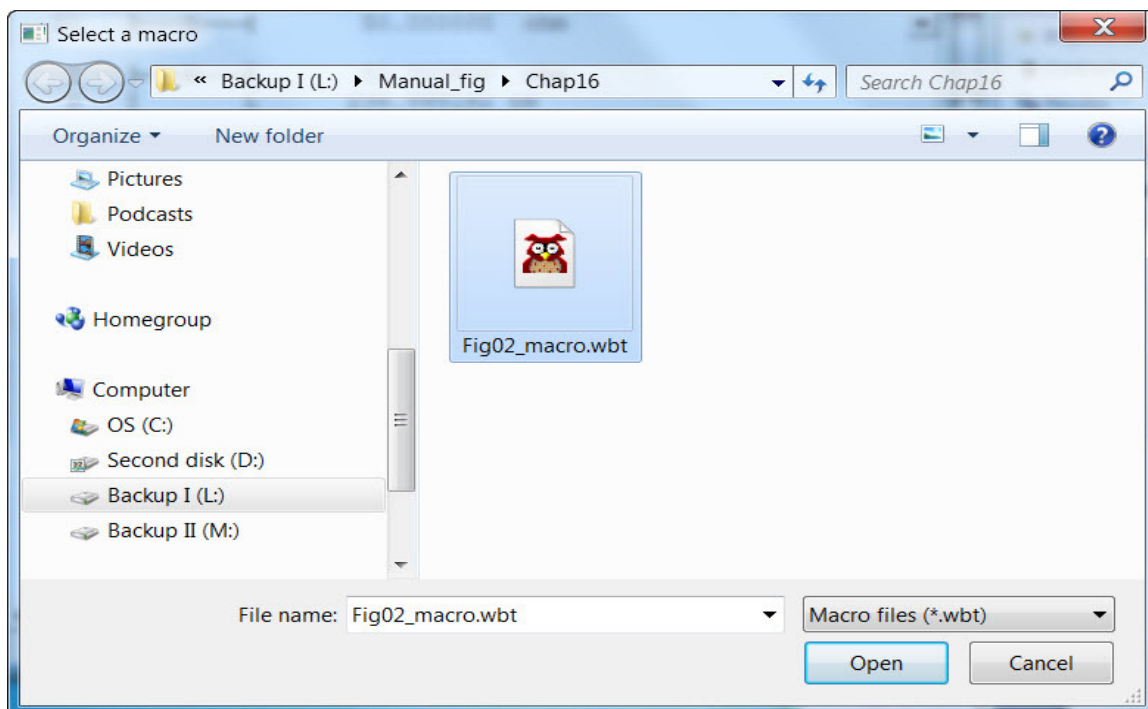
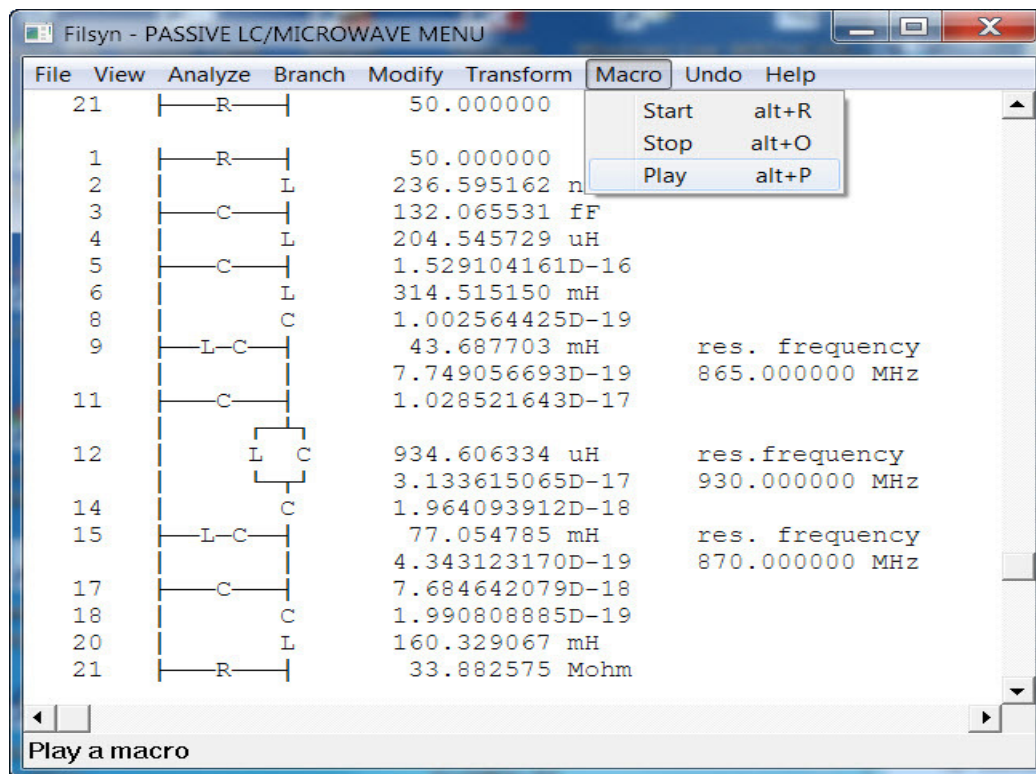




You do not need to specify any branch number, the program will find some appropriate ones. The resulting structure is now:



Next we stop the macro recording using the **Macro->Stop** menu item and then undo the transformation using the **Undo->Undo** menu item. This gets us the original circuit back. Finally we run the macro we have just recorded using the **Macro->Play** option and selecting the macro file:



As indicated above, the macro files are text files and we show the first few lines of the macro we just created to demonstrate:

```
WinWaitExist ("Filsyn - PASSIVE", 2)
SendMenuTo ("Filsyn", "View Suppress")
SendKey ("UP ~")
IntControl (35, 100, 0, 0, 0)
WinWaitExist("Filsyn - PASSIVE", 2)|
SendMenuTo("Filsyn", "Modify Norton Simple")
WinWaitExist ("<New Passive", 2)
SendKey(" 0 {TAB} 0 {TAB} 5.000000D+01 {TAB} 0.000000D+00 ~")
```

The macro first turns printing off, then performs the operations (the Norton transformation step) then turns printing back on and displays the same circuit which is the one we had above on page 353.

Usually we would not use the macro for such a simple job. Its main use would be to record a large series of steps that we expect to repeat on other designs in the future. As such, the process is extremely useful for developing script files for circuits that have no such script files as yet.

APPENDIX A**REFERENCES**

- [1] A. Antoniou: Digital Filters: Analysis and Design. McGraw Hill, N.Y. 1979
- [2] H.J. Blinchikoff, A.I. Zverev: Filtering in the Time and Frequency Domains. Wiley, N.Y. 1976
- [3] P.O. Brackett, A.S. Sedra: Direct SFG Simulation of LC Ladder Networks with Applications to Active Filter Design. IEEE Trans. on Circuits & Systems, Vol. CAS-23 Feb. 1976 pp. 61-67
- [4] L.T. Bruton: RC-Active Circuits, Theory and Design. Prentice Hall, N.J. 1980
- [5] E. Christian, E. Eisenmann: Filter Design Tables and Graphs. Transmission Networks Internat., N.C. 1977
- [6] E. Christian: LC Filters: Design Testing & Manufacturing. Wiley, N.Y. 1983
- [7] R.W. Daniels: Approximation Methods for Electronic Filter Design. McGraw Hill, N.Y. 1974
- [8] S. Darlington: Synthesis of Reactance Fourpoles which Produce Prescribed Insertion Loss Characteristics. J. Math. Phys., Vol. 18 1939 pp. 257-353
- [9] G. Daryanani: Principles of Active Network Synthesis and Design. J. Wiley, N.Y. 1976
- [10] A. Fettweis: Digital Filter Structures Related to Classical Filter Networks. AEU Vol. 25 Feb. 1971 pp. 79-89
- [11] A. Fettweis: A Simple Design of Maximally Flat Delay Digital Filters. IEEE Trans. Audio Electroacoust. Vol. AU-20 1972 pp 112-114
- [12] A. Fettweis: Wave Digital Filters: Theory and Practice. Proc. of the IEEE, Vol. 74, No. 2 February 1986, pp.270-327
- [13] I.M. Filanovsky: A Generalization of Filters with Monotonic Amplitude-Frequency Response. IEEE Trans. Circuits & Systems, Part I. Vol. 46 Nov. 1999 pp 1382-1385
- [14] P.E. Fleischer, J. Tow: Design Formulas for Biquad Active Filters Using Three Operational Amplifiers. Proc. IEEE Vol. 61 May 1971 pp. 662-663
- [15] P.E. Fleischer, K.R. Laker: A Family of Active Switched Capacitor Biquad Building Blocks. Bell System Tech. Jour., Vol. 58 pp. 2235-2269, Dec. 1979

References

- [16] T.G. Foxall, A.I. Ibrahim, G.J. Hupe: Minimum Phase CCD Transversal Filters. IEEE Jour. Solid State Circuits Vol. SC-12 Dec. 1977 pp. 638-642
- [17] M. Fukado: Optimum Filters of Even Order with Monotonic Responses. IRE Trans. Circuit Theory Vol CT-6 1959 pp 277-281
- [18] J. Gensel: On the Design of RC-Active Follow-the-Leader Feedback Filters. Int. J. Circuit Theory & Applic. Vol. 5 1977 pp. 361-365
- [19] M.S. Ghausi, K.R. Laker: Modern Filter Design. Prentice Hall, N.J. 1981
- [20] F.J. Harris: On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform. Proc. IEEE Vol. 66 Jan. 1978 pp. 51-83
- [21] W.E. Heinlein, W.H. Holmes: Active Filters for Integrated Circuits. R. Oldenbourg, Munich, Germany 1974
- [22] O. Herrmann, W. Schuessler: Design of Non-Recursive Digital Filters with Minimum Phase. Electr. Letters Vol. 6 May 28, 1970 pp. 329-330
- [23] D. M. Himmelblau: Applied Nonlinear Programming. McGraw Hill Book Co., NY 1972 Chapter 4
- [24] D.S. Humpherys: The Analysis, Design and Synthesis of Electrical Filters. Prentice Hall, N.J. 1970
- [25] D. Kahaner, C. Moler, S. Nash: Numerical Methods and Software. Prentice Hall, NJ 1989 Chapter 9.
- [26] C. Kurth: Frequenzweichen mit Allsperrbereich, Frequenz, Vol. 17 March-May 1963 pp. 113-122, 158-164 and 189-202.
- [27] S. Lawson, A. Mirzai: Wave Digital Filters. Ellis Horwood Ltd., 1990 England
- [28] P. Leistner, T.W. Parks: On the Design of FIR Digital Filters with Optimum Magnitude and Minimum Phase. AEU Vol. 29 June 1975 pp. 270-274
- [29] R. Levy: A General Equivalent Circuit Transformation for Distributed Networks. IEEE Trans. Circuit Theory Vol. CT-12 Sep. 1965 pp. 457-458
- [30] G.L. Matthaei, L. Young, E.M.T. Jones: Microwave Filters, Impedance Matching Networks and Coupling Structures, McGraw Hill Book Co., N.Y. 1964 p. 221 ff.
- [31] J.H. McClellan, T.W. Parks, L.R. Rabiner: A Computer Program for Designing Optimum FIR Linear Phase Digital Filters. IEEE Trans. Audio & Electroacoustics, Vol. AU-21 Dec. 1973 pp. 506-526

- [32] B. J. Minnis: Classes of Sub-Miniature Microwave Printed Circuit Filters with Arbitrary Passband and Stopband Widths. IEEE Trans. on Microwave Theory and Techniques, Vol. 30 Nov. 1982 pp. 1893-1900
- [33] G.S. Moschytz, P. Horn: Active Filter Design Handbook. Wiley, Chichester, England 1981
- [34] E. L. Norton: Constant Resistance Networks with Application to Filter Groups, Bell System Tech. J., Vol. 16 1937 pp. 178-193
- [35] H. Ozaki, J. Ishii: Synthesis of a Class of Strip-line Filters. IEEE Trans. Circuit Theory Vol. CT-5 June 1958 pp. 104-109
- [36] A. Papoulis: A New Class of Filters. Proc. IRE Vol. 46 1958 pp. 649-653
- [37] A. Peled, B. Liu: Digital Signal Processing. Wiley, N.Y. 1976
- [38] M. D. J. Powell: An Efficient Method for Finding the Minimum of a Function of Several Variables Without Calculating Derivatives. Computer Journal, Vol. 7 1965 pp. 303-307
- [39] L.R. Rabiner: Approximate Design Relationships for Lowpass FIR Digital Filters. IEEE Trans. Audio & Electroacoustics Vol. AU-21 Oct. 1973 pp. 456-460
- [40] L.R. Rabiner, B. Gold: Theory and Application of Digital Signal Processing. Prentice Hall, N.J. 1975
- [41] J.D. Rhodes: Theory of Electrical Filters. J. Wiley & Sons, London 1976
- [42] R. Saal: Handbook of Filter Design. AEG-Telefunken, Berlin, Germany 1979
- [43] A. Sedra, P.O. Brackett: Filter Theory and Design. Matrix, Champaign, IL 1978
- [44] J.K. Skwirzynski: Design Theory and Data for Electrical Filters. Van Nostrand, London, England 1967
- [45] G. Szentirmai: On the Realization of Crystal Bandpass Filters. IEEE Trans. Circuit Theory Vol. CT-11 June 1964 pp. 299-301
- [46] G. Szentirmai (Ed.): Computer Aided Filter Design. IEEE Press, N.Y. 1973
- [47] G. Szentirmai: Synthesis of Multiple Feedback Active Filters. Bell System T. J. Vol. 52 Apr. 1973 pp. 527-555
- [48] G. Szentirmai: Crystal and Ceramic Filters. Chap 4 in G.C. Temes, S.K. Mitra: Modern Filter Theory and Design. J. Wiley, N.Y. 1973

References

- [49] G. Szentirmai: FILSYN - A General Purpose Filter Synthesis Program. Proc. IEEE Vol. 65 Oct. 1977 pp. 1443-1458
- [50] G. Szentirmai: Interactive Filter Design by Computer, Circuits & Systems, Oct. 1978 pp. 1-13 and Dec. 1978 pp. 1-8.
- [51] G. Szentirmai: Diplexer Design is Easy with CAD, Microwaves, Vol. 19 April 1980 pp. 60-64.
- [52] G. Szentirmai: Filter Approximation Methods in J.G. Webster, Ed.: Wiley Encyclopedia of Electrical and Electronics Engineering, J. Wiley, N.Y. 1999 Vol. 7 pp. 449-480
- [53] G. Szentirmai: Two-Port Equivalences for Bandpass Filters, IEEE Trans. on Circuits and Systems, Part I, Vol. 47 pp. 1431-1437, Sept. 2000
- [54] G.C. Temes, S.K. Mitra: Modern Filter Theory and Design. Wiley, N.Y. 1973
- [55] J.P. Thiran: Recursive Digital Filters with Maximally Flat Group Delay, IEEE Trans. Circuit Theory Vol. CT-18, 1971 pp. 659-664
- [56] J. Tow: Design and Evaluation of Shifted Companion Form Active Filters. Bell System T. J. Vol. 54 March 1975 pp. 545-568
- [57] E. Ulbrich, R. Piloty: On the Design of Allpass, Lowpass and Bandpass Filters with Constant Group Delay of the Chebyshev Type. AEU Vol. 14 Oct. 1960 pp. 451-467
- [58] H. Watanabe: Approximation Theory for Filter Networks. IRE Trans. Circuit Theory Vol. CT-9 Sep. 1961 pp. 341-356
- [59] A.I. Zverev: Handbook of Filter Synthesis. Wiley, N.Y. 1967
- [60] ---: Programs for Digital Signal Processing. IEEE Press N.Y. 1979
- [61] R. Levy: Characteristics and Element Values of Equally Terminated Achieser-Zolotarev Quasi-Low-Pass Filters. IEEE Trans. On Circuit Theory, Vol. CT-18, No. 5, Sept. 1971 pp. 538-544
- [62] W. Cauer: Theorie der Linearen Wechselstromschaltungen. Akademie-Verlag, Berlin, 2nd Ed. 1954 pp. 386-7
- [63] T. A. Abele: Transmission Line Filters Approximating a Constant Delay in a Maximally Flat Sense. IEEE Trans. Circuit Theory Vol. CT-14 Sep. 1967 pp. 298-306
- [64] C.J. Wellekens, A.N. Godard: Simultaneous Flat Approximations of the Ideal Low-Pass Attenuation and Delay for Recursive Digital, Distributed, and Lumped Filters. IEEE Trans. On Circuits and Systems, Vol. CAS-24, No. 5, May 1977 pp. 221-230

APPENDIX B

MANUAL SYNTHESIS

B.1 Introduction

One of the greatest strengths of the **Filsyn** program is the generality of the available passive or microwave structures. This generality leads to a certain amount of complexity in specifying a ladder structure. The computer-generated configuration was made available primarily to alleviate this problem. Most users will find the computer-selected structure satisfactory. This is especially true, since the available commands allow us to change any ladder structure into any other.

For those who need different or more general structures, the program provides further facilities in the form of manual synthesis.

B.2 Outline of the ladder synthesis procedure

To explain the method clearly, we must spend some time in outlining the general ladder synthesis procedure. The synthesis of a given transfer function proceeds as follows:

- a)** One must select (and preferably sketch) the required configuration and confirm that it is consistent with the transfer function.
- b)** One must select the appropriate immittance function(s) to be used for the synthesis.
- c)** One must synthesize the required structure (calculate the element values) twice: once in the forward direction and again in the reverse direction. Only one synthesis is possible if extreme termination is specified; we may be able to get by with one synthesis in some lowpass or highpass filters.
- d)** Finally, by comparing the results of these two syntheses, we must recover possibly missing elements, calculate terminations, and check the numerical precision achieved in the process. This step can be avoided if no elements are missing, the terminations are known and we are certain of the accuracy of the results.

Understanding step **a)** comes with experience and familiarity with filter theory, which we have no space to go into here. Selecting the appropriate immittance functions, step **b)**, is also an involved process and help, in the form of a computer-selection, is available. However, for those requiring complete generality, we *have* the option of selecting the immittance functions.

To understand this process however, we must outline the method used to calculate the values of the individual elements. This is the well-known “zero-shifting” method but with a number of additional features. We take one of the open- or short-circuited immittance functions and, fully or partially, remove those poles of this immittance, which coincide with one of the transmission zeros of the transfer function. If the pole is fully removed (i.e. the whole term in the partial fraction expansion is subtracted completely), the transmission zero is fully realized and is removed from the list of zeros still to be

realized. If the pole is only partially removed (i.e. only part of the term is subtracted), the transmission zero is *not* fully realized it remains on the list and will have to be fully realized later. Note that in this version of the program, only poles at extreme frequencies can be removed partially. Those at finite frequencies are always fully removed. This was done in order to simplify the procedure and also because the partial removal of finite frequency poles has never known to have been used.

However, immittance function poles hardly ever coincide with transmission zeros, except if we force them to. One can initially force this to happen only at zero and/or infinite frequencies and this is why a passive filter *must* have at least a single transmission zero at zero and/or infinite frequencies to be realizable in ladder form without tightly coupled inductors. Later on, this coincidence will be forced by partial removal of an already coincident pole-transmission zero pair. Such a partial removal will subsequently cause a shift in the locations of the zeros of the remainder function (they will all move towards the pole we are partly removing). If we remove just enough of the pole so that one zero of the remainder moves exactly over a transmission zero, then the reciprocal of the remainder *will* have a coincident pole-transmission zero pair. Naturally, the poles of the remainder stay unchanged.

Recalling that a realizable immittance function must have an alternating sequence of poles and zeros, and that they must also have singularities of one kind or another at zero and infinite frequencies, we have all the information we need and are ready to follow a graphical presentation of a couple of examples.

The first example (Fig. 22 below) starts with the short-circuited impedance Z_1 at the input (top) side and the first line on the left shows the pole-zero pattern (not to scale) of Z_1 , as well as the locations of the two transmission zeros F_1 and F_2 . There is an additional single transmission zero at infinity, where Z_1 itself has a pole. This pole is the one we must use at the start because this is the only coincident pole-transmission zero pair. Hence we remove a part of this pole such that the nearest zero moves up to underneath the, say, first finite frequency transmission zero. This step yields a series inductance and the inverted remainder (Y_2) now has a coincident pole-transmission zero, which can be fully removed, yielding a series resonant circuit in shunt.

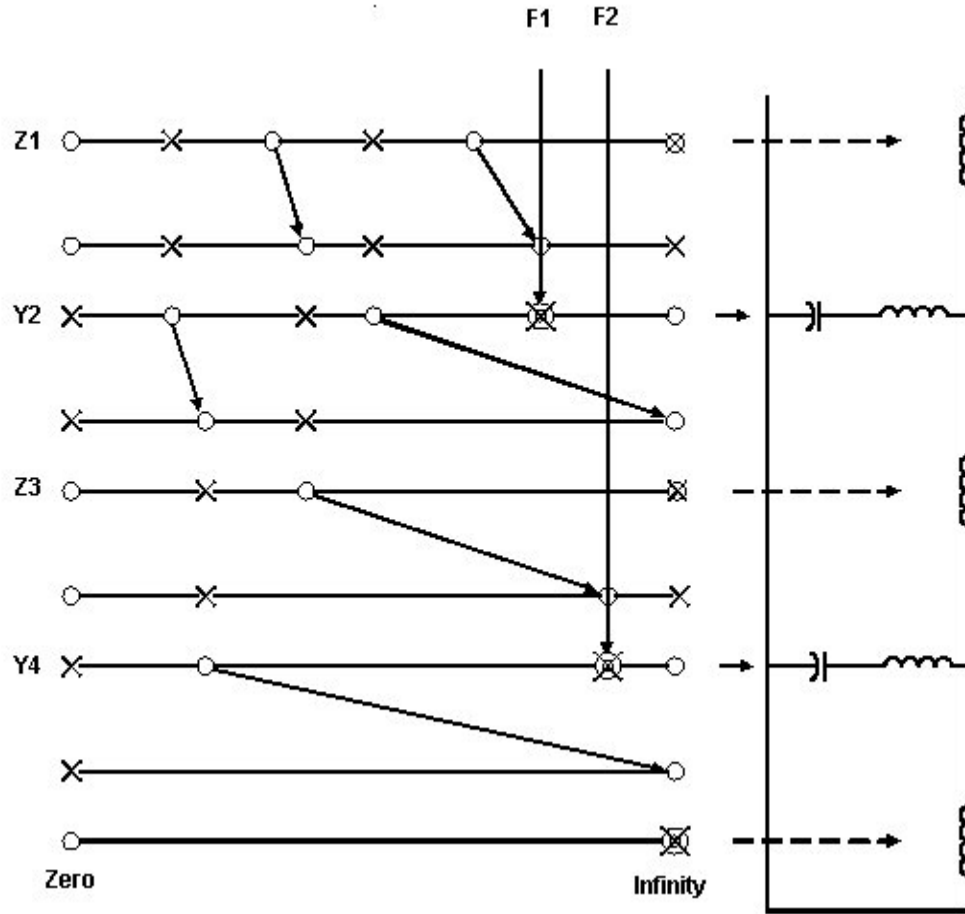


Fig. 22 Zero-shifting procedure

Inverting the remainder again, we completed a cycle and start over again. The only pole of the impedance Z_3 we can use is the one at infinity, and the situation is as it was at the beginning. Repeating this cycle yields another inductor in series followed by a resonant circuit in the shunt branch, realizing the second finite transmission zero. The last remainder is a series L, which fully realizes the transmission zero at infinity.

Using this graphical representation, the procedure seems deceptively simple and the pole-zero pattern, including the relative locations of the transmission zeros, makes it so. This pattern does show clearly what possible steps we may take next. One must realize, that this example was a lowpass and that the more general bandpass case will usually allow more choices.

In bandpass cases, a number of choices are usually available leading to a number of equivalent realizations. Some of these equivalents differ only in the values of their elements, others may look entirely different.

In the current version of the program, the procedure is somewhat simpler, in that we need not specify the finite transmission zero, we may let the program select it for us. We have

to select the zero(s) if they are either complex quadruplets or if we need to select two of them.

B.3 Immittance function selection

Consider the problem of selecting an appropriate immittance function for the synthesis. We recall that two polynomials, $E(s)$ and $F(s)$ are available to generate these immittance functions and since all immittances are odd rational functions of the frequency, we need an even and an odd polynomial. There are, in general, four such polynomials available:

$$\begin{array}{ll} \text{Even sum} & (\text{ES}): E_e(s) + F_e(s) \\ \text{Odd sum} & (\text{OS}): E_o(s) + F_o(s) \\ \text{Even difference (ED):} & E_e(s) - F_e(s) \\ \text{Odd difference (OD):} & E_o(s) - F_o(s) \end{array}$$

Some of these functions may or may not exist: therefore the program displays when a passive synthesis is about to begin, indicating which of these are present. A few general rules about the existence of these functions are as follows:

- a) Only the sum polynomials (ES and OS) exist if extreme termination is specified. In this case the selection is unique.
- b) All four polynomials exist if any one of the following conditions, or their combination, is specified:
 - i) Linear-phase lowpass,
 - ii) Predistortion,
 - iii) Odd degree parametric bandpass,
 - iv) Functional input,
 - v) $R_1 \neq R_2 \neq 0$ in a lowpass or highpass case.
- c) If none of these conditions exist, three of the four polynomials will be present.

The existing polynomials are stored for the duration of the run; hence even if one choice does not work, we may try another.

If only two polynomials are present, we have no choice. This happens when one of the terminations is extreme and only the forward synthesis can be performed.

If three polynomials are present, one of them will be of a lower degree than the other of the same parity. Usually this is the difference (ED or OD) polynomial, but not always. Normally this function is to be avoided, because some of the components of the filter will be missing if this function is used.

The remaining two polynomials will be used for both the forward and the reverse syntheses, yielding a symmetrical or an antisymmetrical network.

There *are* cases however, when this lower degree polynomial is useful, especially very high order filter functions (order 25 and above). Numerical problems may be encountered in such cases, and it was experimentally found, that the lower-degree functions yield substantially more accurate elements in higher-order cases than the others. The missing elements must then be obtained by manual comparison of the forward and reverse syntheses. It is advisable to use the computer-selected immittance, at least for the first time. If that is not satisfactory, we may select our own, but by then we will have a much better picture of the situation. In any case all the input data may be saved at the beginning of the design, making it much easier to repeat it if that becomes necessary.

If all four polynomials are present, one of them will again be of lower degree than the other one of the same parity. Ignoring this one we are left with three polynomials: either two even and one odd or two odd and one even. We can form *two* odd immittance functions from these three polynomials and we will use one for the forward synthesis and the other for the reverse one. Otherwise the choice is arbitrary, and the two possible alternatives lead to different, but equivalent networks. Again the computer selection is recommended at the first try.

Only the parametric bandpass cases present any difficulty. Instead of developing and explaining complicated rules, we recommend the use of the computer-selected functions. They may be replaced if they turn out to be inappropriate. A little experimentation will yield the right function and the pole-zero pattern will be found to be an invaluable tool.

Lastly, the immittance must be defined to be an impedance or an admittance. The program always expects this function as an *even/odd* ratio and we must identify this ratio as an impedance or an admittance. The computer selection is indicated by the IMP parameter, which is either T (true) or F (false). One way to select this parameter is to pick it arbitrarily and look at the pole-zero pattern or try the synthesis of a branch. If the result is incorrect, back up and start over again with the other choice.

Another method is to look at the very first branch needed, as well as the polynomial degrees. Assume for the moment, that we need a series inductor as the first branch. The impedance must then have a numerator that is of higher degree than the denominator, because the impedance must go to infinity when the frequency goes to infinity. Assume furthermore, that the overall filter degree is odd, in which case the odd polynomial is of the same degree as the filter, while the even one is one degree lower, (except in parametric bandpasses). Hence the even/odd ratio goes to zero with increasing frequency, i.e. it is of the wrong form for an impedance, and it must therefore be the admittance. All other combinations may be argued and determined in a similar way.

Finally it should be mentioned that since the program is able to find the proper immittance functions in its auto synthesis mode, we can utilize this fact. The method is to simply perform the auto synthesis and request the intermediate results. The printout we obtain this way will indicate which immittance functions were used both in the forward and the reverse syntheses and we can then use these for our manual synthesis. It is advisable to save the input data in a file so that restarting the design needs no repetition of data entry.

B.4 Manual synthesis examples

B.4.1 Bandpass with Quadruplet of Zeros

This is a reasonably complex bandpass with a complex quad and two purely imaginary (jw axis) transmission zeros, illustrating as many of the program's features as possible, including use of quadruplet of complex zeros. The specification entered into the data entry screen is shown below. The filter is an LC bandpass, with passband from 800 Hz to 1250 Hz and 0.25 dB passband loss ripple. The stopband has two zeros on the imaginary axis at 600 Hz and at 1333.33 Hz and a complex quad at $\pm 200 \text{ Hz} \pm j 1000 \text{ Hz}$ and two zeros both at zero and infinite frequencies. The frequencies are entered in the **Detail Parameters** window below.

<New Filter>

Filter kind

- ☒ LC
- ☐ Microwave
- ☐ Active RC
- ☐ IIR Digital

Quarter-wave/Sampling freq (Hz)

0.00000

Filter type

- ☐ Low pass
- ☐ High pass
- ☒ Band pass

Lower passband freq (Hz)

800.000000E+00

Upper passband freq (Hz)

1.250000E+03

Passband type

- ☐ Max. flat
- ☒ Equal ripple
- ☐ Functional
- ☐ Sloping

Band-edge loss/return loss (dB)

0.25000000

Loss Slope (dB/oct)

6.00000000

Flat loss (dB)

0.00000000

Function Type

- ☒ E
- ☐ F

Multiplier

0.00000000

Stopband

- ☐ Monotonic
- ☐ Equal min
- ☐ Placer
- ☒ Specified

Lower stopband freq (Hz)

0.00000

Loss (dB)

50.00000000

Upper stopband freq (Hz)

0.00000

Loss (dB)

50.00000000

Detail Parameters

of zeros at zero

2

of zeros at infinity

2

of unit elements

0

Parametric

- ☒ Conventional
- ☐ Parametric
- ☐ Matching

R1

50.000000E+00

R2

50.000000E+00

ZS parameter

-1

Q for predistortion

0.00000000

Predistortion

- ☒ Passband only
- ☐ Pass/stop bands

Shifted bandpass trans.

Center frequency (Hz)

0.00000

☐ Odd parametric

☐ Save design data

OK Cancel

The **Detail Parameters** is where we enter the transmission zeros:

	Real Part	Imaginary Part
1	0.00000	600.000000E+00
2	0.00000	1.333330E+03
3	200.000000E+00	1000
4	0.00000	0.00000
5	0.00000	0.00000
6	0.00000	0.00000
7	0.00000	0.00000
8	0.00000	0.00000
9	0.00000	0.00000
10	0.00000	0.00000
11	0.00000	0.00000

It turns out that the computer does not generate a configuration that can be synthesized (it generates a circuit with some negative element values), hence we are forced to use the manual synthesis method. Our starting point is the screen:

<Synthesis>

Structure

☐ Lattice

☒ Ladder

Configuration

☐ Auto

☒ Manual

Numerator

☒ Even sum

☐ Even difference

Form

☒ Microwave

☐ Wave digital

Direction

☒ Forward

☐ Reverse

Denominator

☒ Odd sum

☐ Odd difference

Format

☒ Decimal

☐ Hexadecimal

Impedance selection

☒ Auto

☐ Manual

This is an

☒ Impedance

☐ Admittance

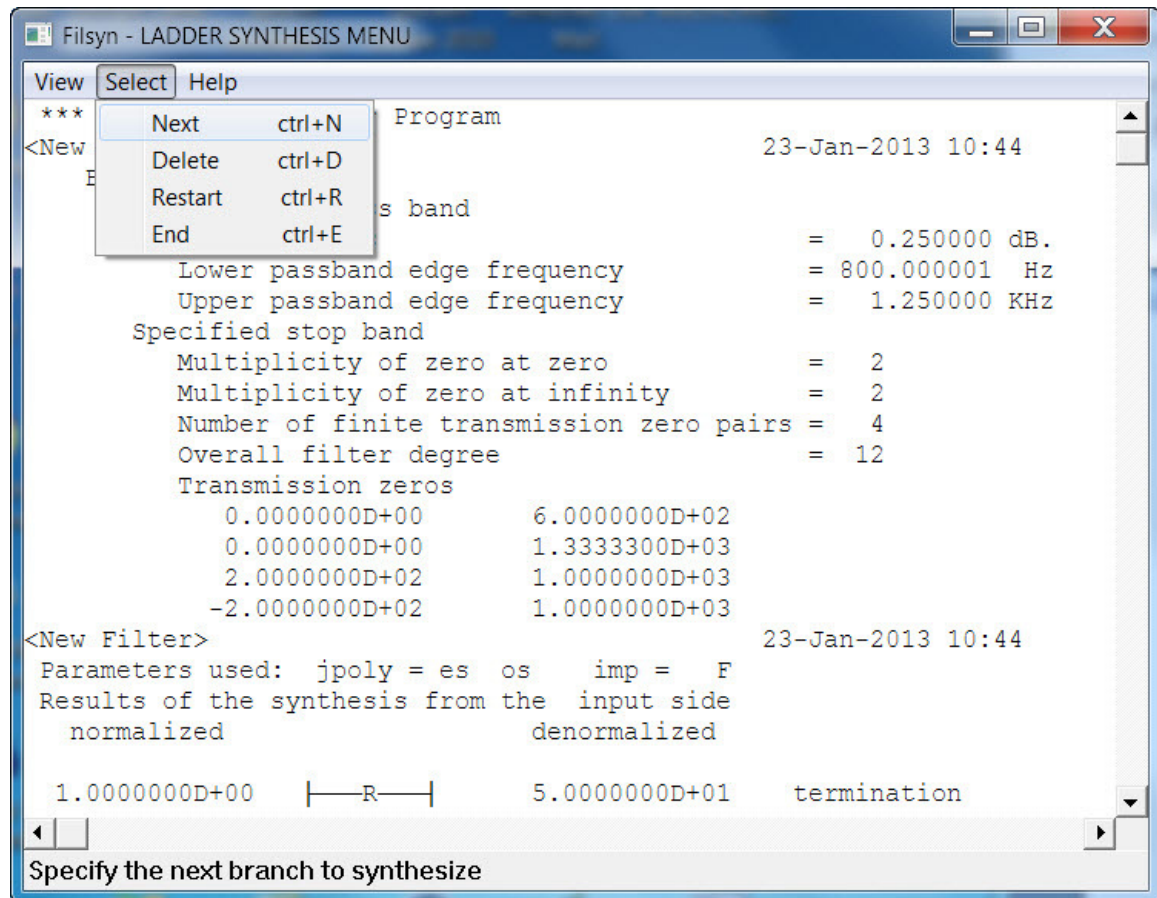
Existing polynomials:

☐ Print polynomials

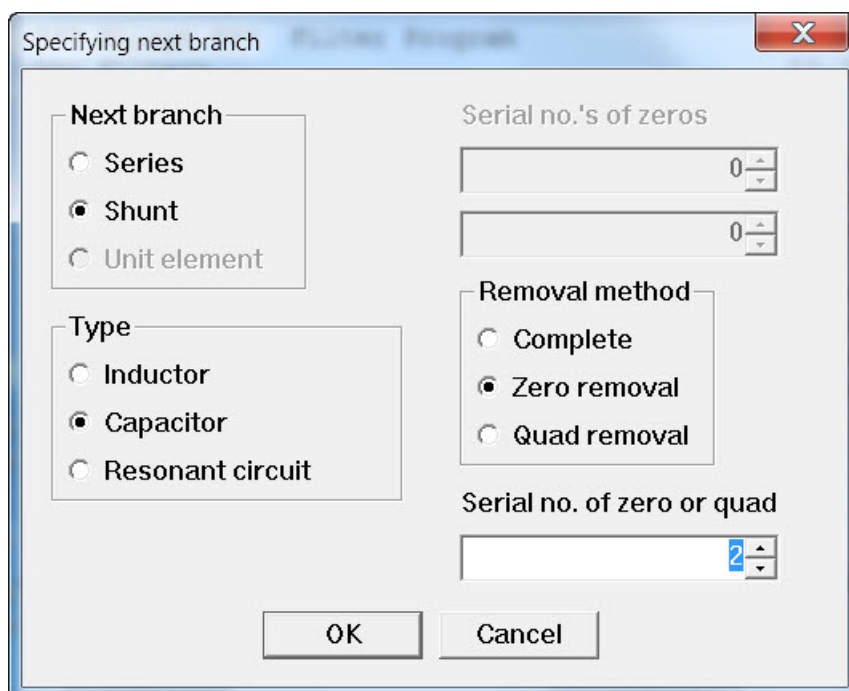
OK Cancel

where we selected the **Manual** option, using the **Forward** option that we must start with and left the selection of the functions to the program. Note that at the bottom of the screen we can see that the ES, OS and ED polynomials are present and if we need them, they can be sent to a text file, where we can see them in detail.

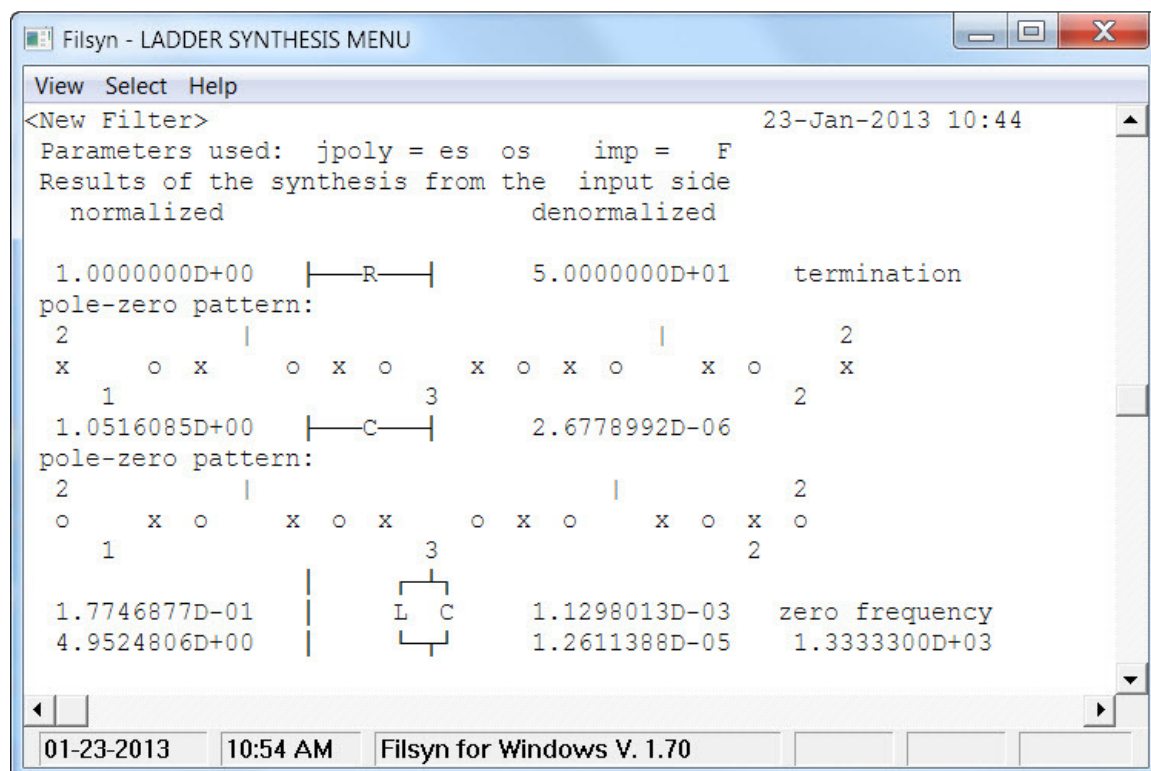
This brings us to the synthesis screen shown below. The menu choices are few, the most significant is the **Select->Next** option that brings up the branch/section select menu.



We start with a transmission zero removal section that uses a shunt C to prepare for it and the zero is consequently the one that is above the passband:



Clicking on the **OK** button brings up the circuit as shown below:

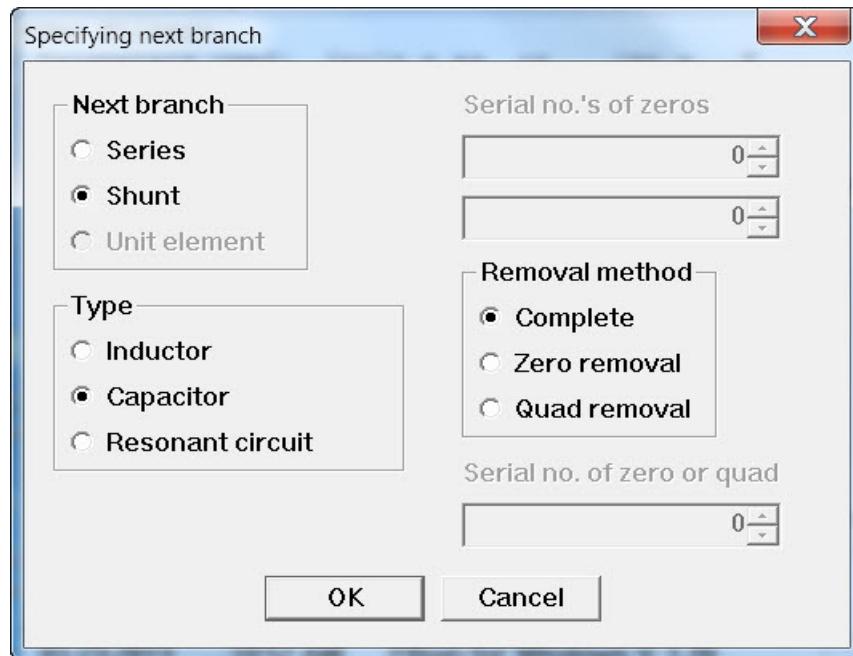


Note the four lines headed by the title: 'pole-zero pattern:' that appears before each branch is computed and printed.

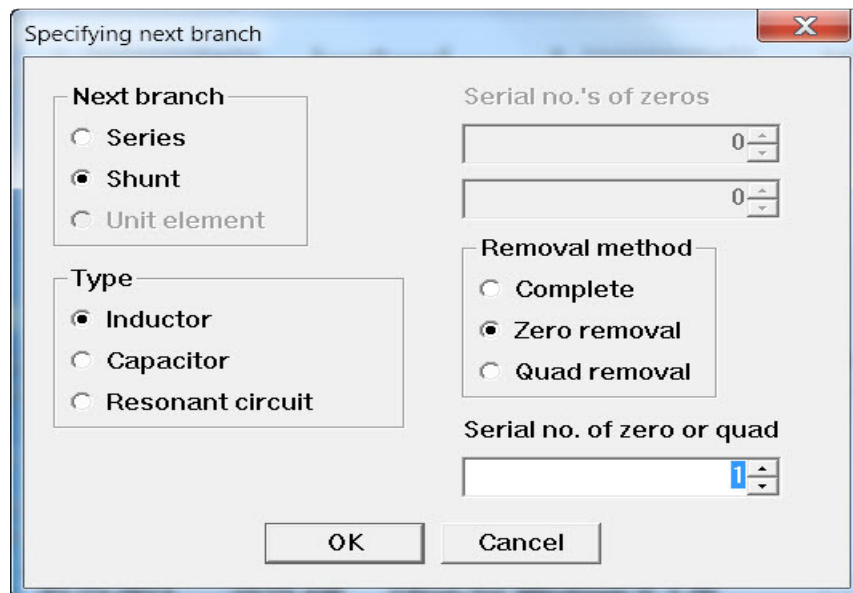
The top line indicates the double zeros at zero and infinity and the two i's indicate the relative positions of the passband edge frequencies. The second line shows the pole-zero sequence of the impedance function while the bottom one shows the relative locations of

the finite transmission zeros. No. 3 is the complex quad, and usually is in the middle of the passband. This is extremely useful in that it shows the situation as described at the beginning of this appendix and helps in selecting both the correct immittance and the correct synthesis step if we run into trouble.

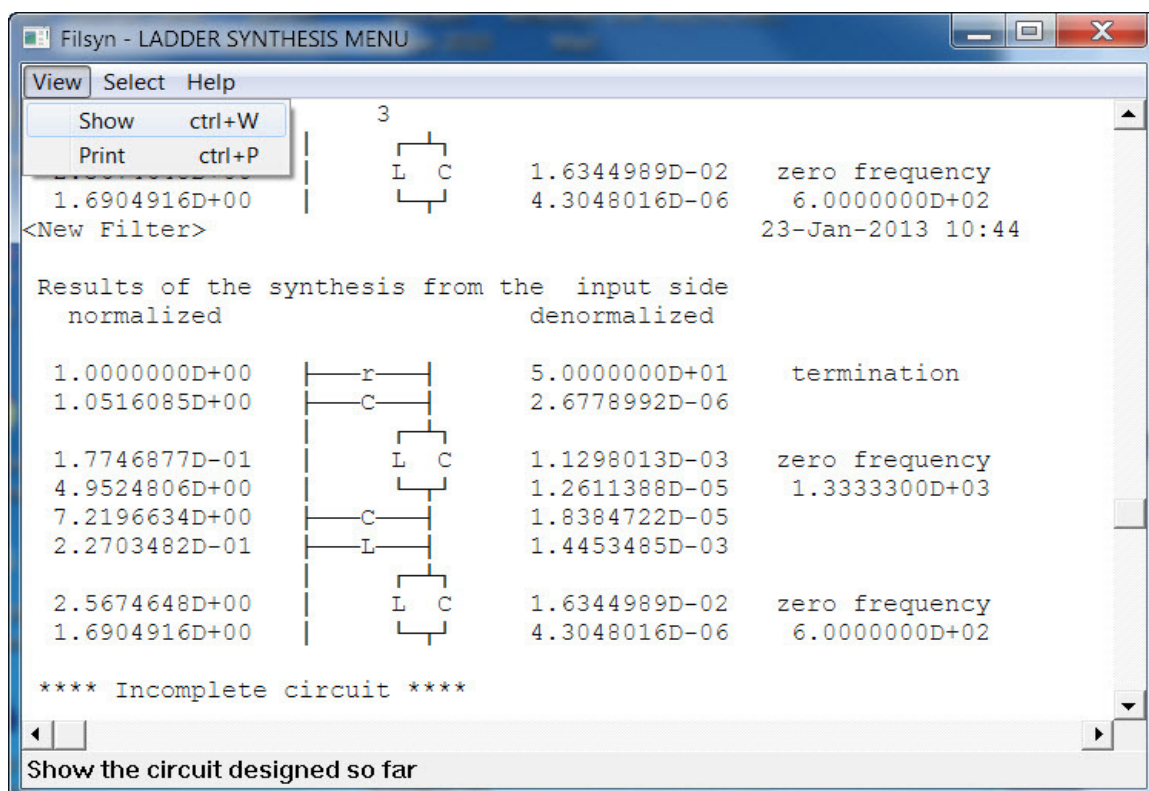
Next we must remove a single shunt capacitor if we want to be able to synthesize the *same* circuit in the reverse direction:



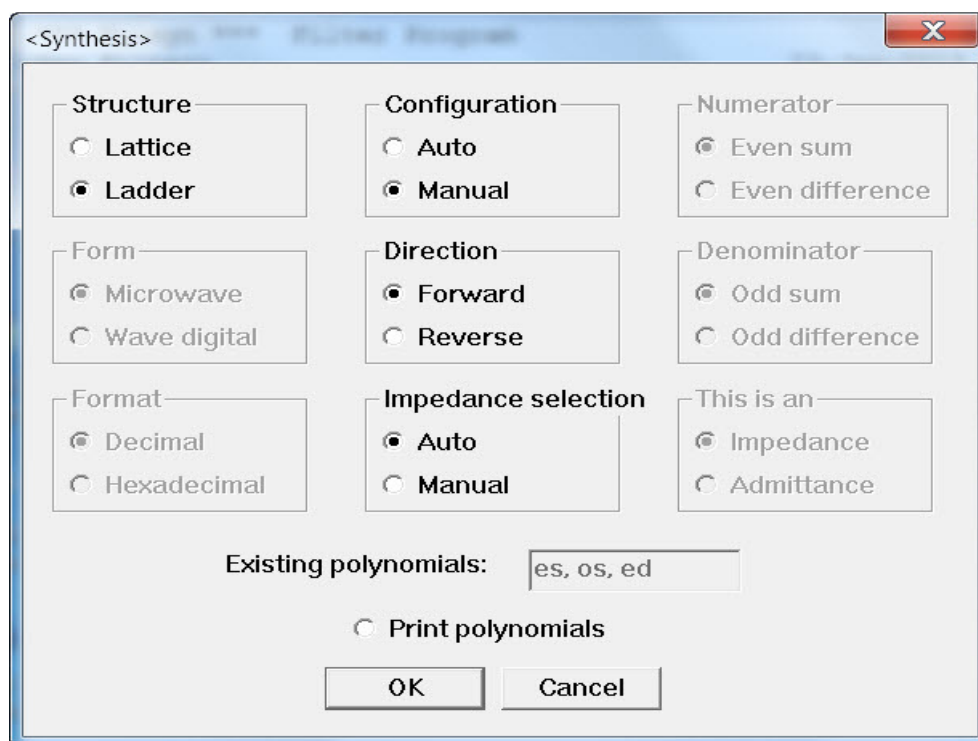
We follow that with another zero removal section, starting this time with a shunt inductor:



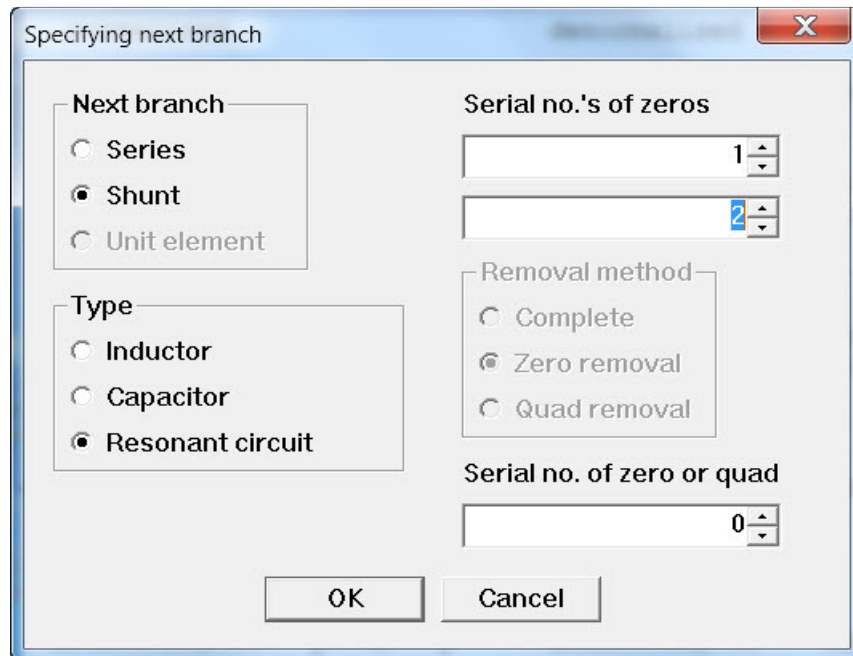
At this point we look at the circuit as it is, by using the **View->Show** menu option:



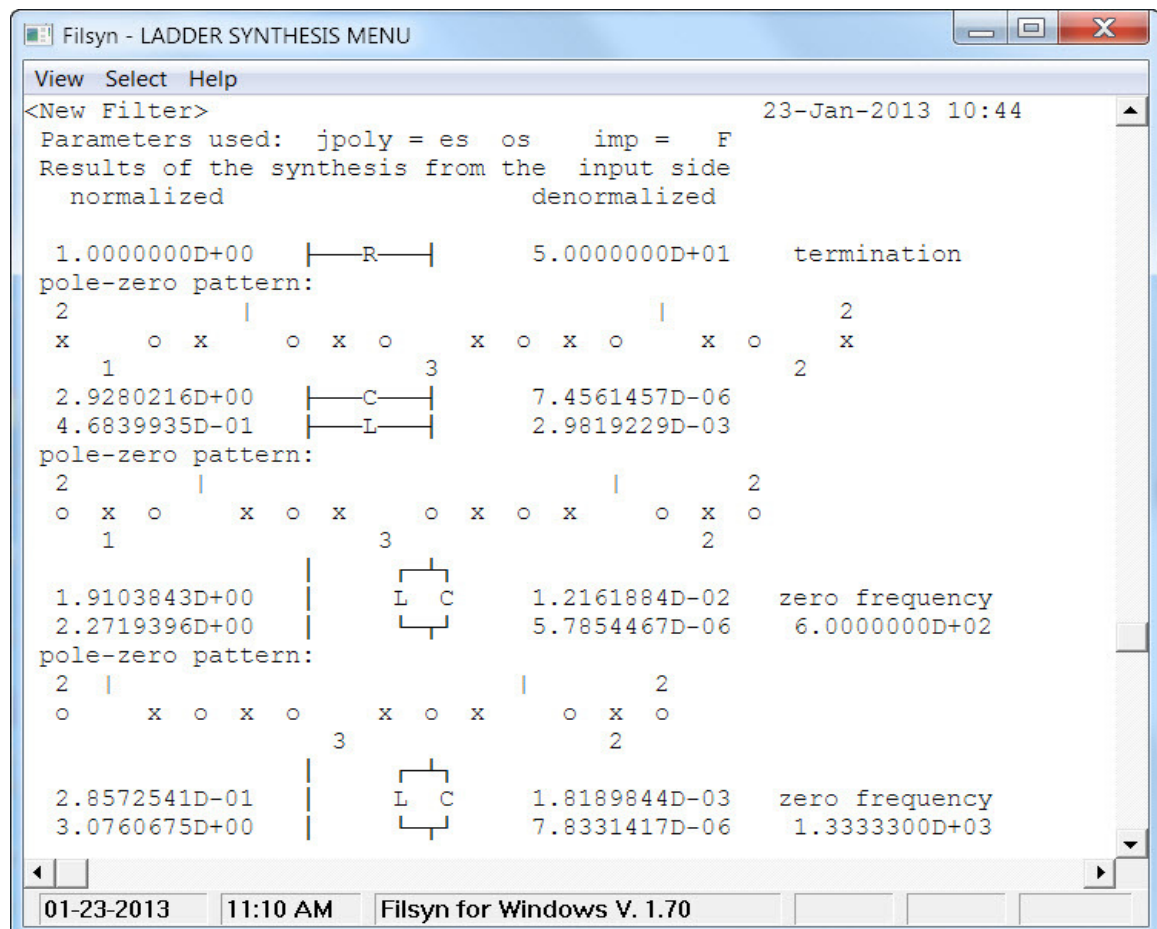
It turns out, that we do not like this circuit at all and wish to delete it completely and start over again. A branch may be deleted by the use of the **Select->Delete** menu option, which deletes the last branch. Since we have a total of 5 branches, we must use this menu option 5 times and we are back at the starting point. Alternatively, we can simply use the **Select->Restart** option to achieve the same effect and get to the starting point:



This time we start with a shunt resonant circuit that will implement both transmission zeros at the same time:



The resulting circuit is shown next:



Next we remove a single shunt inductor:

Specifying next branch

Next branch

☐ Series

☒ Shunt

☐ Unit element

Type

☒ Inductor

☐ Capacitor

☐ Resonant circuit

Serial no.'s of zeros

0

0

Removal method

☒ Complete

☐ Zero removal

☐ Quad removal

Serial no. of zero or quad

0

OK Cancel

Instead of showing the additional shunt L, we specify the next section that will implement the complex quad of transmission zeros and will start with a shunt C. The menu selection for this section is as shown below:

Specifying next branch

Next branch

☐ Series

☒ Shunt

☐ Unit element

Type

☐ Inductor

☒ Capacitor

☐ Resonant circuit

Serial no.'s of zeros

0

0

Removal method

☐ Complete

☐ Zero removal

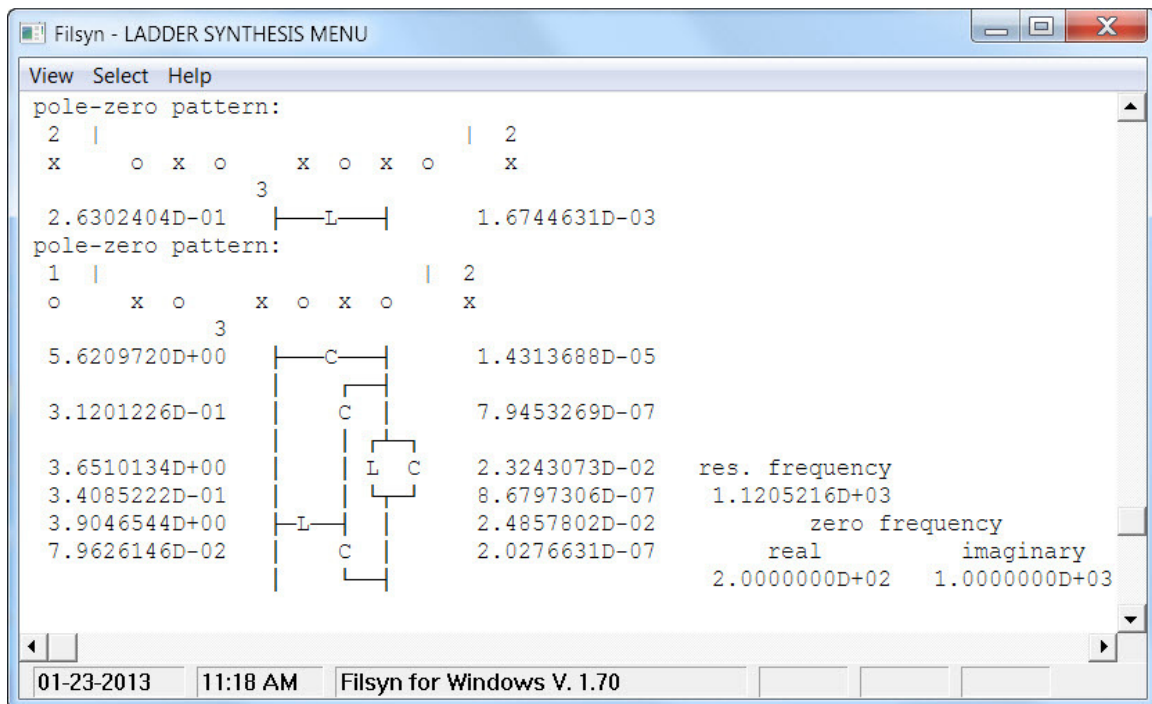
☒ Quad removal

Serial no. of zero or quad

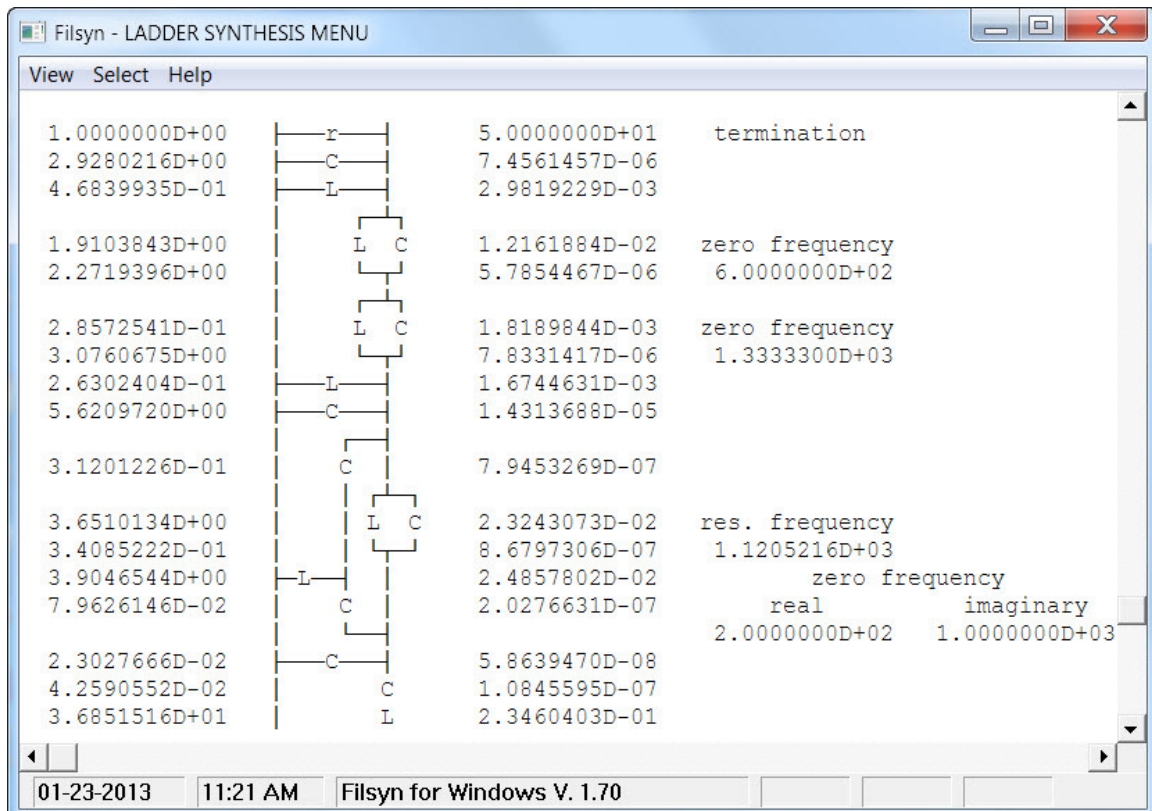
3

OK Cancel

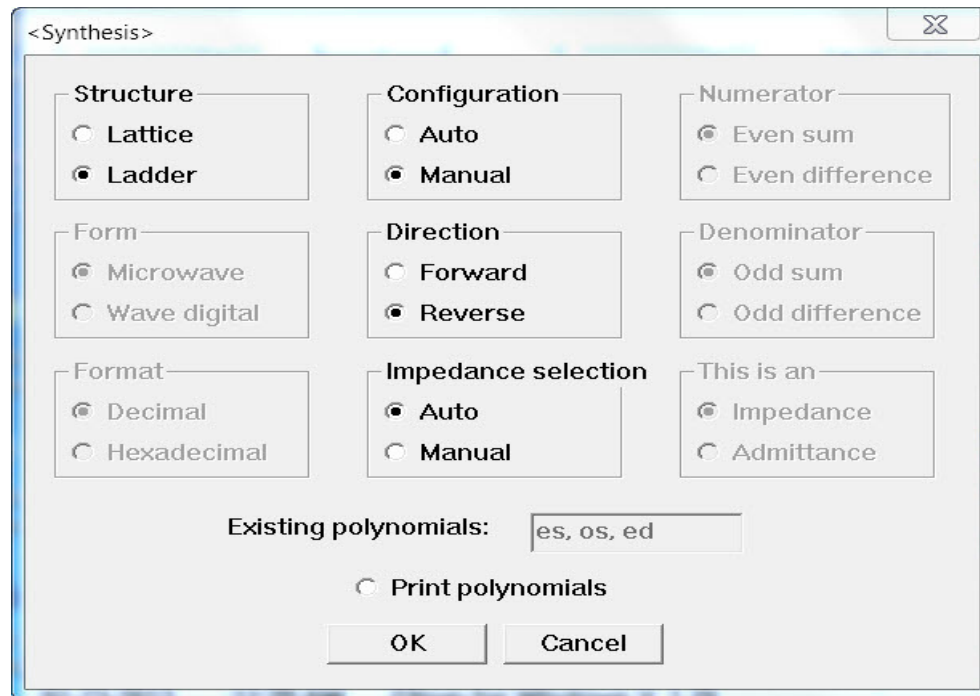
The previous two steps yield the following part of the circuit:



We are close to being done, we need a shunt capacitor removal, then a series capacitor and finally a series inductor removal. At this point we have completed the forward synthesis and must next repeat the process in the reverse direction. The **View->Show** Menu shows the complete circuit:

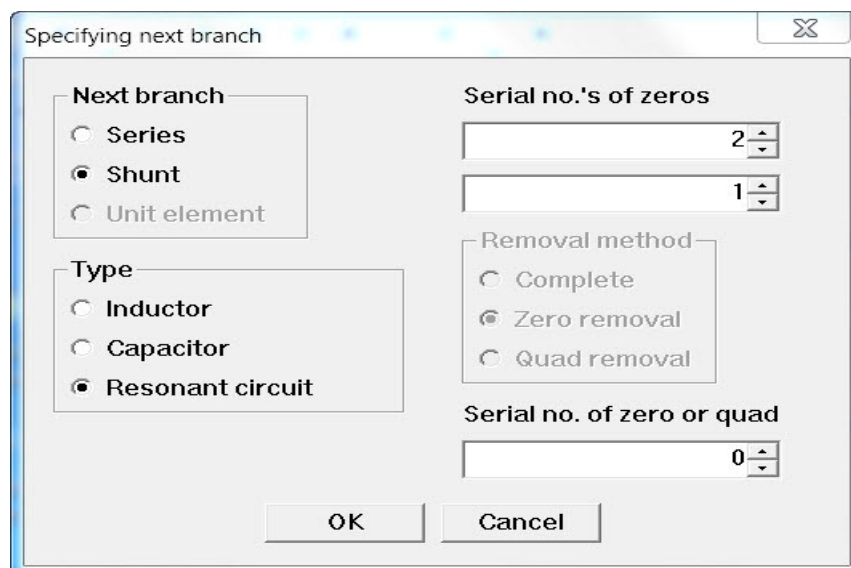


To restart the synthesis in the reverse direction, we use the **Select->End** menu option which is followed by the impedance selection menu, where the only change is that in the **Direction** box we must select **Reverse** instead of **Forward**.

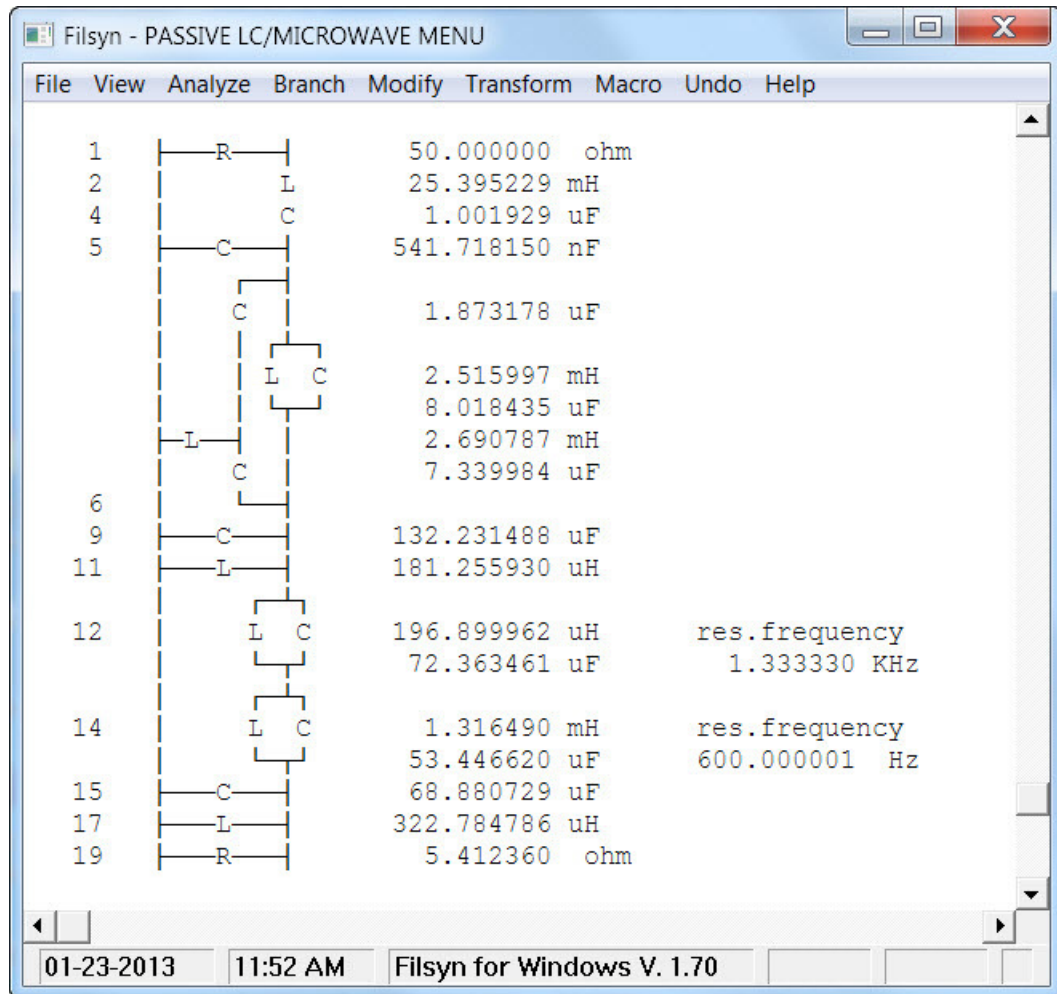


In the special case of an open- or short-circuited termination (used for multiplexer design), there is only the forward synthesis and the **Select->End** menu option will terminate the synthesis.

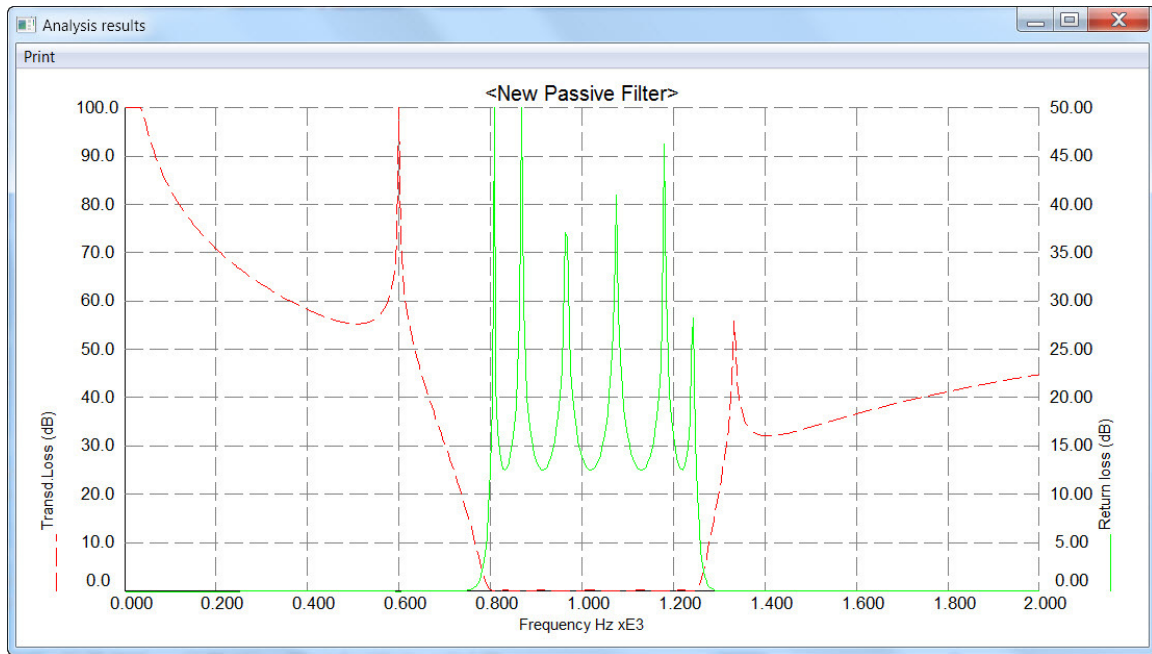
Clicking OK we are again in synthesis mode and can synthesize the same circuit in the reverse direction, starting with a series L, series C, shunt C together with the quad and so forth. The only thing we have to be careful about is when we get to the spot where we implement the two zeros at real frequencies; the sequence of zeros must also be reversed in order that the program should be able to complete the synthesis:



This step is followed by a shunt C and a shunt L, after which we use **Select->End** again. This time the program does not go back to the beginning of the synthesis mode, but goes directly into the analysis segment:



Because both terminations are printed, the program compared the two circuits, found them to represent identical configurations, found the values obtained to be of sufficient accuracy, and therefore the program completed the design. Now we are in the passive LC/microwave analysis segment of the program and the only item of interest is a frequency domain analysis to validate the correctness of the synthesis:



While this manual synthesis method is quite complex, it is also very powerful and the availability of the pole-zero pattern of the immittance at every step helps a great deal.

B.4.2 Comments

Note that the two (forward and reverse) syntheses yielded the *same* structure but different element values. This is typical of bandpass filters and also microwave highpasses with unit elements, and implies a hidden impedance transformation. If we compare the corresponding element values from the forward and reverse syntheses, their ratio is found to be 2.5482818 for *all* of them, with only the last digit varying due to roundoff. This is the value of the impedance transformation and will also be the value used to calculate R_2 from R_1 , as well as any other missing element. The constancy of this ratio is an indicator of the accuracy maintained throughout the computation and shows an accuracy of at least 8 decimal places here.

For very high degree filters this may not be the case, i.e. numerical problems may arise. We then have two options. If a few consecutive elements can be found in the middle of the circuit where this ratio is reasonably constant (to 4 or more decimal places), we may use this ratio with half the elements (the first half) from the forward synthesis and the rest from the reverse synthesis, suitably scaled. Alternatively, the discarded, lower degree polynomials may be used for the synthesis. They seem to yield better precision in such cases, although some of the elements will be missing from both syntheses. However, a careful comparison will readily yield these elements.

B.4.3 Microwave bandpass example

Consider a microwave bandpass with no finite transmission zeros. We will use three unit elements and the monotonic stopband feature to meet two stopband requirements as indicated in the data input sequence below.

<New Filter>

Filter kind

- ☐ LC
- ☒ Microwave
- ☐ Active RC
- ☐ IIR Digital

Quarter-wave/Sampling freq (Hz)

100.000000E+06

Filter type

- ☐ Low pass
- ☐ High pass
- ☒ Band pass

Lower passband freq (Hz)

45.000000E+06

Upper passband freq (Hz)

55.000000E+06

Passband type

- ☐ Max. flat
- ☒ Equal ripple
- ☐ Functional
- ☐ Sloping

Band-edge loss/return loss (dB)

0.500000000

Loss Slope (dB/oct)

6.00000000

Flat loss (dB)

0.00000000

Function Type

- ☒ E
- ☐ F

Multiplier

0.00000000

Stopband

- ☒ Monotonic
- ☐ Equal min
- ☐ Placer
- ☐ Specified

Lower stopband freq (Hz)

40.000000E+06

Loss (dB)

30.00000000

Upper stopband freq (Hz)

60.000000E+06

Loss (dB)

30.00000000

of zeros at zero

0

of zeros at FQ or FS/2

0

of unit elements

3

Parametric

- ☒ Conventional
- ☐ Parametric
- ☐ Matching

R1

75.000000E+00

R2

75.000000E+00

ZS parameter

-1

Q for predistortion

0.00000000

Predistortion

- ☒ Passband only
- ☐ Pass/stop bands

Shifted bandpass trans.

Center frequency (Hz)

0.000000

☐ Odd parametric

☐ Save design data

OK **Cancel**

The screen shown below indicates that we need 4 zeros at zero and 3 at the quarter-wave frequency. We will see later, that the requirements are met with plenty to spare and the main reason for that is that we specified a conventional structure, which needs an even overall degree. A parametric design might have needed one fewer components.

Filsyn - MAIN MENU

File Design Analysis Help

*** Filsyn *** Filter Program

<New Filter> 23-Jan-2013 14:46

Band-pass filter

Equal ripple pass band

Band-edge loss = 0.500000 dB.

Max. passband vswr = 1.984056

Preshifted Lower passband edge frequency = 40.119960 MHz

Lower passband edge frequency = 45.000000 MHz

Upper passband edge frequency = 55.000000 MHz

Quarter-wave frequency = 100.000000 MHz

Monotonic stopband

Multiplicity of zero at zero = 4

Multiplicity of zero at Quarter-wave fr. = 3

Number of unit elements = 3

Overall filter degree = 10

<Synthesis>

Structure

- ☐ Lattice
- ☒ Ladder

Configuration

- ☐ Auto
- ☒ Manual

Numerator

- ☒ Even sum
- ☐ Even difference

Form

- ☒ Microwave
- ☐ Wave digital

Direction

- ☒ Forward
- ☐ Reverse

Denominator

- ☒ Odd sum
- ☐ Odd difference

Format

- ☒ Decimal
- ☐ Hexadecimal

Impedance selection

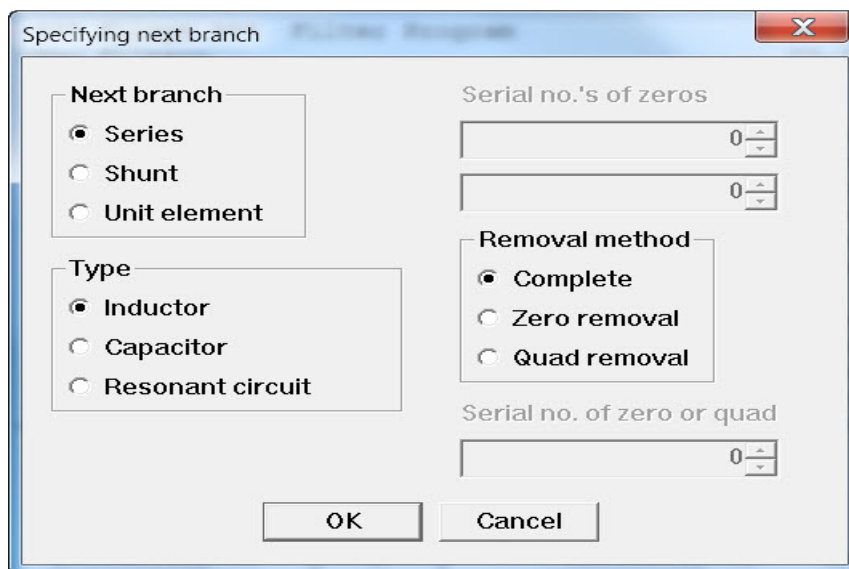
- ☒ Auto
- ☐ Manual

This is an

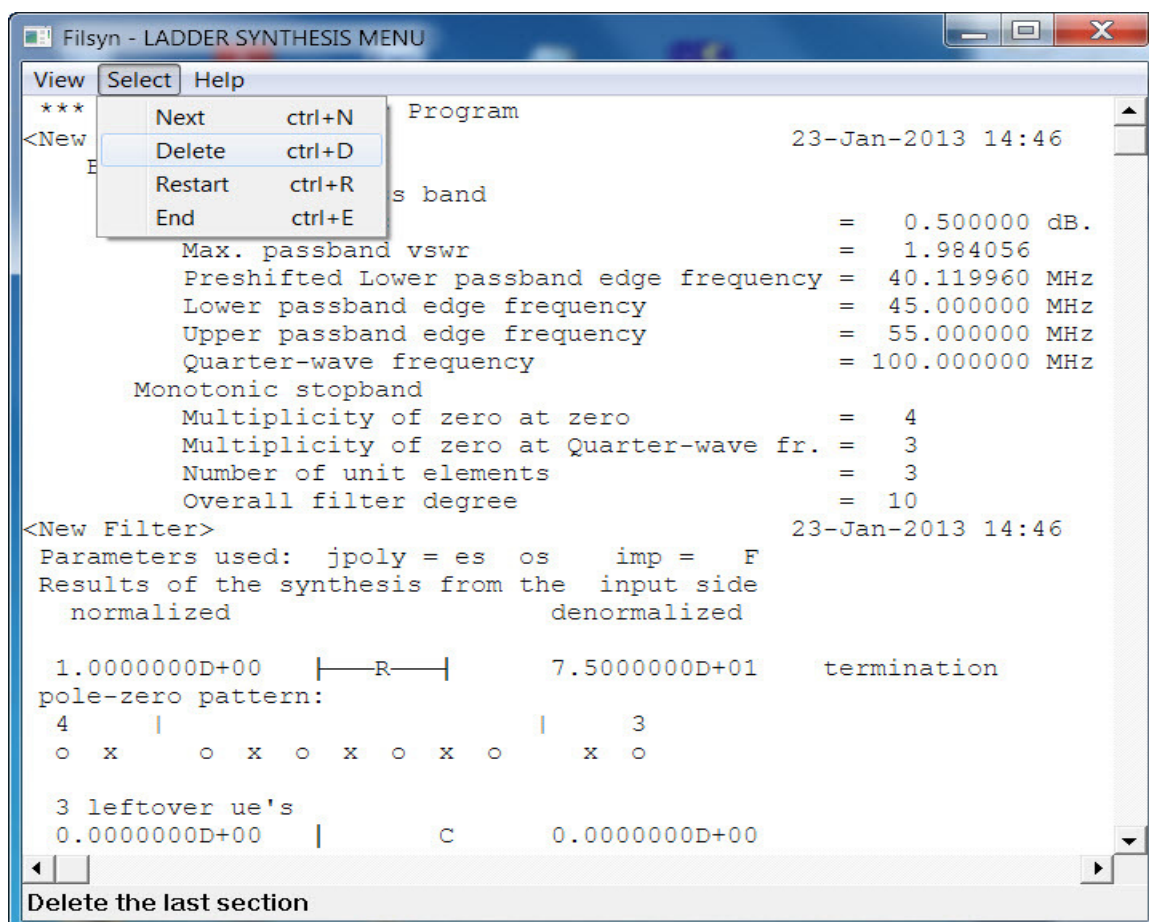
- ☒ Impedance
- ☐ Admittance

Existing polynomials:

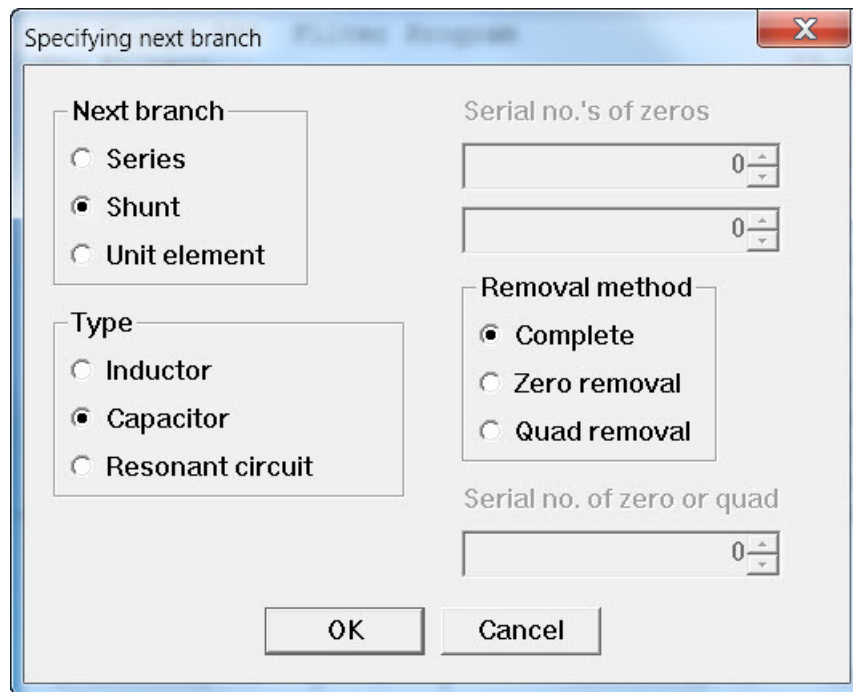
Selecting the manual synthesis mode again as shown above, first we try to remove a series inductance:



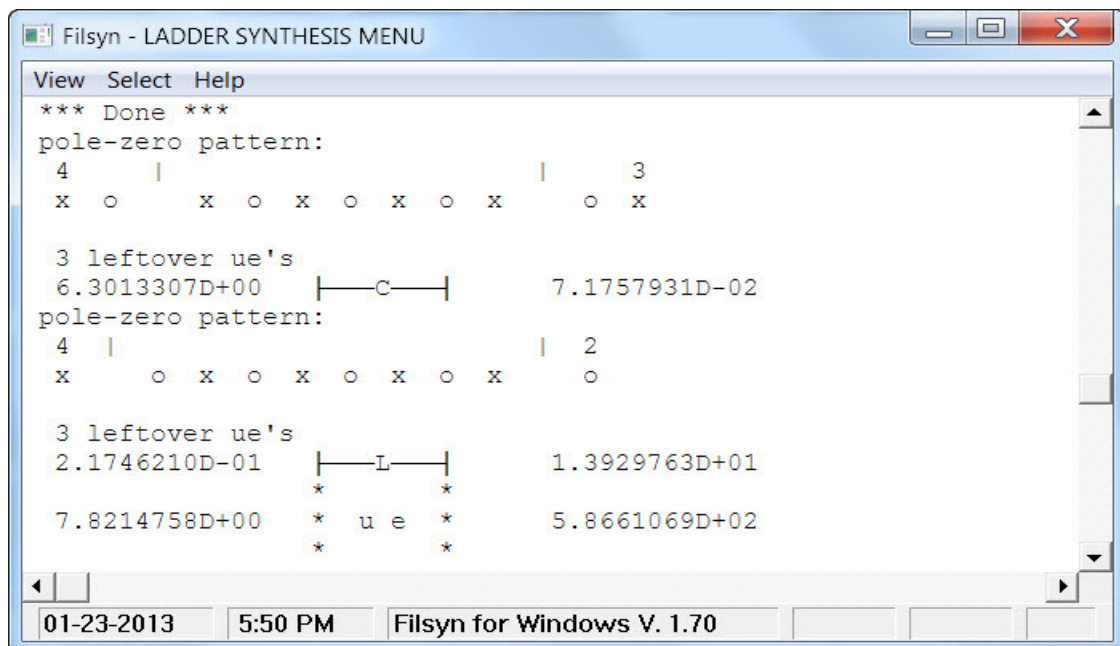
That does not work, since the selected impedance has no pole at the quarter-wave frequency. This is indicated by the resulting zero element value; that it is shown as a capacitor is immaterial.



Note the additional line in the 'pole-zero pattern' display, this tells us how many unit elements we have left to implement at any particular moment. Deleting this incorrect branch, we try a shunt capacitance instead:

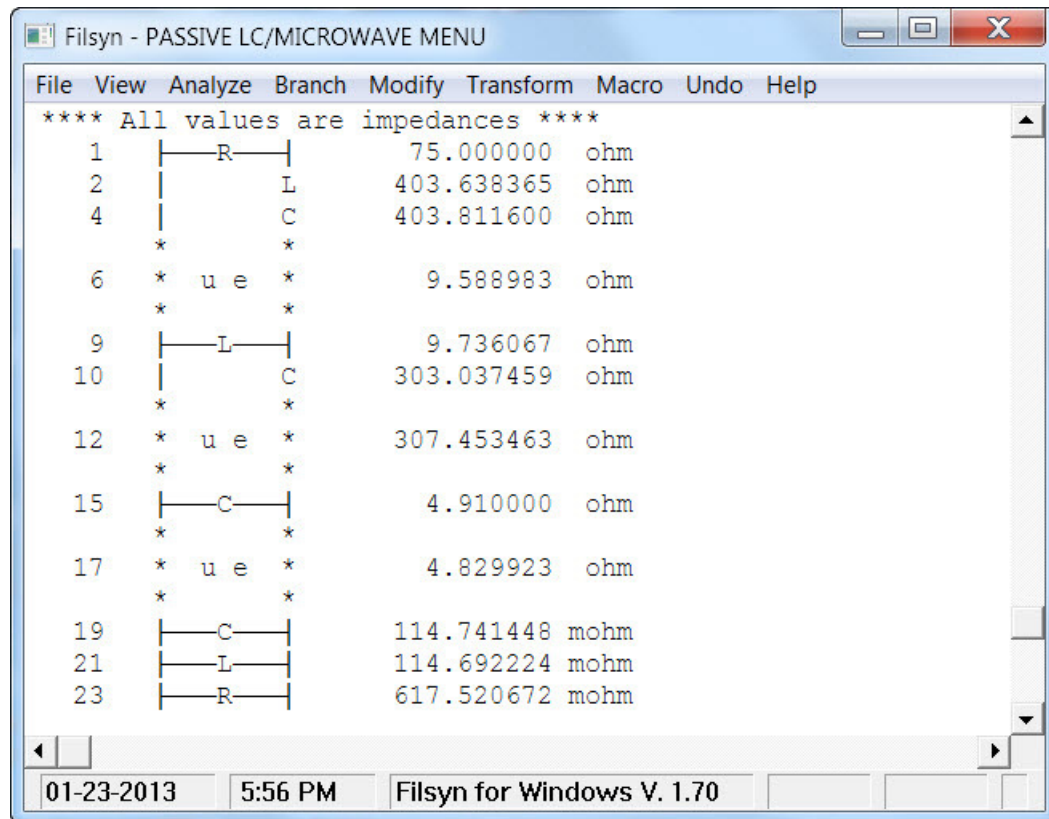


This works fine and we follow that with a shunt inductor and a unit element. The unit element removal does not show the pole-zero pattern:



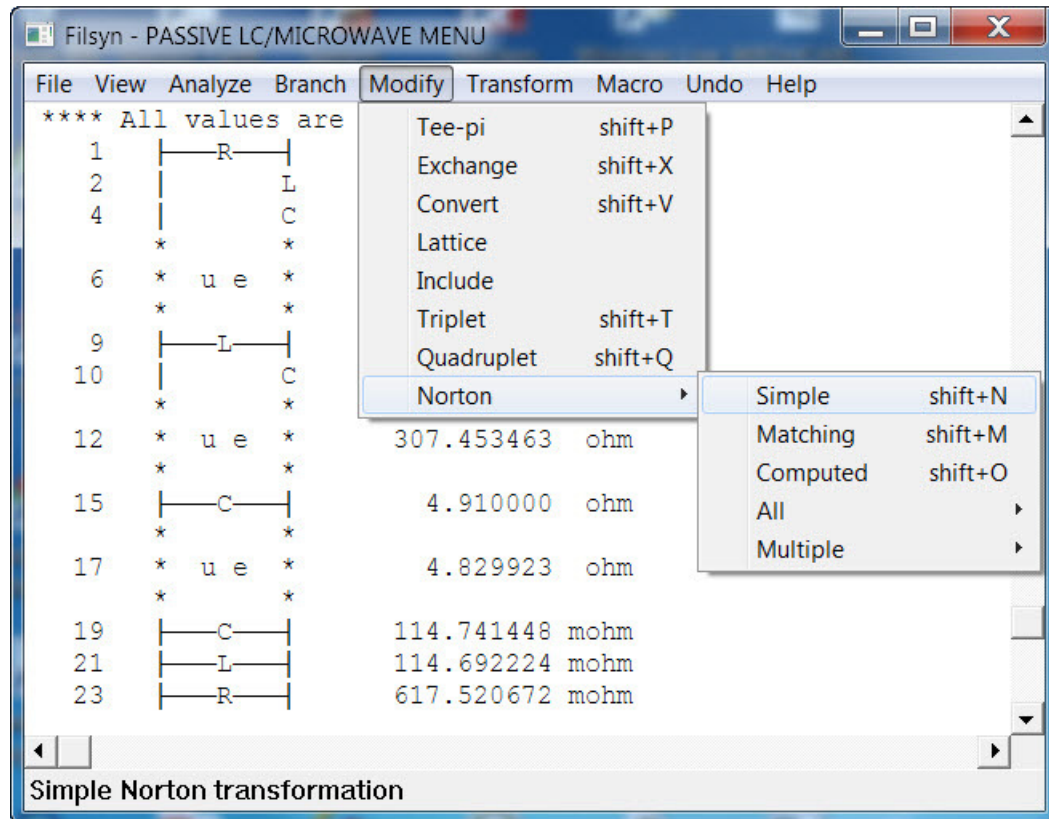
We implement the rest of the circuit more or less arbitrarily; another shunt C, a unit element, a series C followed by a shunt L, next another unit element and finishing up with a series L and a series C.

Because this filter had no finite transmission zeros, any arbitrary realization, even the one we picked above, may be synthesized in the reverse direction. After the program compared the two syntheses and completed the design, the final results are shown below:

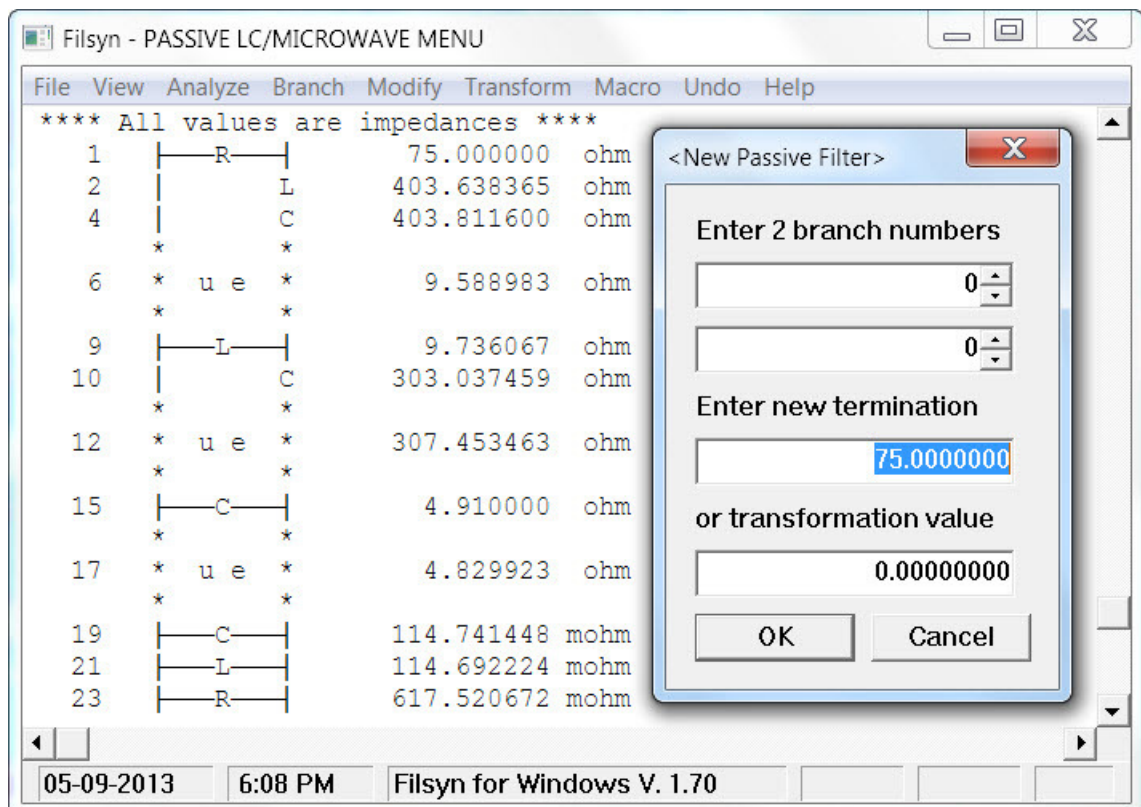


These element values are not very practical. Either a different synthesis or the application of some of the manipulative steps of the analysis segment or both, will however, yield an acceptable set of element values.

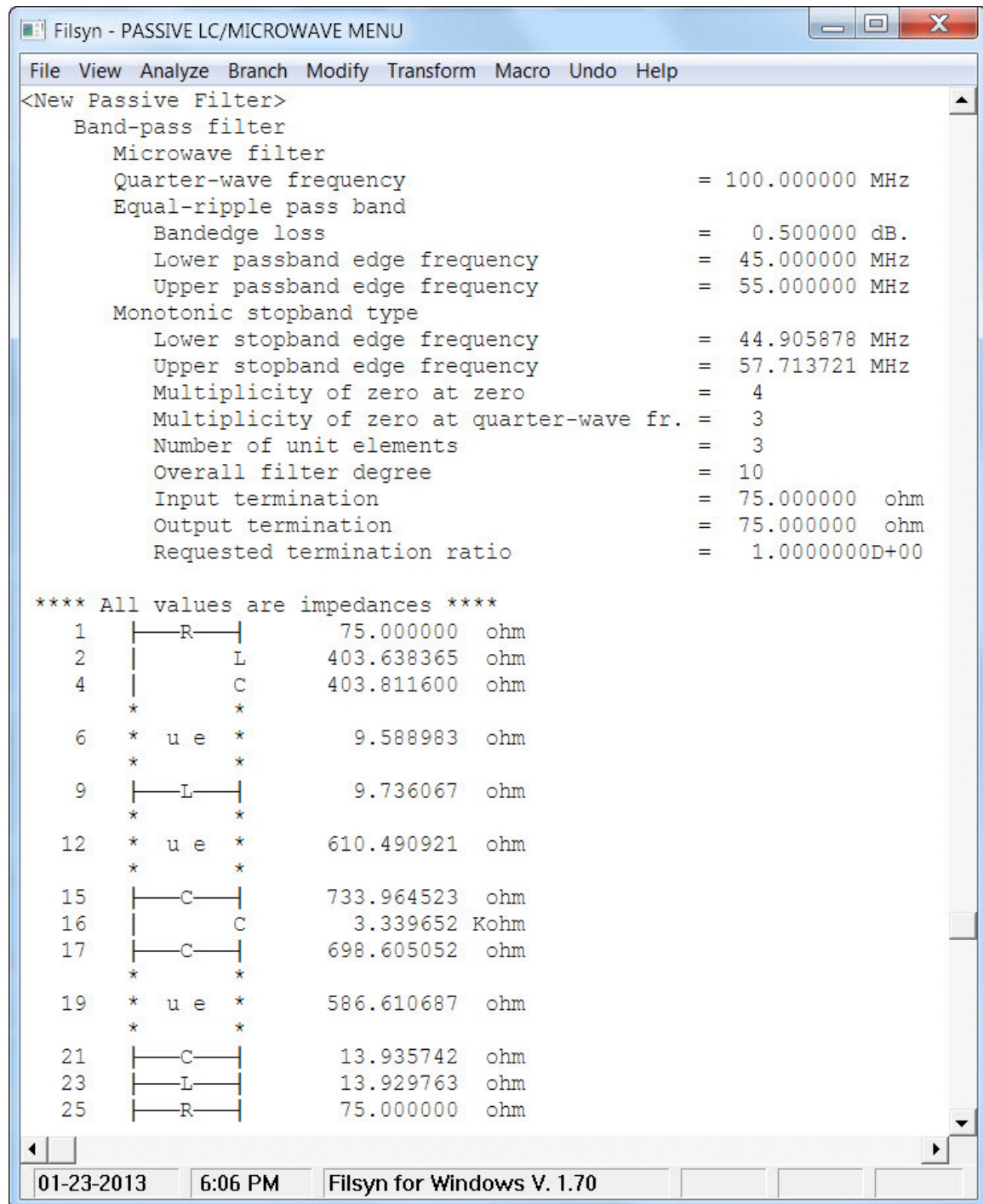
To demonstrate how to perform some of these manipulations, let us try to perform a Norton transformation in order to change the output termination to 75 ohms also. We select the **Modify->Norton->Simple** menu item:



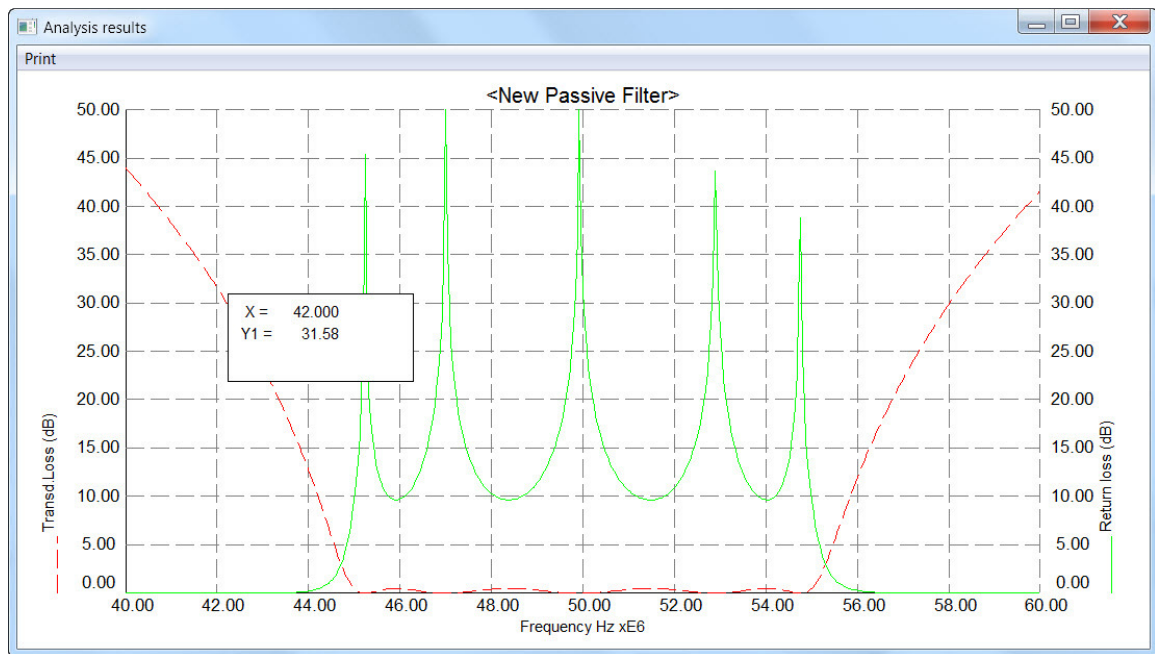
which leads to the data input screen, where we only enter the required output termination value:



The program selects a pair of elements and performs the transformation. However, while the output termination is now larger, it is still not quite 75 ohms. The computer repeats the process once more to yield the required termination in the circuit shown below:



Finally, we show the frequency domain analysis results:



One interesting feature of the plot is illustrated here. The mouse pointer turns into a crosshair inside the plot. If you move this crosshair close to a curve and push the right mouse button, the program displays the coordinates (i.e. the frequency and the loss here) of the nearest point on the curve.

APPENDIX C

PARAMETRIC BANDPASS FILTERS

Conventional bandpass filters (the kind that were the only ones known until the early 1960's) end like Fig. 1 case **a**) or **b**) or a combination, at both ends (assuming finite terminating resistances):

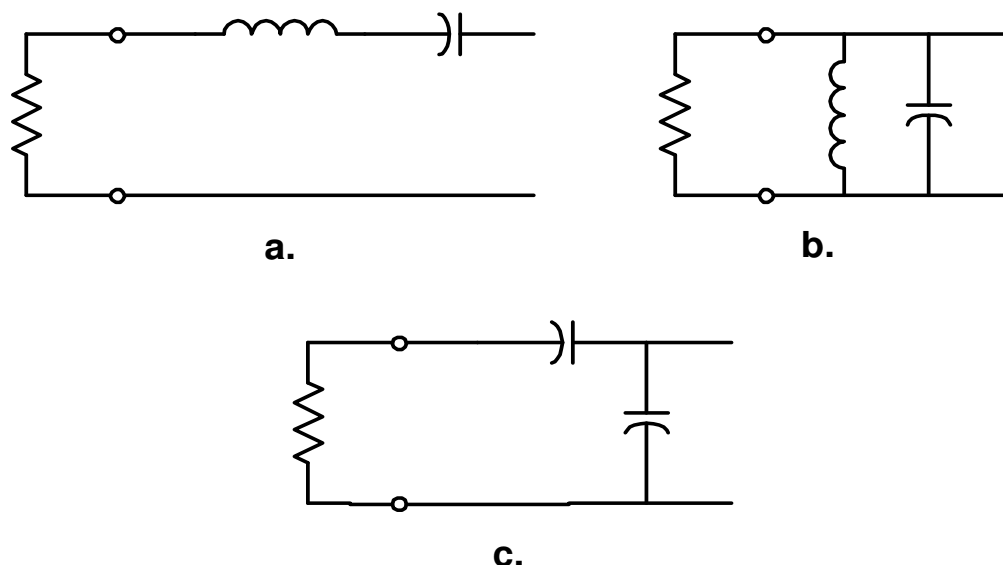


Fig. 23 Bandpass filter ends

It is clear that while internal inductors could be used efficiently to create finite frequency transmission zeros, the two inductors at the ends of the network were always needed and could not be so employed.

Parametric filters, or rather ways of using them in insertion loss design theory, were discovered to circumvent this problem and yield filters more efficient in their utilization of inductors. These could have end structures of the form **c**) in Fig. 23 above at either or both ends and in fact, any combination of two out of the three ending is now possible. If both ends are like figure **c**) above, we have a symmetric parametric case, while if **c**) at one end is combined with either **a**) or **b**) at the other, we have an asymmetric parametric filter of odd degree. Note that conventional filters are always of even overall degree. The dual to figure **c**) is also possible, but is not used in practice because it uses *more* rather than fewer inductors. Only the efficiency with respect to the number of *inductors* is improved; efficiency with respect to the overall degree is reduced somewhat. Therefore these designs should *not* be used in active and digital filters and only rarely in microwave filters.

In the asymmetric parametric case both $E(s)$ and $F(s)$ will be of odd degree, i.e. both will have at least one real root. The value of the real root of $F(s)$ is a free parameter (hence the name *parametric*) that will have a small effect on the overall loss behavior of the filter. There is a default value available which can be replaced by any non zero real value by the

user. In the symmetric parametric case, $F(s)$ *must* have a pair of real roots, one positive the other negative, usually equal in magnitude, i.e. representing a factor

$$(s^2 - a^2)$$

in this polynomial. The presence of this factor in $F(s)$ will guarantee that $E(s)$ will also have a pair of real roots, but *not* vice versa. Therefore symmetric parametric cases may be specified by functional input if $F(s)$ is entered, but not necessarily if $E(s)$ is the function specified. If we *do* specify the $E(s)$ polynomial with two real roots as a parametric filter and it turns out not to be parametric, the program will print a comment to that effect.

The program usually calculates the characteristic function such that the passband is exactly maximally flat, but only *approximately* equal ripple. However, the deviation from the exactly equal ripple behavior is *always* negligible, is more or less independent of the bandwidth and improves with increasing overall degree. On the other hand, the behavior of the stopband is more affected by this parameter, unless its value is equal or close to the default value, which is the geometric mean of the passband edge frequencies.

Finally, the symmetric parametric cases need finite terminations at both ends of the filter. Extreme termination at one end is possible only in the asymmetric (odd degree) parametric cases.

During the synthesis of a passive or microwave parametric filter, either the forward or the reverse synthesis (for asymmetric cases) or both (for symmetric cases) will be missing an element at the far end of the structure. These missing elements can only be recovered when the two syntheses are compared and the program does this both for the computer-generated configuration and for the user selected one. In the case of the user specified structure however, it can only do so if the two syntheses represent the *same* structure and we used **Forward** for the first synthesis and **Reverse** for the second (see *Appendix B* above). Also, this comparison may not work for one of the last few elements, but the program prints a warning message to that effect. If such a warning is printed, the value of the indicated element must be checked.

APPENDIX D

THE ZS PARAMETER

This parameter affects the design of certain even degree lowpass and highpass filters and is of significance *only* if passive LC or microwave realization is used.

Consider an (even) 4th degree, lowpass function of the elliptic type, with an equal minima stopband and an equal ripple passband up to 1 KHz. The direct elliptic design would yield the curve labeled “**Case a**” in Fig. 24, which is not realizable in ladder form without the use of tightly coupled inductors. This is because the loss at infinite frequency is finite. This is the case with the ZS parameter set to -1 and yields the mathematically most efficient design. In active or digital realizations we may retain this case since it presents no difficulties of realization.

A ladder-realizable function can be derived from the above function if we shift the highest trans- mission zero to infinity. This is the “**Case b**” curve in the figure. The penalty we pay is a slight increase in the transition bandwidth, or, as shown in the figure, a decrease in the minimum stopband loss. However if we had excess stopband loss, this may be acceptable.

This new transfer function still has the undesirable property, that its loss at zero frequency is not zero. The consequence of this is that the two terminations may *not* be equal and their ratio will depend on this non zero loss. One can eliminate this problem by shifting the lowest frequency where the loss is zero, down to zero frequency. The resulting characteristics will behave like curve “**Case c**” in the figure and this corresponds to the case $ZS = 1$. The price we pay is a further widening of the transition between pass and stop bands, or a further reduction in the minimum stopband loss. These transformations are executed to preserve the passband edge frequency and the program *also* preserves the stopband edge frequency in favor of the stopband loss if an elliptic design is specified. If the stopband loss is reduced below the requirement, the program will increase the filter degree.

These distinctions, of course, also occur in other than elliptic function filters. Any time the passband is of equal ripple, the distinction between $ZS = 0$ and $ZS = 1$ exists for even degree low- passes and highpasses; otherwise it is irrelevant and ignored. Similarly, if an equal minima stopband is selected in even degree low- and high-passes with arbitrary passband behavior, $ZS = -1$ and $ZS = 0$ will yield different results; but if there is at least a single transmission zero at infinity for lowpasses, or at zero for highpasses, the distinction disappears.

If a bandpass is obtained from a lowpass by the familiar lowpass-to-bandpass transformation, $ZS = -1$ is still unacceptable for ladder realization but $ZS = 1$ is now unnecessary, because Norton (impedance) transformations can always be applied to adjust the terminations.

The program uses the most efficient method available, depending on the circumstances, *unless* we explicitly specify the value of this parameter. For instance, for active RC and digital filters the value $ZS = -1$ is automatically selected. For passive LC and microwave filters $ZS = 1$ is the computer selected value if the two terminations are equal and $ZS = 0$ otherwise. In this latter case the actual value of the termination ratio (the ratio of the output termination to the input one) will depend on the value of the loss at zero (or infinite) frequency.

For microwave filters with unit elements, $ZS = -1$ is immaterial because the explicit elliptic design is not known and the **Placer** segment must be used. If we then specify at least one transmission zero at the quarter-wave frequency, the problem is eliminated. Finally the $ZS = 1$ case is *not* available, because the indicated transformation cannot be performed.



Fig. 24 The effects of the ZS parameter

APPENDIX E

REFLECTION COEFFICIENT ZEROS

As explained in *Section 3.4* and *Appendix F*, the zeros of the $F(s)$ polynomial are the reflection coefficient zeros at one end of the filter and those of $F(-s)$ are the reflection coefficient zeros at the other end. For most filters, $F(s)$ is pure even or odd, the zeros of $F(s)$ and $F(-s)$ are identical and are on the $j\omega$ axis; one has no choice in selecting these zeros. Under certain circumstances, such as

- linear-phase filters,
- predistorted filters,
- functional input

or their combination, the roots of $F(s)$ will become complex and they may individually be selected to be either in the right- or left-half of the s -plane. A different selection will yield an equivalent filter, but different element values. This feature may be used to obtain better element value distribution, or to help to absorb parasitic reactive elements at one or both ends of the filter.

Not much help is available except the fact that the elements nearest to the terminations will be the largest at one end and the smallest at the other if the zeros of $F(s)$ are all in one side of the s -plane.

If the roots of $F(s)$ are complex, the program offers several options:

- a) *Hurwitz* $F(s)$, where all the zeros of $F(s)$ are in the left side of the s -plane.
- b) non-*Hurwitz* $F(s)$, with a default algorithm to select the location of the zeros,
- c) user-selected placement of the zeros.

In case **c**) the signs of the real parts are individually specified by entering the proper number of +1's and -1's. Since the roots themselves are not visible, we must do this by trial and error.

E.1 Bessel filter with Hurwitz $F(s)$

As an example, consider a Bessel polynomial filter with no finite transmission zeros and realize the filter in passive form. First we select the Hurwitz $F(s)$ polynomial (all zeros have negative real parts). The data input screen is shown below:

<New Delay-Filter/-Line/-Equalizer>

Delay type

- ☒ Max. flat
- ☐ Equal ripple
- ☐ Equalizer

Implementation

- ☒ LC elements
- ☐ Active RC
- ☐ Microwave
- ☐ IIR digital

Data

FQ or FS (Hz)

Normalization freq (Hz)

Network type

- ☒ Filter
- ☐ Delay line

Delay value (sec)

Stopband

- ☐ Equal min.
- ☒ Specified

End of passband (Hz)

Start of stopband (Hz)

Degree

Zeros at infinity

Transm. zeros

Terminations

Q-value

Hurwitz type

- ☒ Hurwitz
- ☐ Non Hurwitz

Selection

- ☒ Auto
- ☐ Manual

Indicators

Shifted bandpass trans.

Center frequency (Hz)

☐ Odd parametric

☐ Save design data

OK **Cancel**

Skipping everything else, the passive synthesis yielded the following circuit:

Filsyn - PASSIVE LC/MICROWAVE MENU

File View Analyze Branch Modify Transform Macro Undo Help

*** Filsyn *** Filter Program

<New Delay-Filter/-Line/-Equalizer> 24-Jan-2013 10:41

Linear phase low-pass filter

Maximally flat delay in band

Low frequency delay = 300.000000 usec

Approx. 1% delay bandwidth = 1.436899 KHz

Specified stop band

Multiplicity of zero at infinity = 5

Number of finite transmission zero pairs = 0

Overall filter degree = 5

1	—R—	50.000000 ohm
3	—C—	5.583142 uF
4	—L—	6.863043 mH
5	—C—	1.987990 uF
6	—L—	3.133316 mH
7	—C—	431.018593 nF
9	—R—	49.982683 ohm

01-24-2013 10:42 AM Filsyn for Windows V. 1.70

E.2 Default algorithm

We redesign the exact same filter but with a non-Hurwitz $F(s)$ using the default algorithm. The data input screen is as follows, where we change only the Hurwitz options:

<New Delay-Filter/-Line/-Equalizer>

Delay type

- ☒ Max. flat
- ☐ Equal ripple
- ☐ Equalizer

Implementation

- ☒ LC elements
- ☐ Active RC
- ☐ Microwave
- ☐ IIR digital

Network type

- ☒ Filter
- ☐ Delay line

Delay value (sec)

300.000000E-06

Stopband

- ☐ Equal min.
- ☒ Specified

End of passband (Hz)

0.00000

Start of stopband (Hz)

0.00000

Degree

0

Zeros at infinity

5

Transm. zeros

50.000000E+00

Terminations

50.000000E+00

Q-value

0.00000000

Hurwitz type

- ☐ Hurwitz
- ☒ Non Hurwitz

Selection

- ☒ Auto
- ☐ Manual

Indicators

Shifted bandpass trans.

Center frequency (Hz)

0.00000

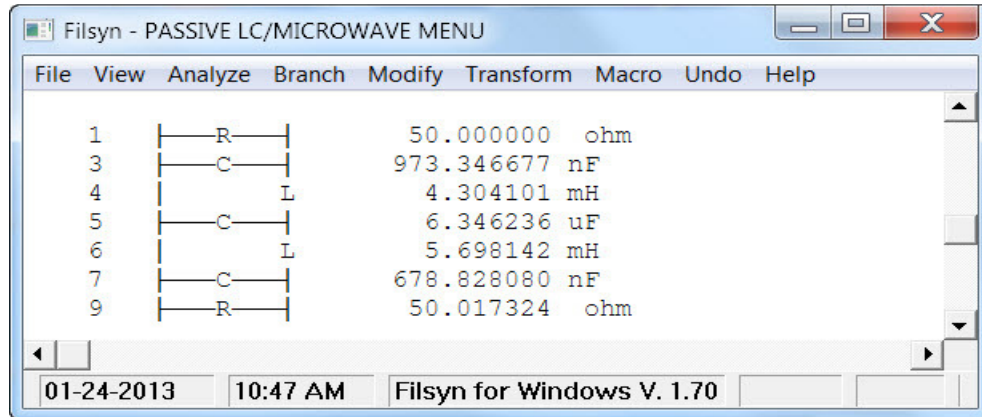
☐ Odd parametric

☐ Save design data

OK Cancel

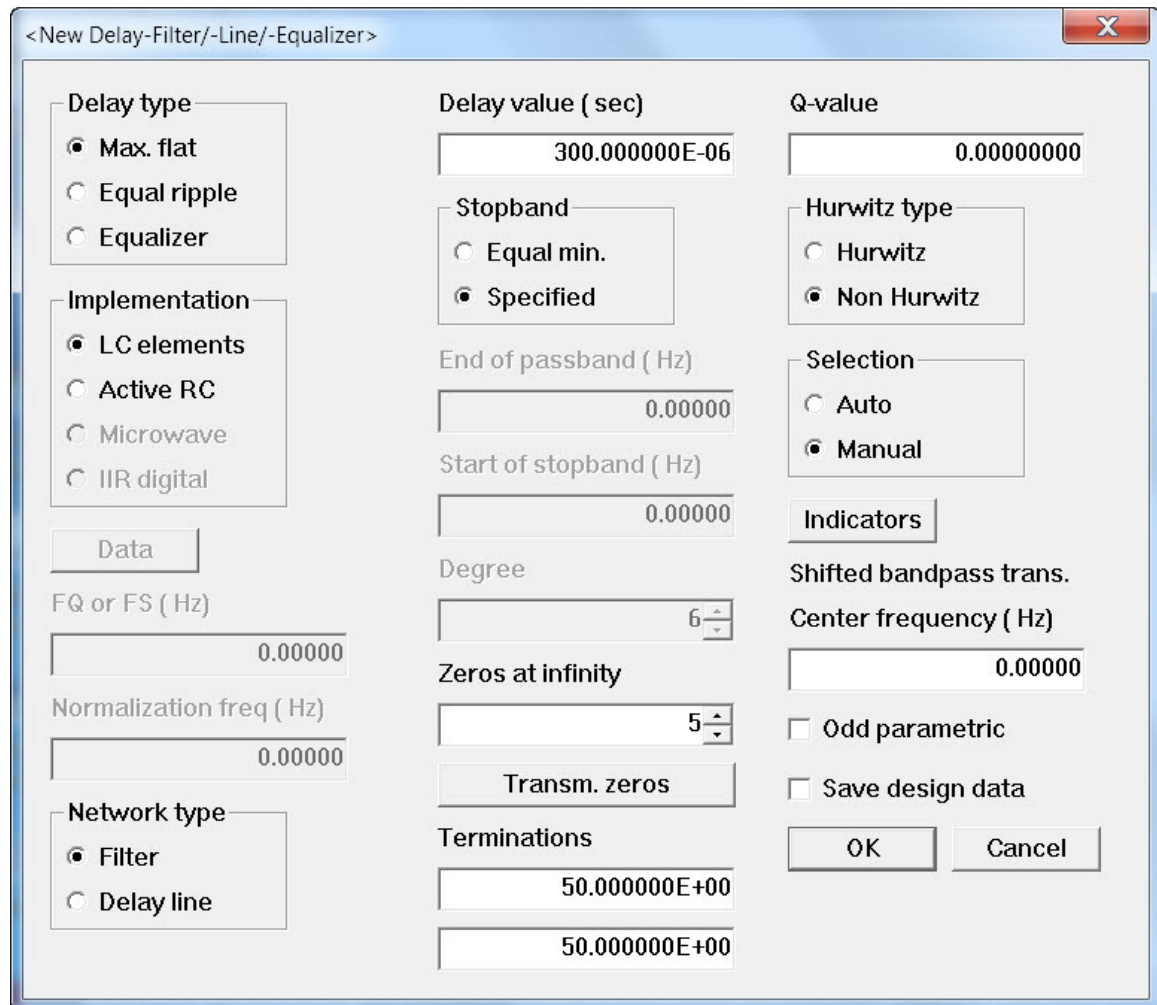
A design may easily be repeated with only a few data items changed, if we save the data using the last option on this screen. Since the number of items is small, we did not avail ourselves of this option here.

The synthesis yields a circuit that is quite different:

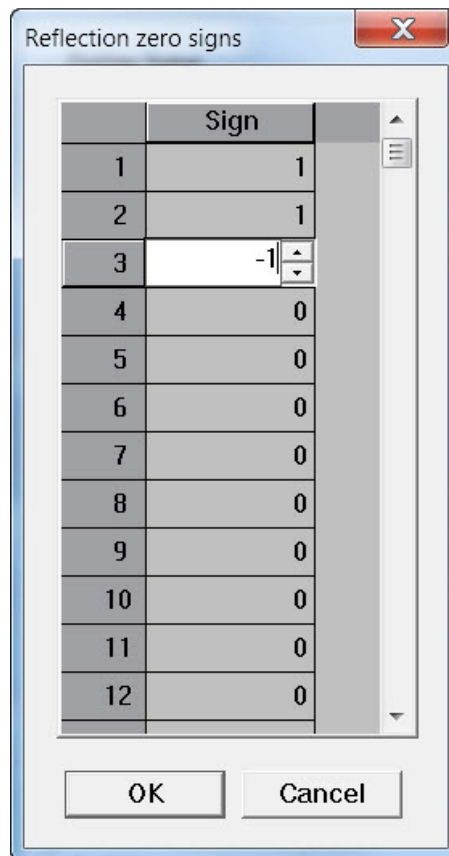


E.3 User-specified $F(s)$ zero distribution

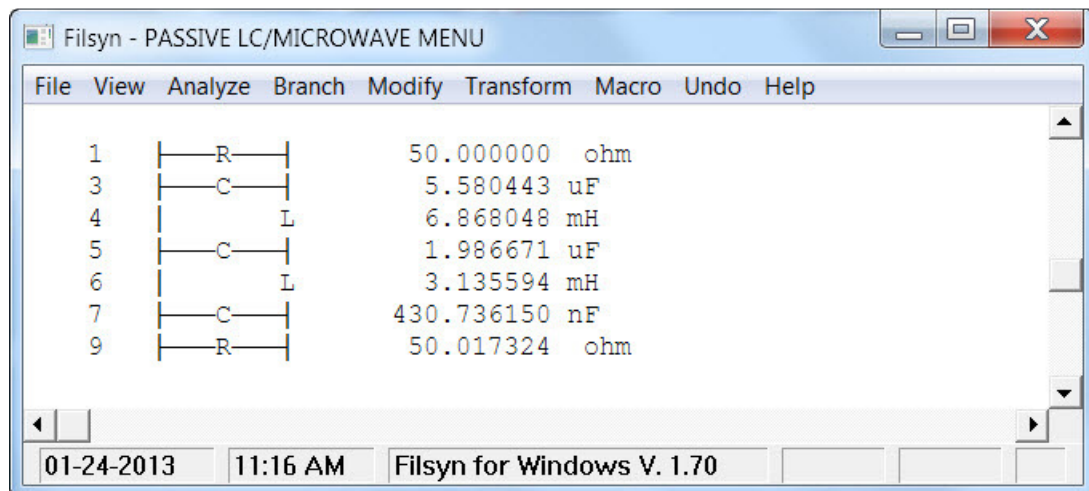
Lastly, we redesign this filter again, this time specifying the signs explicitly.



$F(s)$ is of 5th degree and has one real and two pairs of complex conjugate zeros, i.e. three distinct real parts. Therefore we need three signs to indicate their locations (+1 for the right half plane, -1 for the left). Clicking on the **Indicators** button, we enter:



The structure obtained is different again, although only slightly, since the filter degree is very low:



All three networks exhibit identical loss, phase and delay characteristics, but their output impedances differ. This feature is recommended, with the default algorithm for simplicity, for all **functional input** design cases.

E.4 Modified Bessel filter

Our next example illustrates another case, where control of $F(s)$ is very useful. This is a *Bessel* filter combined with an equal minima type stopband. The Hurwitz option, or even

the non Hurwitz option with the default sign combination does not yield a realizable computer-generated configuration. One could find an implementation with all positive elements, but with more than the minimal number of components. Nevertheless there *is* a minimal, realizable ladder and can be obtained as follows. The specification is shown in the next screen:

<New Delay-Filter/-Line/-Equalizer>

Delay type

- ☒ Max. flat
- ☐ Equal ripple
- ☐ Equalizer

Delay value (sec)

300.000000E-06

Q-value

0.00000000

Implementation

- ☒ LC elements
- ☐ Active RC
- ☐ Microwave
- ☐ IIR digital

Stopband

- ☒ Equal min.
- ☐ Specified

Hurwitz type

- ☐ Hurwitz
- ☒ Non Hurwitz

Data

FQ or FS (Hz)

0.00000

Normalization freq (Hz)

0.00000

Selection

- ☐ Auto
- ☒ Manual

Network type

- ☒ Filter
- ☐ Delay line

End of passband (Hz)

0.00000

Start of stopband (Hz)

1.800000E+03

Degree

6

Zeros at infinity

0

Transm. zeros

50.000000E+00

Terminations

50.000000E+00

Indicators

Shifted bandpass trans.

Center frequency (Hz)

0.00000

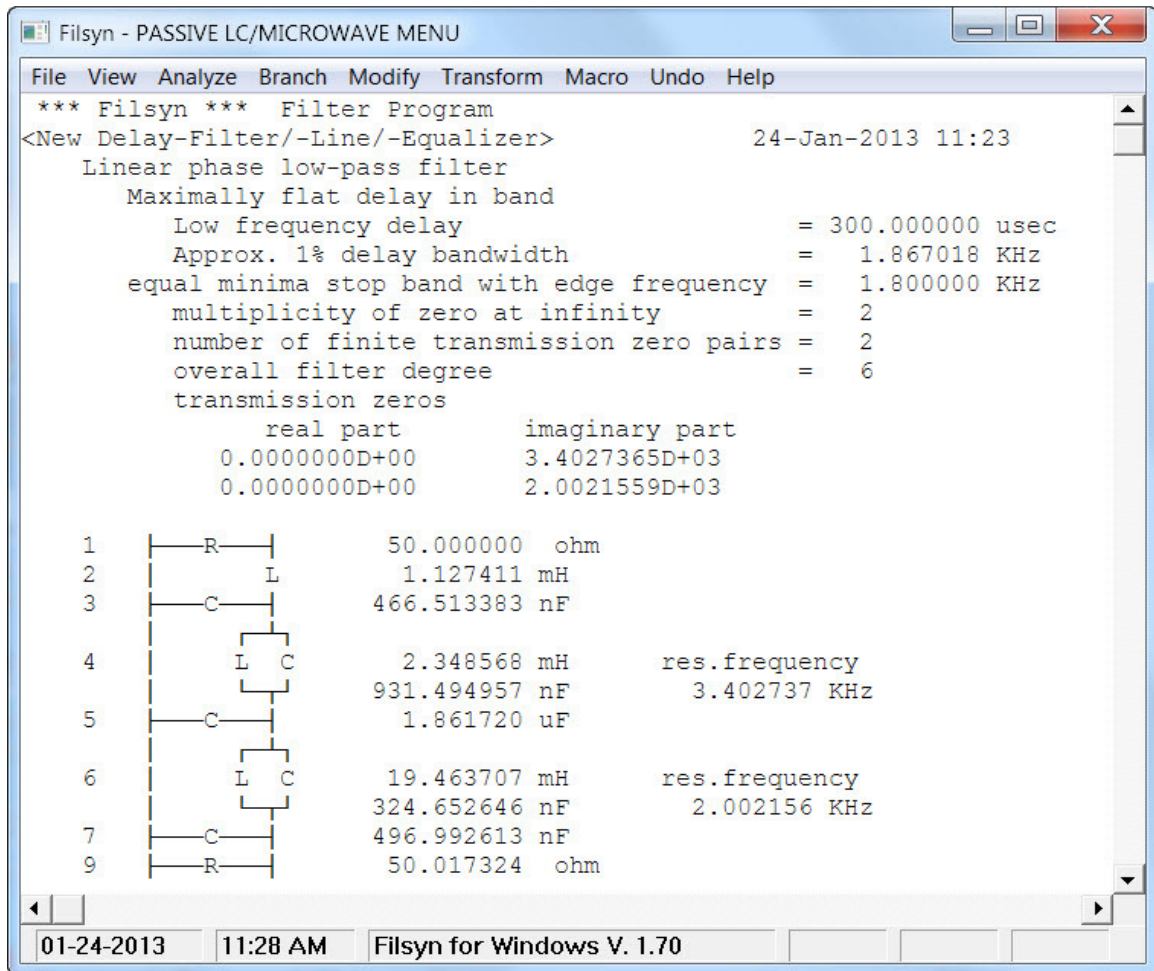
☐ Odd parametric

☐ Save design data

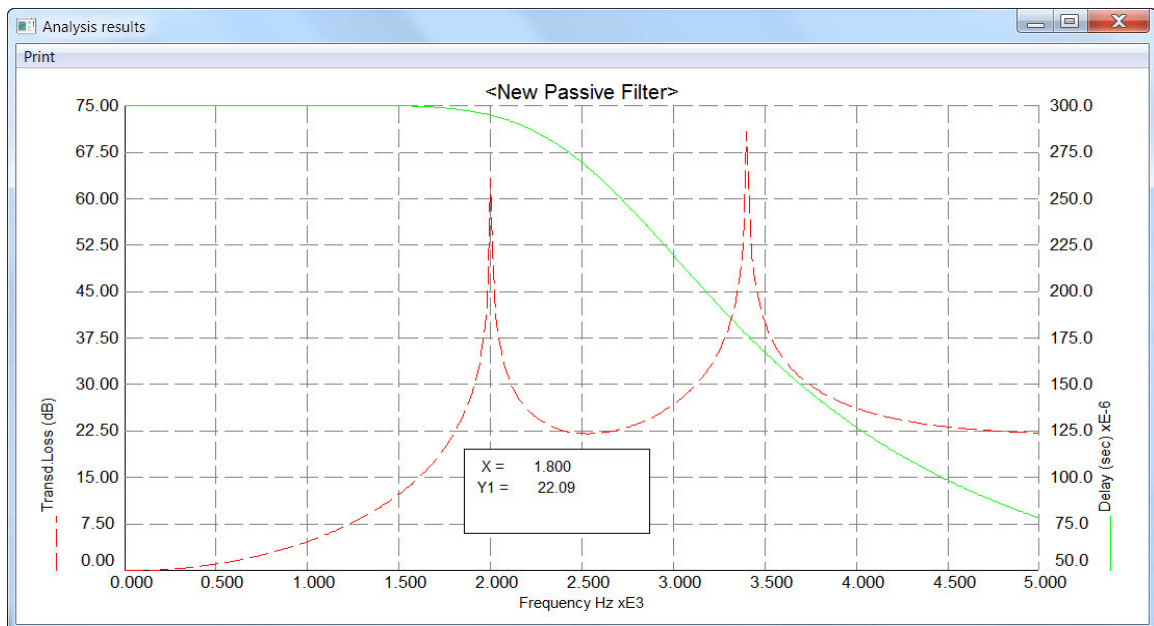
OK Cancel

The sign sequence $-1, 1, -1$ – was found by trial and error. In a low-degree case it is simple to try all combinations of signs (actually only half of them, because changing *all* the signs will just turn the network end-for-end), but this could become cumbersome with increasing degree. Nevertheless, currently no other way exists to handle this situation.

Proceeding, we find that the computer-generated configuration exists with minimal number of all positive components:



The analysis of this circuit yields the expected response:



which shows that we obtained a minimum stopband loss of about 22.1 dB.

APPENDIX F

PREDISTORTION

Because inductors are inherently lossy components, the effects of this dissipation can be taken into account in the design. This dissipation has an effect both on the passband and on the stopband behavior of the filter. In the passband the loss increases towards the edge(s) of the passband, while in the stopband(s), the loss does not rise as fast and as high as expected in the area near the passband.

We can compensate (approximately) for these effects. The stopband impairment may be counteracted using the **Placer** option by shifting the locations of the transmission zeros, while the passband rounding can be compensated for by the use of predistortion in the program.

This appendix demonstrates both of these procedures and their effects on a simple lowpass filter. They assume that all inductors have the same constant, frequency independent, dissipation factor, which is an approximation, because these quantities are usually functions of the frequency. More accurate compensation can only be achieved by loss equalizers, or touchup optimization.

Compensating for the stopband effects of dissipation is harmless, but predistortion in the passband is not without its side effects. One is the unequal terminations resulting in the low- or high-pass filter cases, the other, and more serious one, is the deterioration of the return loss of the predistorted filter. Sensitivity to component variations will also be affected. Therefore, if return loss requirements are specified or a very stable filter is needed, predistortion is *not* recommended.

F.1 Elliptic Lowpass

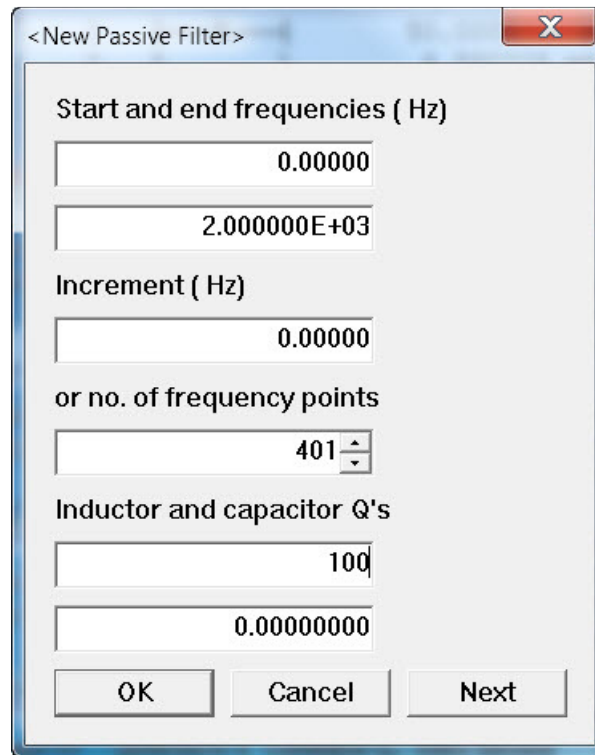
First we design an elliptic lowpass with no concern for the inductive dissipation to see the effect on the final results. The specifications are as follows:

The summary shows that we need a 10th degree filter:

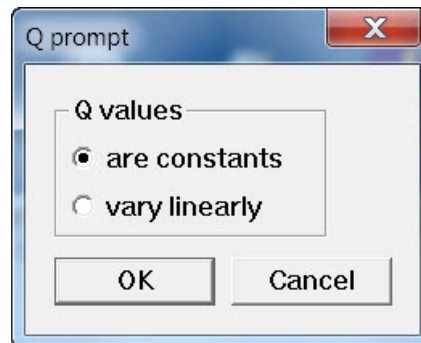
Component	Value	Unit	Notes
1	50.000000	ohm	
2	8.592208	mH	
3	4.385278	uF	
4	11.810847	mH	res.frequency 1.674560 KHz
5	764.817891	nF	
6	4.068665	uF	
7	8.172637	mH	res.frequency 1.212429 KHz
8	2.108459	uF	
9	2.766657	uF	
10	3.914074	mH	res.frequency 1.053550 KHz
11	5.830432	uF	
12	2.434869	uF	
13	2.863799	mH	res.frequency 1.089253 KHz
14	7.454871	uF	
15	485.957598	nF	
16	50.000000	ohm	

The analysis (not shown) indicates that we have about 3.5 dB loss to spare in the stopband, a marginal amount.

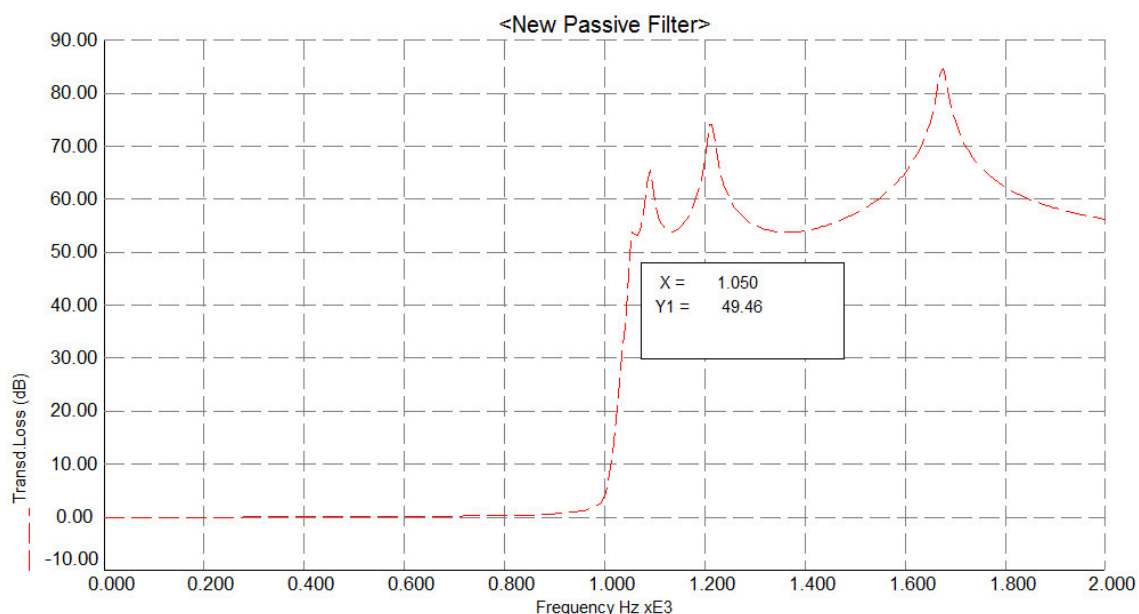
Next check the filter performance with ideal capacitors and lossy inductors with a Q value of 100 at the edge of the passband:



Assume that this inductance Q is a linear function of the frequency (the dissipation factor is constant):

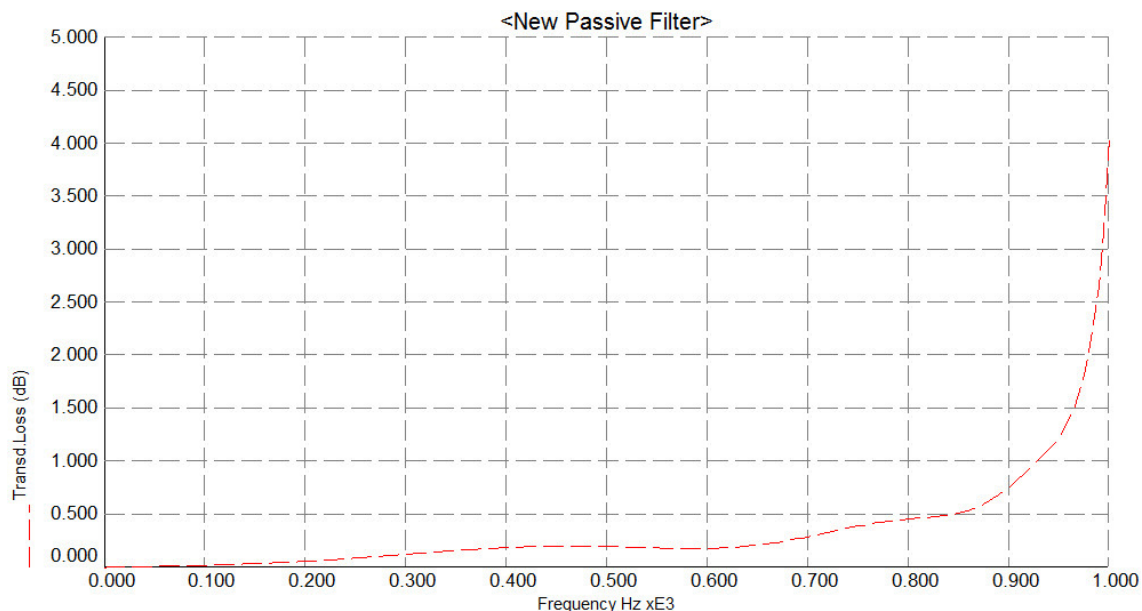


The results, displayed graphically are as follows:



One comment is in order about this design. Because it is of an even degree and the terminations are specified as equal, the transmission zeros have been shifted since the ZS parameter under these circumstances will be set to be 1.

This result is unsatisfactory. The passband edge loss is about 4 dB (see the passband detail below) and the loss at 1050 Hz barely makes the requirement. We must therefore redesign this filter, and we may find that the filter degree must increase to provide more loss to spare.



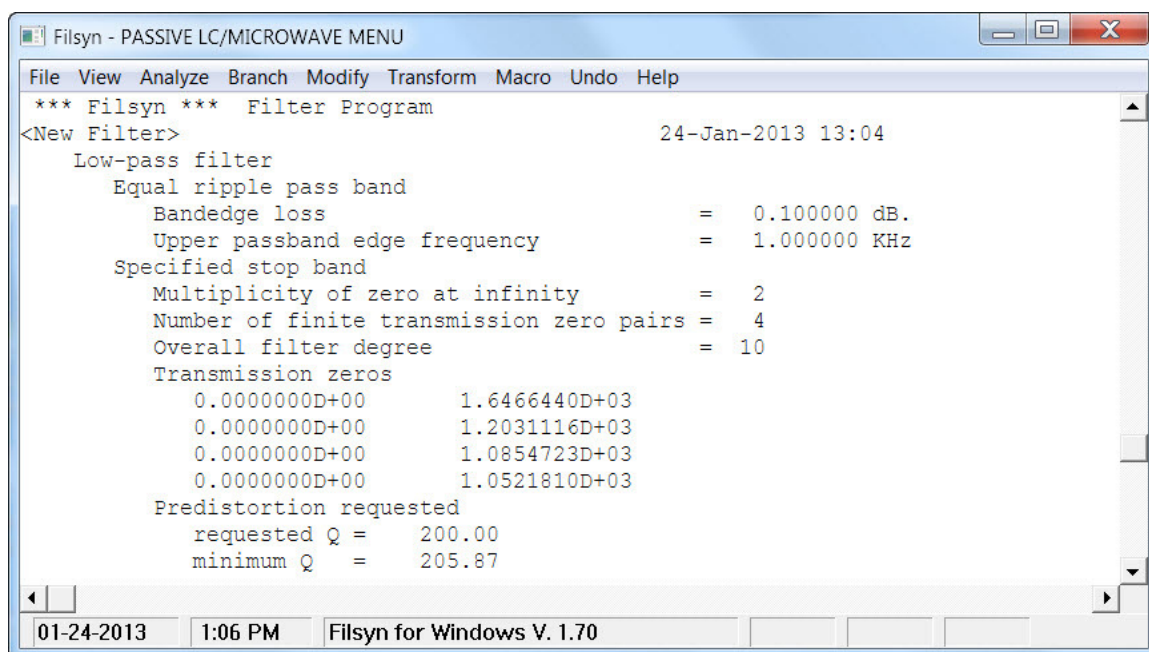
F.2 Predistorted Elliptic Lowpass

If we also wish to use the stopband compensation method, we must use the **Placer** option and we specify a 10th degree filter as follows. The **Detail** screen specifies the Stopband as

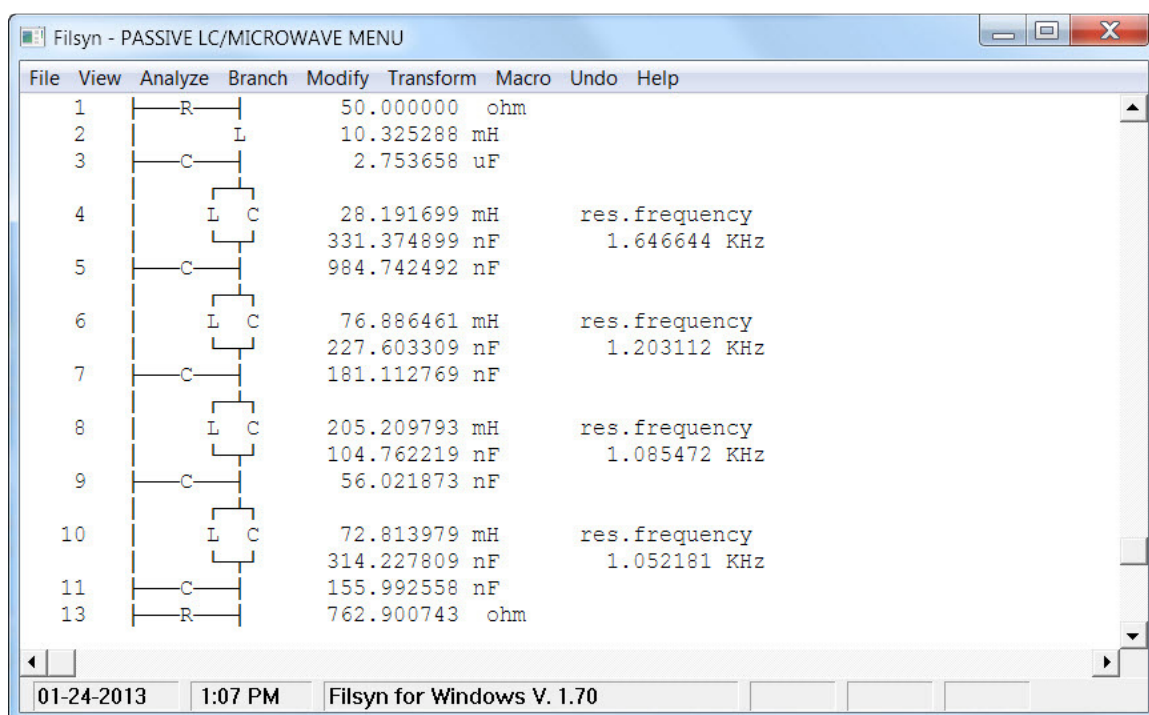
starting at 1.05 KHz and 50 dB required loss and also specifies four transmission zeros, which, together with the two at infinity, makes this a 10th order filter.

A comment or two are in order. First, the output termination is selected to be slightly different from the input one. This has the effect of setting the ZS parameter equal to 0 instead of 1. Consequently the transmission zeros will *not* be shifted, as they were in the elliptic case above. Second, the Q value used is the inverse of the average of the dissipation factors of the inductors and capacitors. Because the capacitors are assumed ideal, we specify *twice* the given inductor Q. Third, the **Predistortion** entry offers the opportunity to have **Placer** compensate for the lossy effects in the stopband as well as the passband. We wish to do both. The two procedures will use the same Q value.

The design summary below shows that we requested predistortion and that the minimum possible value of Q is a bit higher than what we requested. In this case the requested value is automatically replaced by the minimum permissible value. If the difference is substantial, we will be asked if we wish to do that, otherwise the program will *not* perform the predistortion.

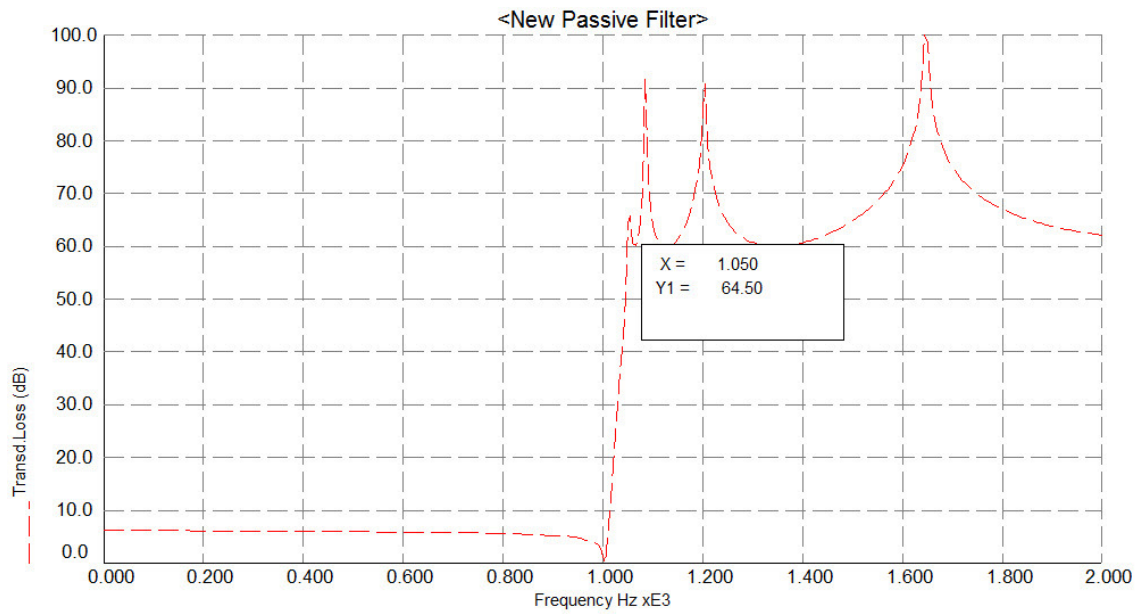


Next we also note that the terminations are quite different and this is due to the predistortion. The design summary outlines this clearly:

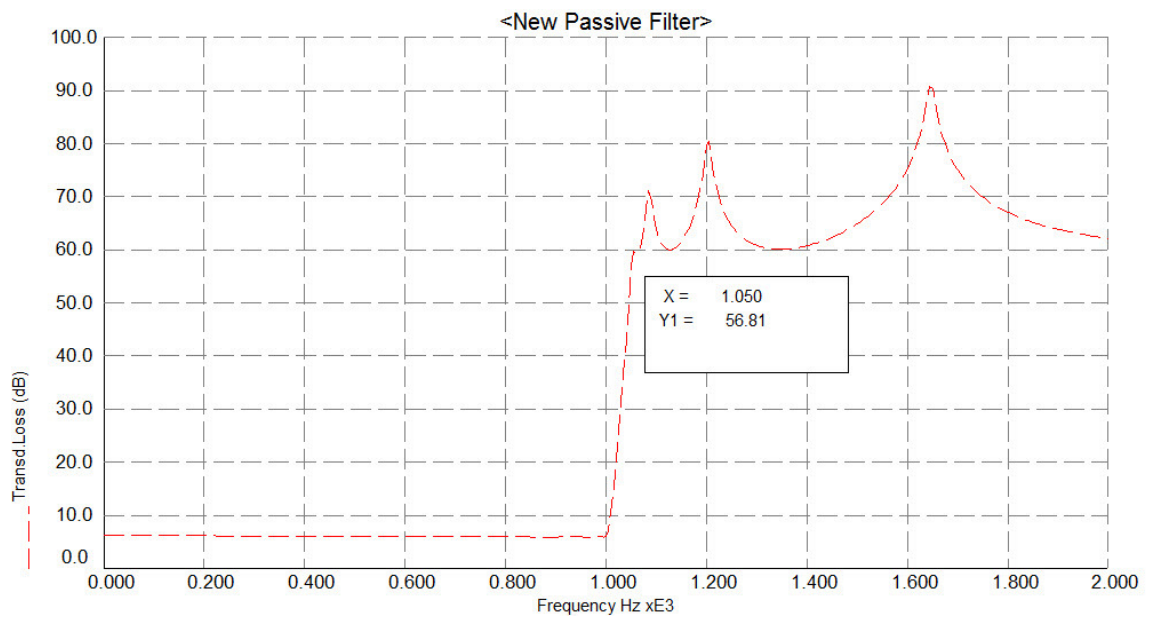


A frequency scan shows graphically, what predistortion does to the ideal (lossless) behavior:

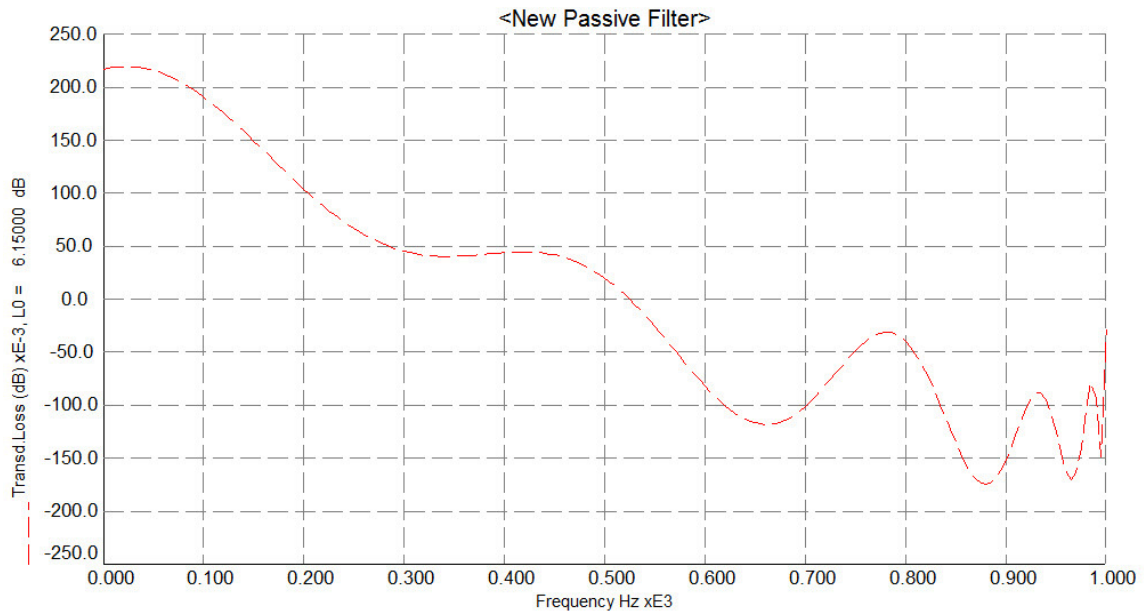
Predistortion



On the other hand if we include the dissipation in the analysis, we get a much better performance:



The passband loss is now quite flat as shown in detail below (note the vertical scale):



This shows an overall passband loss variation of a shade under 0.4 dB, not quite the 0.1 dB theoretical value, but substantially improved from the 4 dB variation in the non-predistorted case. The stopband is also better, although not quite ideal. The stopband corner is a bit higher than 1.05 KHz, but the loss at 1.05 KHz is still acceptable. If we wish, the design could be repeated with a stopband edge frequency specified to be a bit lower than 1.05 KHz.

Note that predistortion should *not* be used for linear-phase lowpass filters. The phase and hence the delay characteristics are *not* preserved by this operation, therefore the resulting filter will not be linear-phase any longer. Neither should this be used in the case of singly terminated designs – used for multiplexers – since the input impedance will also be modified.

Microwave filters may not be predistorted.

APPENDIX G

DEFINITIONS AND USEFUL RELATIONSHIPS

G.1 Popular filter functions

A) Butterworth filters

These are the earliest known filter types and the least efficient. Their passband loss is maximally flat, their stopband loss is monotonic (i.e., all the transmission zeros are at extreme frequencies). Fig. 25 shows the basic low-, high- and band-pass characteristics. One can use the following expressions to solve for the degrees needed to meet your requirements:

$$A = (10^{a_{\min}/10} - 1)/(10^{a_{\max}/10} - 1) \quad (G.1)$$

and

$$n > (\log A)/(2 \log \Omega_s) \quad (G.2)$$

where

$$\Omega_s = \begin{cases} f_{BS}/f_B & \text{for lowpass} \\ f_A/f_{AS} & \text{for highpass} \\ \min [(f_{BS}^2 - f_A f_B)/(f_{BS}(f_B - f_A)), (f_A f_B - f_{AS}^2)/(f_{AS}(f_B - f_A))] & \text{for bandpass} \end{cases} \quad (G.3)$$

and a_{\max} is the maximum allowable passband loss and a_{\min} is the minimum required stopband loss; f_A (and f_B) are the passband edge frequencies, while f_{AS} (and f_{BS}) are the stopband edge frequencies.

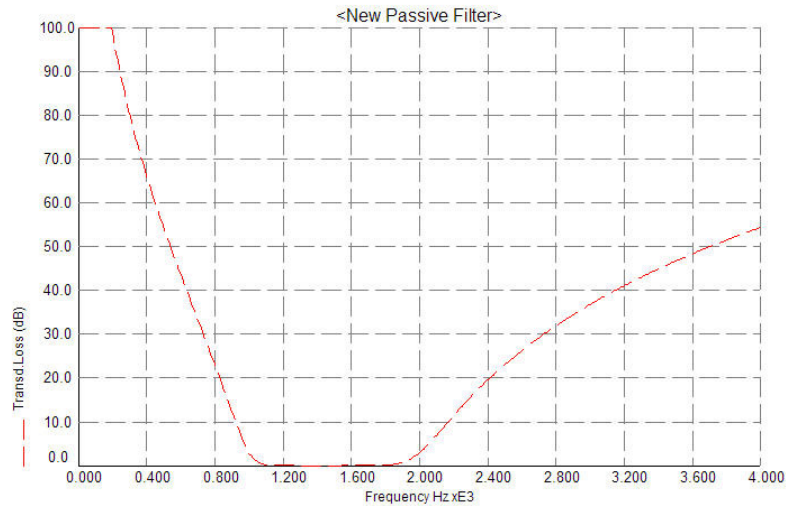


Fig. 25 Butterworth bandpass filter

In the bandpass case, n must be doubled. The number of zeros at zero and at infinite frequencies are equal in the bandpass case and this yields a bandpass filter that is symmetrical on a logarithmic frequency scale. If arithmetic symmetry is required, one can obtain better results by selecting:

$$n_{\infty} = 3 n_0$$

but then the filter is not strictly a Butterworth filter anymore and the above expressions will not hold either.

B) Chebyshev filters

These are filters with equal ripple type passband and monotonic stopband(s). Their characteristics are shown in Fig. 26 and the corresponding degrees are obtained from the expression:

$$n > \log[A^{1/2} + (A-1)^{1/2}] / \log[\Omega_s + (\Omega_s^2 - 1)^{1/2}] \quad (\text{G.4})$$

where A and Ω_s are given by equations G.1 and G.3 respectively and n must again be doubled for the bandpass case. Our comment about the symmetry of the Butterworth bandpass filters applies to this case as well.

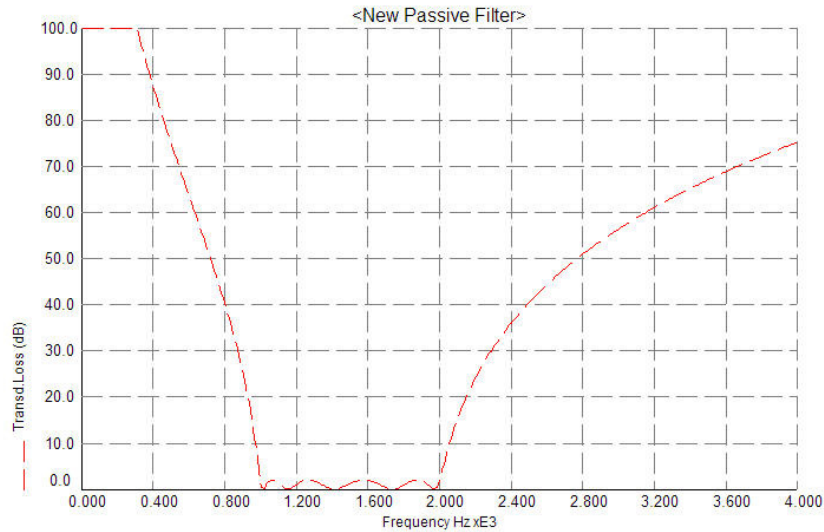


Fig. 26 Chebyshev bandpass filter

C) Inverse Chebyshev filters

These filters are not as popular as the previous two since their realization is as complex as that of the elliptic filters (see below), however, they are not as efficient. They have a maximally flat passband and equal minima type stopband(s) (see Fig. 27). Their degrees are calculated by exactly the same expressions as those used for the Chebyshev filters.

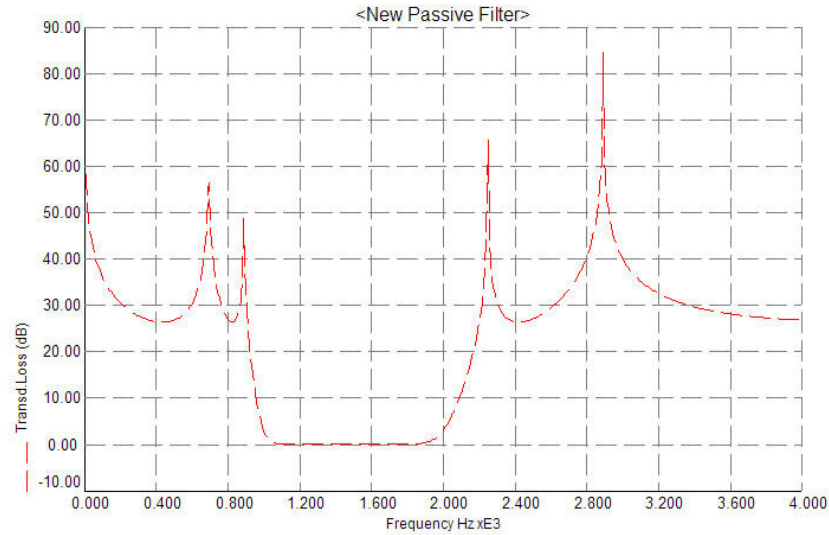


Fig. 27 Inverse Chebyshev bandpass filter

D) Elliptic filters

For uniform passband and stopband requirements, these filters provide the most efficient design. Their passband is of the equal ripple type and the stopbands are of the equal minima type (see Fig. 28). The required degrees can be calculated from the approximate - but very accurate -- expression:

$$n > F(B) \cdot F(C) \quad (G.5)$$

where

$$F(x) \cong (\ln(x + 2x^5 + 19x^9))/\pi$$

and

$$B = 0.5 [(\Omega_s^{1/2} - 1)/(\Omega_s^{1/2} + 1)] \quad (G.6)$$

$$C \cong [1 + 1/2A]/16A \quad (G.7)$$

A and Ω_s are defined as before. The value of n must be doubled again for bandpasses and if n is even and ZS is not specified as -1, equation G.5 above may underestimate the value of n by one. The program however, corrects for this effect.

Occasionally, the loss values a_{\min} and a_{\max} (hence also A) are specified, together with the degree n . One can then approximately calculate the corresponding Ω_s through the following set of expressions:

$$\begin{aligned} D &= e^{(\pi^2 n / \ln C)} \\ D' &= D / (1 + 2D^4) \\ \Omega_s &\cong [(1 + 2D') / (1 - 2D')]^2 \end{aligned} \quad (G.8)$$

where C is given by equation G.7 above. Again if n is even and ZS is not equal to -1 , this expression underestimates Ω_s slightly.

In fact, if we know any three of the four parameters a_{\min} , a_{\max} , Ω_s and n , we can easily calculate the missing value.

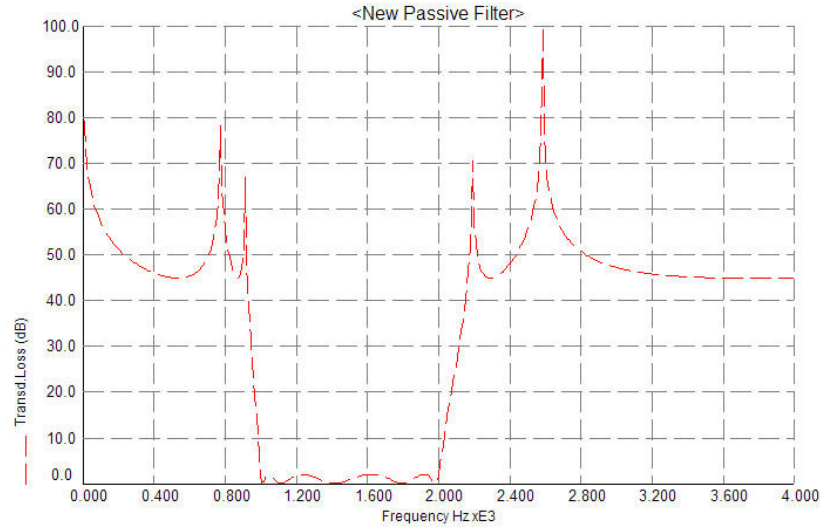


Fig. 28 Elliptic bandpass filter

E) Bessel filters

These are lowpass filters with maximally flat delay in their passband and a loss that is approximately Gaussian (the loss is proportional to $\ln \omega^2$). See also Fig. 29. There is a semi-empirical relationship between the delay at zero frequency τ_0 and the 3 dB loss point f_{3dB} given by:

$$2\pi\tau_0 f_{3dB} \cong [0.693(n-1)]^{1/2} - 0.788/(6.5+n) \quad (G.9)$$

However, this is valid only if all transmission zeros are at infinity. In any other case f_{3dB} will become lower than that obtained from the equation above.

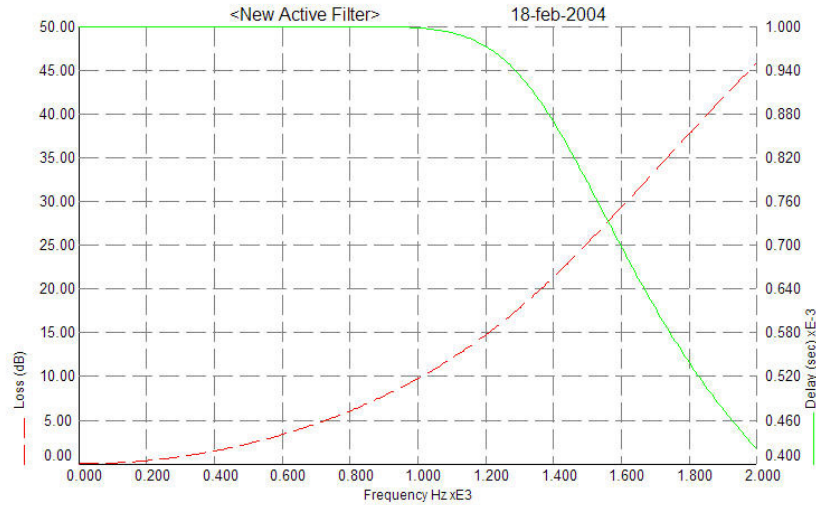


Fig. 29 Bessel filter

F) Equal-ripple delay filters

One can also approximate the constant delay of a lowpass filter in the equal-ripple sense. This uses an iterative procedure, because no closed form solution has been found. The passband loss is monotonic for low ripple values, but becomes more complex for larger values of the ripple.

The relationship between the degree n , the value of the ripple r in % and the delay τ_0 times the ripple-bandwidth ω_0 in rad/sec can be represented by the following empirical expression:

$$\tau_0 \omega_0 \cong 1.47 n - 2.94(1 - 1.62/n^2) + [0.252 n / (1 + 0.105 n)] \log_{10} r \quad (\text{G.10})$$

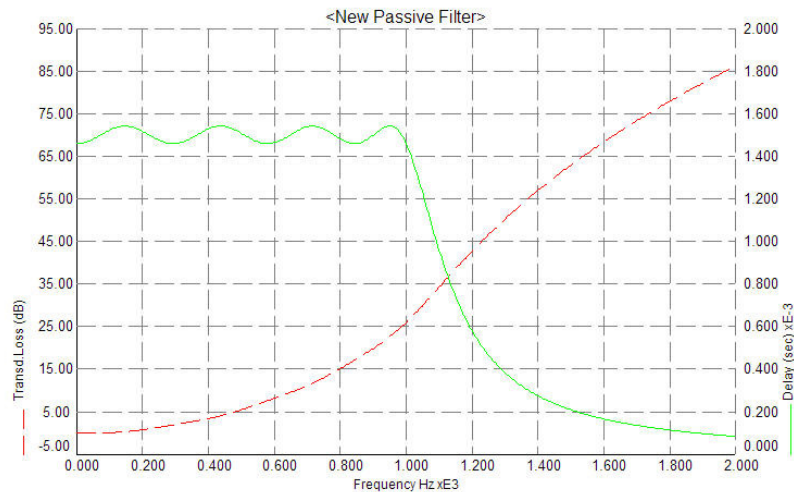


Fig. 30 Equal ripple delay filter

G.2 Other useful relationships

A) Return loss

The return loss in dB of a filter terminated by a resistance R_1 is defined by:

$$a_{\text{ret}} = 20 \log_{10} |(R_1 + Z_{\text{in}})/(R_1 - Z_{\text{in}})| \quad (\text{G.11})$$

where Z_{in} is the impedance looking into the filter from the termination. For lossless, doubly terminated filters a_{ret} is directly related to the filter loss by the relationship:

$$\frac{10^{-a/10}}{10^{-a_{\text{ret}}/10}} = 1 \quad (\text{G.12})$$

Therefore, in the passband, the minimum return loss can be obtained from a_{max} by solving equation G.12 for it:

$$a_{\text{ret min}} = -10 \log_{10} [1 - 10^{-a_{\text{max}}/10}] \quad (\text{G.13})$$

If a return loss requirement exists in the passband, one can obtain the allowable a_{max} from this expression and this value will usually be more restrictive, than a passband ripple. Also, dissipative effects will influence the return loss, but not very seriously. On the other hand, predistortion ruins the return loss completely, therefore it should not be used if return loss requirements are specified. The same is true if equal terminations are required in lowpass or highpass filters.

B) Voltage Standing Wave Ratio

The voltage standing wave ratio (VSWR) in microwave filters is also related to the impedance (Z_{in}) and termination (R_1) values through the expression:

$$\text{VSWR} = [|R_1 + Z_{\text{in}}| + |R_1 - Z_{\text{in}}|]^2 / [4 R_1 (\text{Re } Z_{\text{in}})] \quad (\text{G.14})$$

If the VSWR is specified, the corresponding maximum passband loss value is given by:

$$a_{\text{max}} = 10 \log_{10} [(1 + \text{VSWR})^2 / (4 \text{VSWR})] \quad (\text{G.15})$$

C) Reflection coefficient

Another quantity which also describes the passband mismatch is the reflection coefficient, that is related to the passband loss through the expression:

$$\rho^2 = 1 - 10^{-a_{\text{max}}/10} \quad (\text{G.16})$$

D) Mismatch loss

In the case of even degree lowpass and highpass filters with unequal terminations ($ZS = 0$), the relationship between the ratio of terminating resistors and the loss (mismatch loss) at zero or infinite frequency respectively, is given by the relationship:

$$a_0 = 10 \log_{10} (r + 2 + 1/r)/4 \quad \text{where } r = R_1/R_2 \quad \text{or its inverse} \quad (\text{G.17})$$

Conversely, if a_0 is given, then:

$$r = [A + \sqrt{(A^2 - 1)}]^{a_0/20} \quad \text{where } A = 10 \quad (\text{G.18})$$

This relationship can be used to introduce a flat loss into low- and high-pass filters. One simply calculates the termination ratio from the required loss at zero (or infinite, for highpass filters) frequency and specify that at the time the filter requirements are entered.

For bandpass filters, this method will not work, since unequal terminations can be accommodated without any flat loss. We can, however, trick the program to let us enter a nonzero flat loss. We do this by specifying sloping passband with a low, say 0.00001 dB/octave, slope. The next data item will let us specify an additional flat loss. The disadvantage of this procedure is that we can only use the **Specified** stopband option and **Conventional** type.

E) Definition of functions

For analog as opposed to digital or microwave, filters the independent variable is the complex frequency:

$$s = \sigma + j\omega$$

where $\omega = 2\pi f$ and we usually normalize this to f_{norm} , where f_{norm} is the (upper) passband edge frequency (f_B for low- and band-passes, f_A for highpasses).

The transfer function is then defined as the ratio of the output response to input excitation:

$$H(s) = \text{response/excitation} = Q(s)/E(s) \quad (\text{G.19})$$

where $E(s)$ and $Q(s)$ are real polynomials in s and satisfy the following additional constraints:

- $Q(s)$ is either pure even or pure odd,
- $E(s)$ has zeros with negative (nonzero) real parts,
- degree of Q is not greater than the degree of $E(s)$,

- $|H(s)|_{s=j\omega} \leq 1$ (this condition may be removed for active RC or digital filters).

The zeros of $Q(s)$ are called transmission zeros, while those of $E(s)$ are the poles, also called natural modes, of the filter. Next we define the characteristic function $K(s)$ by:

$$K(s) = F(s)/Q(s) \quad (G.20)$$

where $F(s)$ has as its only condition that it must be a real polynomial of s and has no common roots with $Q(s)$. The characteristic and transfer functions are related through the expression:

$$H(s)H(-s) = 1/[1 + K(s)K(-s)] \quad (G.21)$$

or substituting the polynomials:

$$E(s)E(-s) = F(s)F(-s) + Q(s)Q(-s) \quad (G.22)$$

This is the polynomial form of the celebrated Feldtkeller's relationship (see equation G.12). Normally $Q(s)$ is given. If, in addition, we specify $F(s)$, then $E(s)$ can be calculated uniquely from equation G.22. If on the other hand, $Q(s)$ and $E(s)$ are given, $F(s)$ is *not* specified uniquely. Because $F(s)F(-s)$ is an even polynomial, it will either have double zeros on the $j\omega$ axis, in which case we allocate one of them to $F(s)$ the other to $F(-s)$, or it will have quadruplets of zeros. In this case the left-half pair may be allocated to $F(s)$, the other to $F(-s)$ or vice versa. Since this is true for each quadruplet, the higher the degree, the more different solutions exist.

In any case the polynomial $F(s)$ will contain the reflection coefficient zeros at one end of the circuit, while $F(-s)$ contains those at the other end. The reflection coefficient is defined, say, at the input side as:

$$\rho_{in} = (R_1 - Z_{in})/(R_1 + Z_{in}) = F(s)/E(s) \quad (G.23)$$

with a similar definition with $F(-s)$ for the output side. The return loss is clearly the negative logarithm of the magnitude of the reflection coefficient.

Finally, the normalized analog transformed frequency used in the case of digital or microwave filters is defined as (this is the normalized bilinear Z-transform in the digital case and the Richard's transformation in the microwave case):

$$S = \tanh [s/2f_s]/\tan[\pi f_{norm}/f_s] \quad (G.24)$$

where S is the transformed frequency used to print the transfer function, and is to be used for functional input. Furthermore, f_s is either the sampling frequency for digital filters, or twice the quarter-wave frequency for microwave ones, while f_{norm} is the normalization frequency already defined above.

From the transfer function $H(s)$ we compute the loss α and phase β as follows:

$$\alpha = 20 \log_{10} |H(s)|^2 \quad (G.25)$$

and

$$b = \arg H(s) = \tan^{-1} \{Im[H(s)]/Re[H(s)]\} \quad (G.26)$$

where $s = j\omega = j2\pi f$ and Re and Im indicates the real and imaginary parts respectively. Finally the group delay is defined as:

$$\tau = -db/d\omega \quad (G.27)$$

G.3 Internal computations

The internal computations are done not in terms of the s variable; they are done in terms of a transformed one in the bulk of the program. This helps to maintain numerical precision. In addition, all polynomials are handled in terms of their zeros, rather than their coefficients. Furthermore in the microwave and IIR digital segments a two-step transformation is used because the familiar Richard's transformation (equation G.24) is also needed. All computations are done in double precision (about 14 decimal places) except for the computation of zeros of FIR digital filters, which is done in quadruple precision (about 32 decimal places). Filter degrees are restricted to 50, except FIR digital filters where the maximum length currently is 512.

In the digital segment we perform calculations directly in terms of the z variable. All frequencies are always normalized to the (upper) passband edge frequency or, in the microwave case, to the corresponding transformed frequency. In all cases, all polynomials are always represented in factored forms. In other words, we represent them by their roots and a multiplier instead of their coefficients.

The program runs very fast, response is nearly always instantaneous. Waiting for results may be encountered occasionally if one or more of the following situations are found:

- very high degree (over 25) filters (program limit is 50),
- predistortion,
- functional input,
- linear-phase lowpass with equal ripple delay.

The script files usually run much slower. This is mainly due to the desire to avoid any interference with other programs possibly running simultaneously on the computer. Each operation checks first if the correct window is open and ready for input before input is actually sent. This of course has some overhead that slows the computations somewhat.

APPLICATION NOTE 1

DIPLEXER AND MULTIPLEXER DESIGN

1. Introduction

The design of diplexers and multiplexers is a more difficult problem than that of single-input single-output filters. However, the **Filsyn** program greatly eases the design of multiplexers.

There are exact and approximate methods of diplexer design, but this is not the case for typical n-output multiplexers; here, only an approximate method is available. However, the exact method of diplexer design is used only infrequently. Both methods can best be explained by considering the behavior of filters terminated by a short-circuit at one end.

2. Short-Circuited Filters

We design filters short-circuited at one end (see Fig. 31) for a prescribed voltage loss given by:

$$a' [\text{dB}] = 20 \log_{10} |E / V_2| \quad (1)$$

where E is the source voltage of the ideal voltage source at the short-circuited end and V_2 is the voltage across the load at the terminated end.

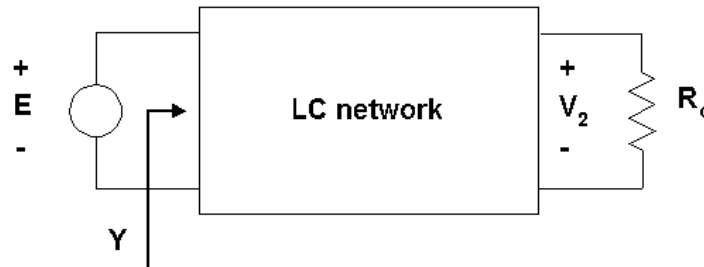


Fig. 31 Voltage-driven (short-circuited) filter

We denote the input admittance of the circuit facing the voltage source by:

$$Y = G + jB \quad (2)$$

A close connection exists between the real part G and the voltage loss a' . This relationship is due to the fact that the filter itself is lossless and any power absorbed by the load, must come from the source. The source delivers this power into the conductance G and therefore:

$$P_{\text{in}} = |E|^2 G = P_{\text{out}} = |V_2|^2 / R_o \quad (3)$$

or using equation (1):

$$R_o G = |V_2/E|^2 = 10^{-a'/10} \quad (4)$$

This equation indicates that in the passband where a' is small, G is approximately equal to $1/R_o$ and in the stopband, where a' is large, G becomes very small.

3. Exact Constant Resistance Filter Pairs

Using the previous derivation, we can design an exact constant resistance diplexer by designing two filters, a lowpass and a highpass, both short-circuited at their input and with voltage losses a'_L and a'_H respectively. If these loss functions satisfy the relationship:

$$\frac{-a'_L/10}{10} + \frac{-a'_H/10}{10} \equiv 1 \quad (5)$$

then the filters are called *complementary*. If both filters are terminated by the same load resistance (see ref.[34]), then paralleling them at their short-circuited end will result in an input conductance:

$$G = G_L + G_H \equiv 1/R_o \quad (6)$$

at all frequencies. If the input admittances of the individual filters were of the minimum-susceptance types (which is the only type the program can generate), equation (6) forces the imaginary parts of these admittances to cancel each other exactly and therefore, yield an input admittance, that is real, constant, and is equal to the common load of the individual filters. As a result, we can replace the voltage source by a source with an arbitrary resistance. In particular, we use a resistance equal to R_o , yielding an arrangement with an ideally perfect match at the common input port.

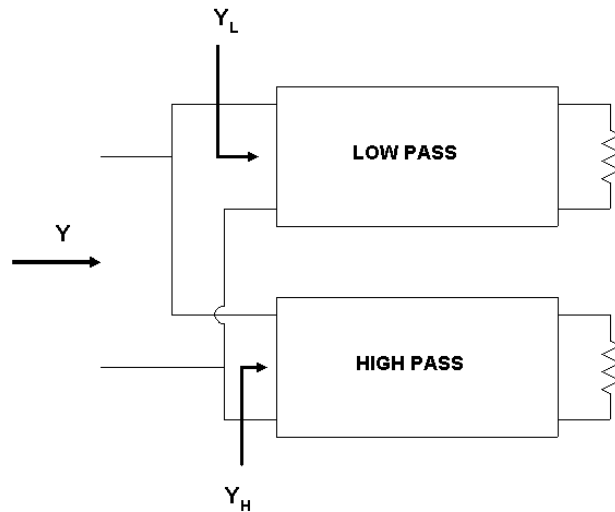


Fig. 32 Paralleled voltage-driven filter pair

The final transducer loss can be evaluated under these conditions by substituting the admittances Y_L and Y_H for the filters and recognizing that their imaginary parts cancel and the circuit will look like Fig. 33 below.

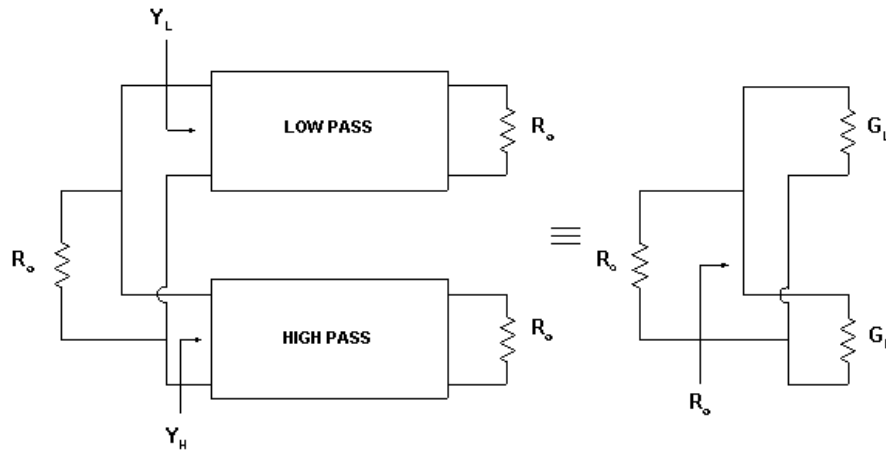


Fig. 33 Terminated filter pair and its equivalent

It is easy to show that the transducer loss through, say, the lowpass filter is given by:

$$a_L = -10 \log_{10} G_L R_o = a'_L \quad (7)$$

and similarly, the loss through the highpass is the same as the voltage loss the highpass filter was designed for. Also, the loop loss (the loss from the output of the lowpass to the output of the highpass) is just the sum of the losses of the two filters.

Equation (5) poses severe restrictions on the filters however. One possibility to meet them is to select both filters to be of the Butterworth types of equal degree and with coincident 3 dB loss points. But this is not a very useful filter combination. A better one is if both filters are of the elliptic type, again of the same degree and designed such that where one has a passband, the other has its stopband and vice versa. If, in addition, one filter has a passband ripple a_p and the other a minimum stopband loss a_s such that they satisfy equation (5), then again all requirements are met and they form an exact diplexer.

Even this is not a very satisfactory filter since any reasonable stopband requirements, say, 30 dB minimum, will correspond to a very low (in this case 0.0045 dB) passband ripple in the other filter, leading to an overdesign and possibly, to nonrealizable elements. In the even-degree filter cases one must also pay attention to the ZS parameter, which was one of the reasons we made this parameter available in the program.

4. Approximately Constant-Resistance Filter Pair

Satisfactory diplexers and multiplexers may be obtained by using the same argument used above, but completely ignoring the requirements imposed by equation (5) (see ref. [51], [26]). This enables us to design the two (or more) filters we intend to parallel, completely independently of each other, except that the passbands may *not* overlap.

Consequently, equation (6) will not be satisfied and neither will the transducer loss values be exactly the same as the voltage loss the filters were designed for.

We must therefore find the transducer loss and the common impedance of the overall structure under fairly general assumptions.

The structure used is shown in Fig. 34 below, where the shunt branch $Y_{\text{corr}} = jB_{\text{corr}}$ is added for the following reason. Because the conditions of equation (5) are not satisfied, the filter susceptances will cancel approximately, but not exactly.

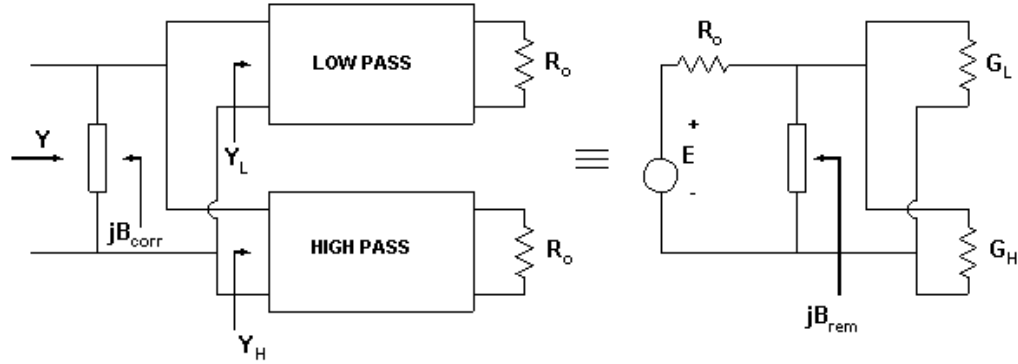


Fig. 34 Paralleled filters with susceptance corrector

The leftover susceptance may then be cancelled more accurately by this additional branch, which is usually nothing more than a single resonant branch. For multiplexers, this may contain several series resonant branches in parallel, each resonating in one of the transition regions. The actual design of these susceptance correctors can be done by trial and error methods, or by iterative optimization techniques, using a program such as *Spice* or *Touchstone*. Often an elementary, two-point matching will suffice. This branch will also increase the loop loss in the transition regions — a desirable feature.

Assuming that these susceptance correctors perform accurately and therefore the input admittance at the common port is essentially real, the situation will be that shown in Fig.4 above, where

$$B_{\text{rem}} = B_L + B_H + B_{\text{corr}} \quad (8)$$

is the uncanceled leftover susceptance and:

$$\begin{aligned} G_L R_o &= 10^{-a'_L/10} \\ G_H R_o &= 10^{-a'_H/10} \end{aligned} \quad (9)$$

In the passband of the lowpass, G_L varies between

$$10^{-a'_{Lmax}/10} < G_L R_o < 1 \quad (10)$$

We assume that B_{rem} and G_H are approximately zero. For instance, if the highpass has a minimum loss of 30 dB in the stopband, then:

$$R_o G_H < 0.001$$

which is always negligible, even in comparison with the ripple of G_L .

In the passband of the lowpass, the circuit therefore consists of a source of resistance R_o facing a load of value G_L . Using equation 10 we can evaluate the maximum transducer loss ripple as follows:

$$a_{Lmax} = 20 \log_{10} [(10^{a'_{Lmax}/20} + 10^{-a'_{Lmax}/20})/2] \quad (11)$$

It is noteworthy that a_{Lmax} is substantially lower than a'_{Lmax} . That is to say, the short-circuited filters may be designed with a much easier set of requirements. Allowances can be made for the neglected residual susceptance, based on experience.

In the stop band, the residual susceptance can still be neglected but the conductance of the highpass does enter the picture (see Fig. 35). Nevertheless, since the input conductance of the lowpass is low, the transducer loss can be shown to be about the same as the voltage loss of the lowpass. The only region (or regions) where we cannot estimate the loss is the transition region between passbands. In this case a detailed analysis must be performed to get an accurate picture.

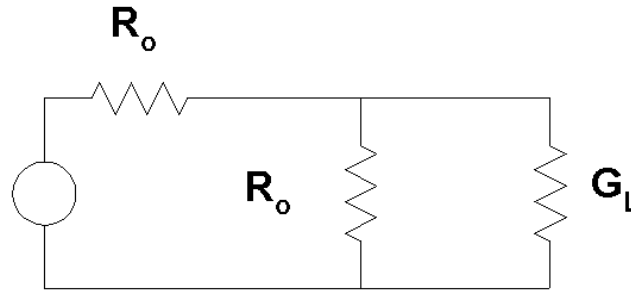


Fig. 35 Stopband equivalent

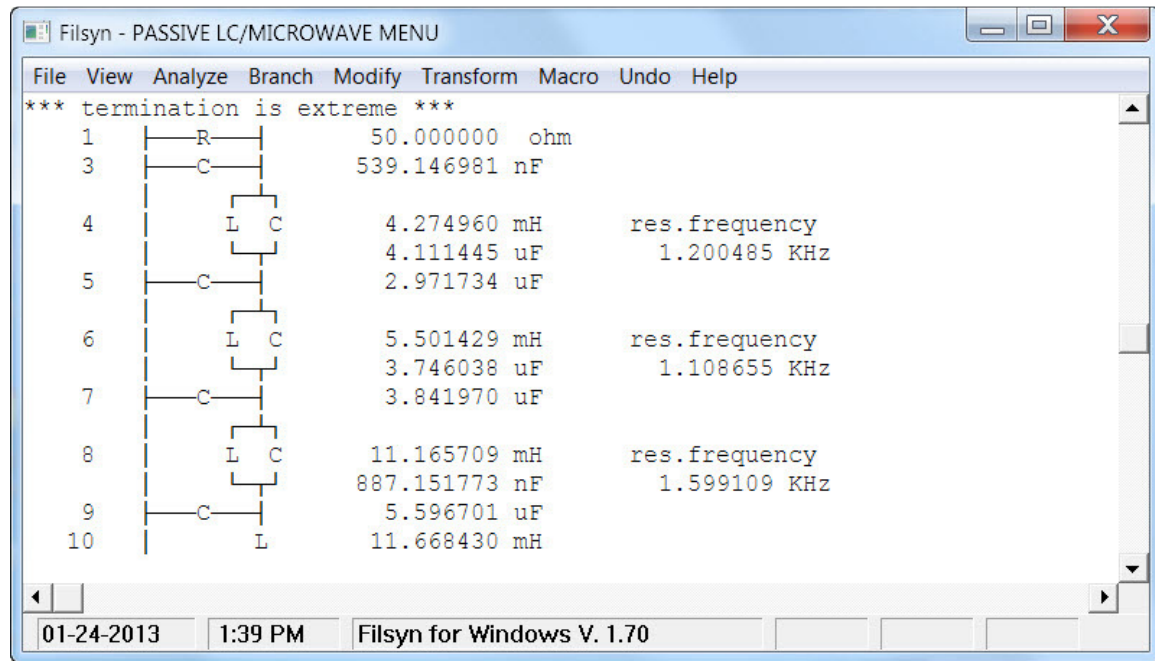
5. Example

The procedure will be illustrated by a lowpass-highpass diplexer design.

We select an elliptic lowpass with transition band from 1 KHz to 1.1 KHz and a 0.5 dB passband ripple. This ripple will be reduced, according to equation 12 above, to about 0.0144 dB. We request a 50 dB stopband loss, that yields an 8th order lowpass. One can easily estimate the loss using the **Misc. filters->Estimate** menu item in the **filscript.exe** program. We need an even degree filter for the following reason. In order to parallel the two filters at their common input, the lowpass must start with a series branch, which of course, is an inductor. An even order lowpass will have fewer inductors than an odd

degree one. Furthermore, we need to specify a ZS parameter of +1 in order that the input conductance should have the correct value although this is not absolutely necessary. The highpass will have to be an odd order elliptic one for the same reason, and the ZS parameter here is immaterial, but the filter should have the same pass-band ripple and transition band.

The computer-generated realization is shown below. The output termination is absent and the fact that the last branch is a series one, indicates that the termination is a short-circuit because otherwise the series branch would have no role to play.



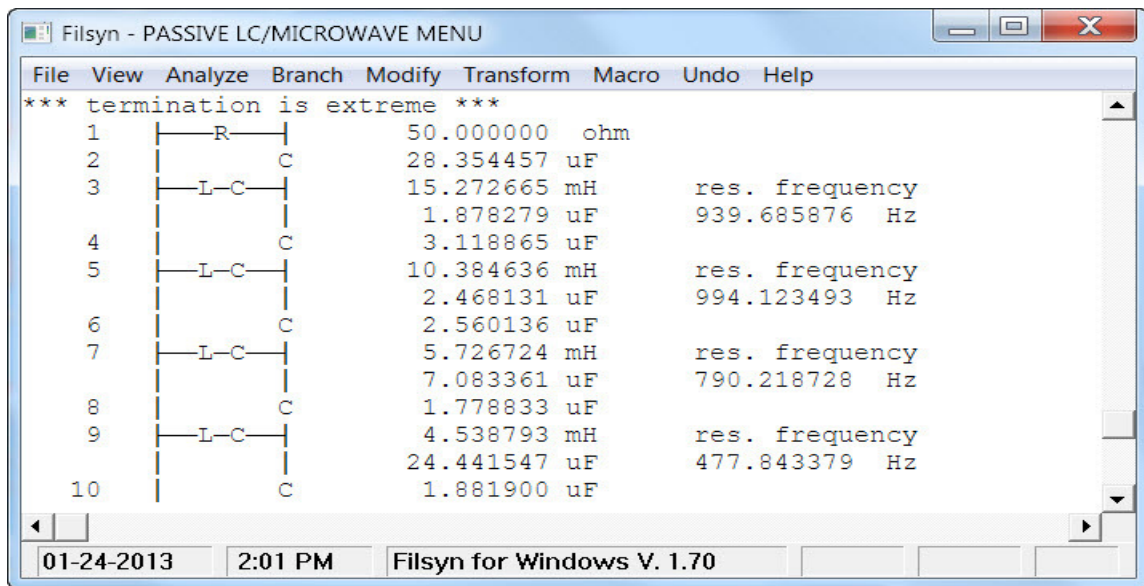
We also generate a file for the *Spice* program and show it below for later reference.

```

<New Passive Filter>
24-Jan-2013 13:45
vin 1 0 ac 2.0
r01 1 2 5.000000E+01
c02 2 0 5.391470E-07
l03 2 3 4.274960E-03
c03 2 3 4.111445E-06
c04 3 0 2.971734E-06
l05 3 4 5.501429E-03
c05 3 4 3.746037E-06
c06 4 0 3.841970E-06
l07 4 5 1.116571E-02
c07 4 5 8.871518E-07
c08 5 0 5.596701E-06
l09 5 6 1.166843E-02
r10 6 0 1.000000E-03
.ac lin 50 2.500000E+01 1.250000E+03
.print ac idb( 6, 0) ip( 6, 0)
.end

```

The highpass specification is similar and the circuit itself is shown below:



This was also written in a disk file in a form readable by *Spice*, not shown here.

Analyzing the two filters separately, they behave as expected. The real parts of their input conductances are shown below in Fig. 36. The values of the admittances in the stopbands are about 5 orders of magnitude smaller; therefore they are not visible on this scale. If we did not specify the $ZS = 1$ parameter, the Lowpass conductance would ripple between 20 mS -- milliSiemens, where 1 Siemens = 1/ohm -- and 22 mS, instead of the 18 mS and 20 mS that the highpass does. The highpass has 64.4 dB minimum stopband loss, while the lowpass has a 53.5 dB minimum rejection.

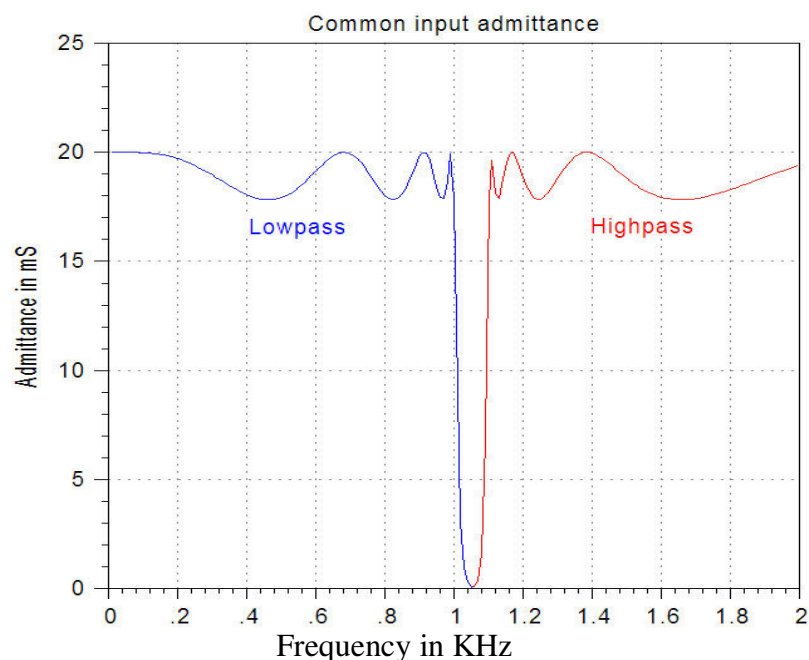


Fig. 36 Input conductances of the two filters

The input susceptances are more interesting and their sum is shown below in Fig. 37. Except for the transition region and its immediate vicinity, these susceptances cancel quite accurately. The sum is close to zero everywhere except near the edges of the passband. We can reduce this sum further by adding a shunt susceptance corrector branch consisting of a series LC resonant circuit. The element values may be determined such that they cancel the susceptance at the passband edge frequencies and this leads to a 95.0 mH inductor and a 242.5 nF capacitor. Adding the susceptance of this branch to the sum, we get the residual, shown in solid line in the same figure. The remainder is less than about 5.6 mS in magnitude in both passbands, which will cause an additional passband loss of less than 0.084 dB. Therefore the overall passband ripple will be less than 0.1 dB for both filters. The susceptance corrector has the further advantage of increasing the loop loss in the transition band. Note that in the figure we deleted some of the data in the transition region, because the values were very large and it would have made the figure difficult to see.

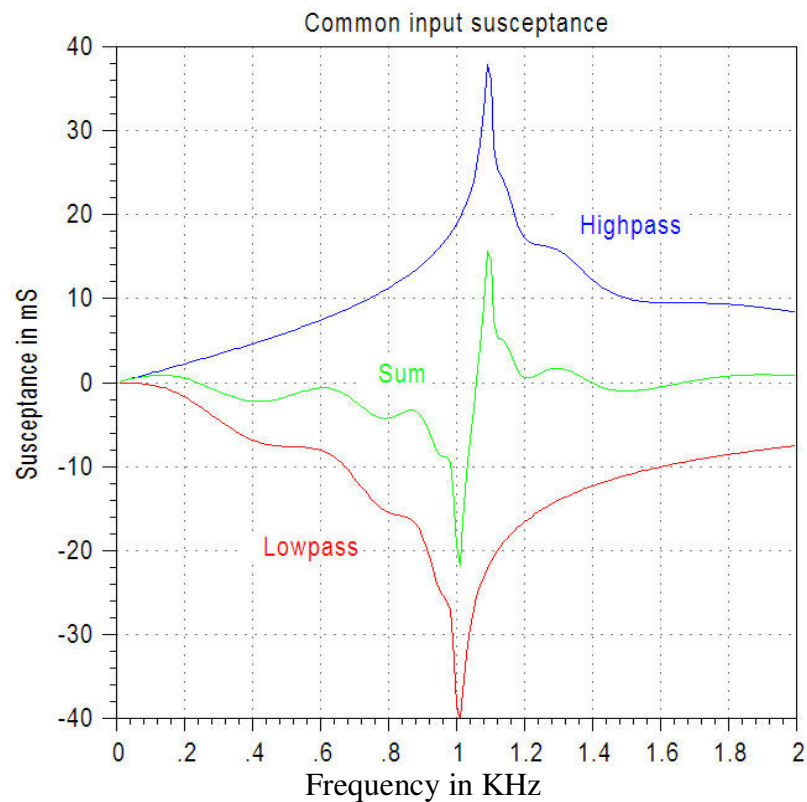


Fig. 37 Input susceptances, and their sum

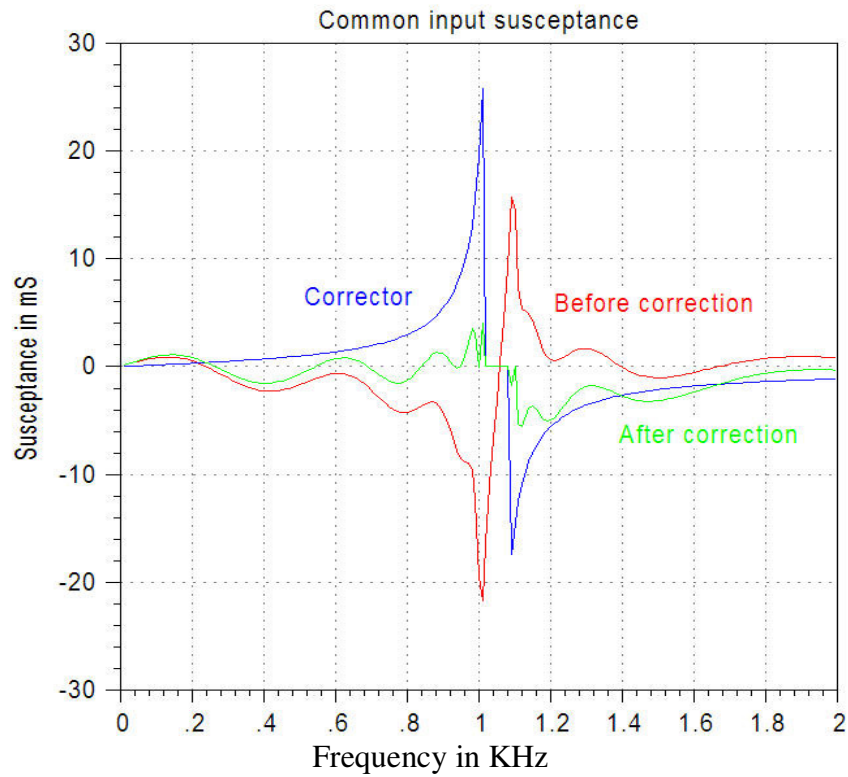


Fig. 38 Residual susceptance, the corrector and corrected susceptance

To check the correctness of these approximate calculations, the two *Spice* files have been combined into a single file, using a text editor. The file describes the complete diplexer, including the susceptance corrector, and is shown below for reference.

```

<New Passive Filter>
* Lowpass filter
r01      2      0      5.000000E+01
c02      2      0      5.819499E-08
l03      2      3      4.013237E-03
c03      2      3      4.359531E-06
c04      3      0      2.845381E-06
l05      3      4      5.815213E-03
c05      3      4      3.543912E-06
c06      4      0      3.557698E-06
l07      4      5      1.227345E-02
c07      4      5      7.917711E-07
c08      5      0      5.075212E-06
l09      5      6      1.298651E-02
* Termination moved here:
r11      6      1      5.000000E+01
* Source moved here:
vin      1      0      ac      2.0

```

```

* Highpass filter
r12      10    0    5.000000E+01
c12      10   13    3.929569E-05
l13      13   14    1.613563E-02
c13      14    0    1.745816E-06
r13      13    0    1.000000E+08
c14      13   15    3.180428E-06
l15      15   16    1.080307E-02
c15      16    0    2.338898E-06
r15      15    0    1.000000E+08
c16      15   17    2.612817E-06
l17      17   18    5.810530E-03
c17      18    0    6.801610E-06
r17      17    0    1.000000E+08
c18      17   19    1.793760E-06
l19      19   20    4.557877E-03
c19      20    0    2.348286E-05
r19      19    0    1.000000E+08
c20      19    6    1.886385E-06
* Susceptance corrector
l10       6    7    9.500000E-02
c10       7    0    2.425000E-07
.ac lin 101      0.000000E+00      2.000000E+03
* Lowpass output
.print ac vdb(2, 0) vp(2, 0)
* Highpass output
.print ac vdb(10, 0) vp(10, 0)
.end

```

This file is the combination of the two files generated by **Filsyn**, but of course, it had to be edited somewhat to accommodate the combination. The highpass part contains some large resistors, because *Spice* does not like floating capacitors.

We used the *Spice* program for a frequency domain analysis with the following results:

1. The lowpass side has a maximum loss of 0.085 dB at 980 Hz and a minimum stopband loss of 53.2 dB
2. The highpass side has a maximum passband loss of 0.076 dB at 1120 Hz and a minimum stopband loss of 64.3 dB

The performance of the diplexer is shown graphically in Fig. 39 below. The loss peak in the transition band is due to the susceptance corrector of course.

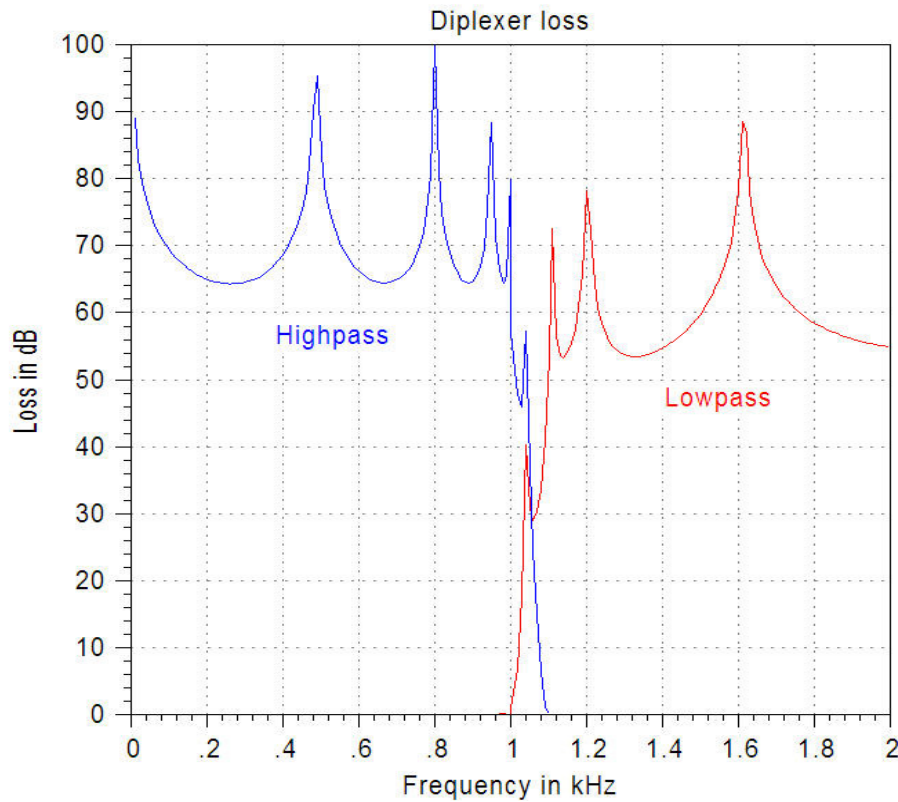


Fig. 39 Diplexer behavior

The *Touchstone* or any other general purpose analysis program can equally well be used for this purpose, with identical results.

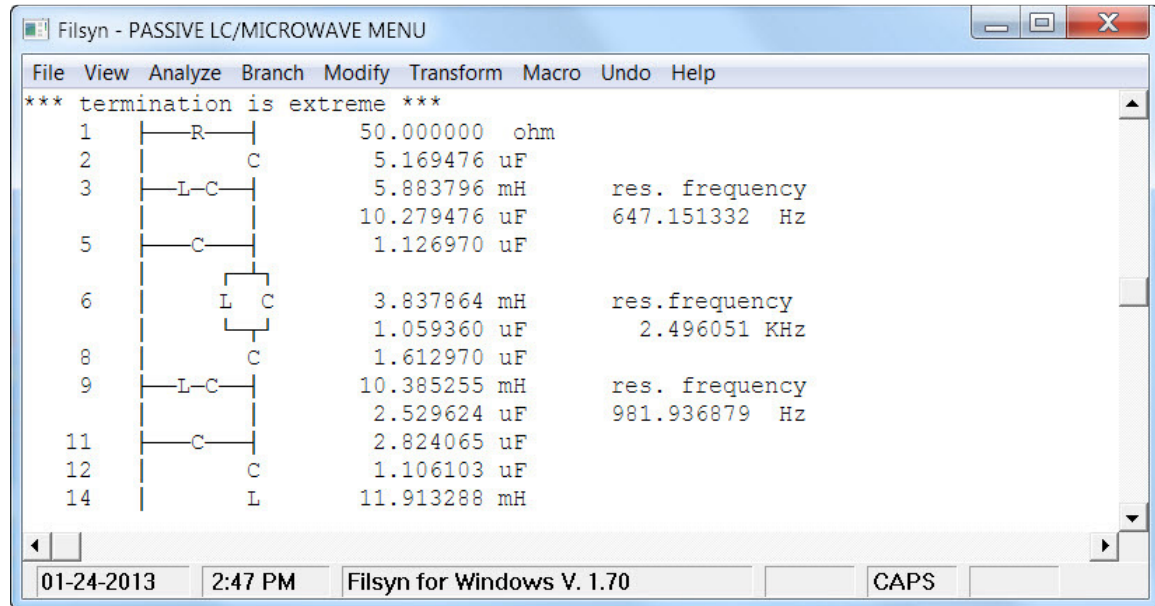
6. Multiplexers

If more than two bands are to be separated or combined, at least one of the filters will necessarily be a bandpass. Such a bandpass can be designed as a short-circuited filter in a manner similar to the lowpass and highpass designs presented above. One problem, specific to this bandpass, is the impedance level seen at the shorted end of the filter.

It was pointed out several times in this manual, that bandpass filter structures have an inherent impedance transforming property, and the output termination is usually not equal to the specified value. In a doubly-terminated case where both terminations are finite, the actual value is clearly indicated and may be modified at will using the **Modify->Norton->Simple** (Norton transformation) menu option. However, in the extreme terminated case, the impedance looking *into* the filter is not apparent, but is of great importance and must be adjusted to the correct value. The menu item is still available to perform the transformation but the value *before* the transformation must first be established by an analysis of the bandpass.

The method is illustrated by a simple bandpass of parametric type. To get a short-circuited output for paralleling, we must specify an even number of zeros (two in this case) at infinite frequency, and, for parametric filters, an odd number of zeros (one in this

case) at zero frequency. Assuming a passband from 1100 Hz to 2000 Hz and 50 ohms termination, a 35 dB minimum loss below 1 KHz and above 2.4 KHz, the program yields the following circuit as an example:



At this point we have no idea what the impedance, or rather the admittance looking into the filter is at the shorted output. We must perform an analysis in the passband first and that indicates that the maximum input admittance is about 22.55 mS. We need a value of 20 mS (50 ohms), hence we need a transformation value of

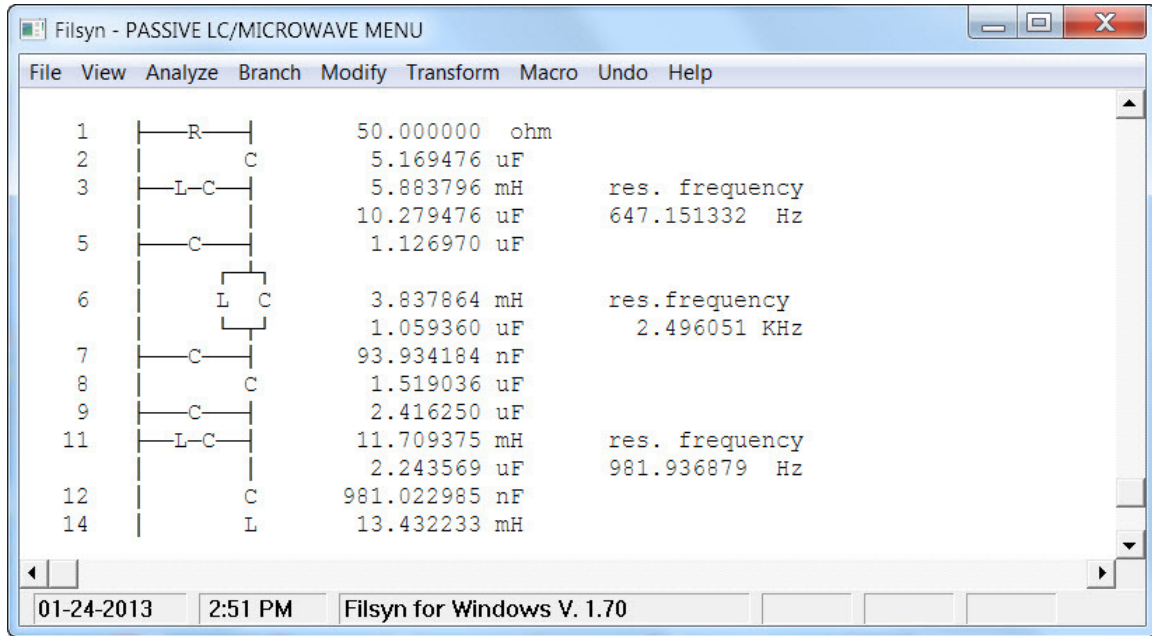
$$t = 22.55/20.0 = 1.1275$$

which we can specify through the **Modify->Norton->Simple** menu option as shown below:

The screenshot shows the '<New Passive Filter>' dialog box. It contains the following fields and buttons:

- Enter 2 branch numbers:** Two input fields, both containing '0'.
- Enter new termination:** An input field containing '0.00000000'.
- or transformation value:** An input field containing '1.1275'.
- Buttons:** 'OK' and 'Cancel' buttons at the bottom.

and this yields the final circuit as:



A subsequent analysis, not shown, confirms that the passband behavior of this bandpass is perfect and its input admittance is of equal-ripple type with 20 mS maximum, as expected.

If this filter is now paralleled with the lowpass of the previous example and a highpass that has a passband above 2.4 KHz, all of 50 ohms impedance, that would provide a 3-way multiplexer. The design of an optional susceptance corrector would need the analysis of the complete circuit and will contain *two* resonators, one resonating in the transition region between the lowpass and the bandpass, the other in the region between the bandpass and the highpass filters. Additional bandpasses may be added for, say, a 4-way multiplexer in a similar manner.

7. Summary

In summary, diplexers and multiplexers can be designed by the use of the short-circuited option of **Filsyn**. A certain amount of additional calculations must be performed outside of the program to calculate the susceptance sum and the appropriate corrector, but these represent a minor effort. By submitting the results to *Spice*, or any other analysis and optimization program, we may use that optimization capability to eliminate the need for this effort.

Essentially the same technique may be used for microwave diplexer and multiplexer designs (see ref.[30]), but we may encounter cases, where a touch-up optimization *will* become necessary. In this case the line length is also available as a variable.

Predistortion must *not* be used in multiplexer design, because this will interfere with the constant conductance property of these short-circuited filters.

APPLICATION NOTE 2

USE OF COMPLEX TRANSMISSION ZEROS

1. Introduction

Filter transmission zeros are usually located on the imaginary s (real frequency) axis. However, complex transmission zeros may also be used, although their implementation in passive LC filters requires bridged-T or twin-T circuits, or occasionally coupled inductors. Nevertheless, they may be sometimes useful to equalize filter delay and also add to the discrimination properties of the filter; thus they are more effective than pure delay equalizers.

In this application note we show, that the delay equalizer command can be used to obtain excellent locations for the complex transmission zeros. We illustrate this by a lowpass example, but the method is applicable to bandpass filters also.

2. Original lowpass

A complex transmission zero adds to the stopband loss an amount that increases with the distance from the passband, but the actual value is difficult to estimate. Therefore, we will start with a lowpass that has a stopband loss that decreases somewhat as we move away from the passband. To do this, we will use the **Placer** segment and we proceed as shown (the requirements are arbitrary, but represent a filter of reasonable complexity). The passband is to 1 KHz with 0.1 dB loss ripple, one zero at infinity and the stopband requirements are:

Filter Parameters

Requirements

	Frequency (Hz)	Loss (dB)
1	1.050000E+03	35.0000000
2	1.500000E+03	25.0000000
3	0.00000	0.00000000
4	0.00000	0.00000000
5	0.00000	0.00000000
6	0.00000	0.00000000
7	0.00000	0.00000000
8	0.00000	0.00000000
9	0.00000	0.00000000
10	0.00000	0.00000000
11	0.00000	0.00000000

Fixed zeros

	Frequency (Hz)	Multiplicity
1	0.00000	0
2	0.00000	0
3	0.00000	0
4	0.00000	0
5	0.00000	0
6	0.00000	0
7	0.00000	0
8	0.00000	0
9	0.00000	0
10	0.00000	0

Movable zeros

	Frequency (Hz)	Multiplicity
1	0.00000	0
2	0.00000	0
3	0.00000	0
4	0.00000	0
5	0.00000	0
6	0.00000	0
7	0.00000	0
8	0.00000	0
9	0.00000	0
10	0.00000	0
11	0.00000	0

or no. of zeros below passband

0

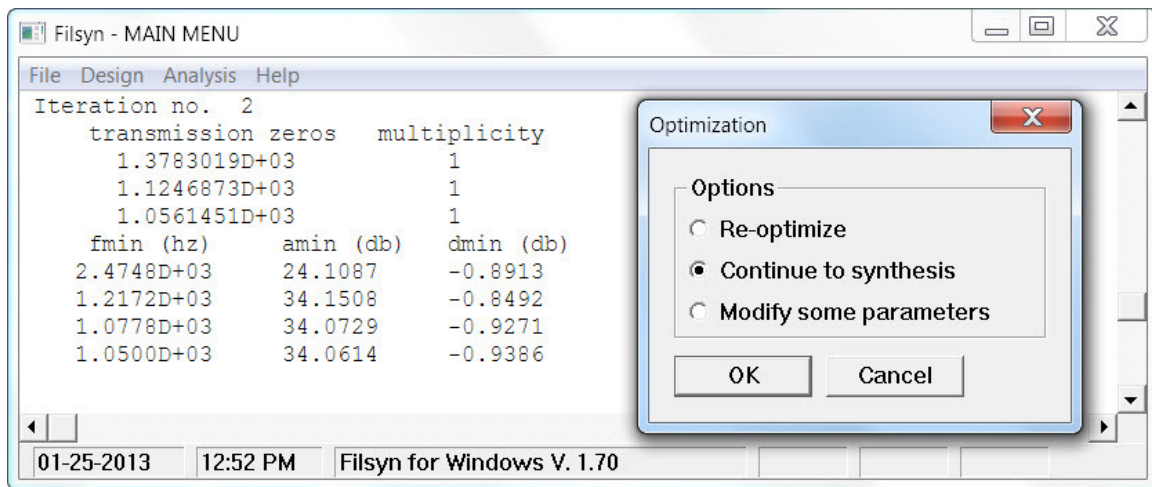
and/or no. of zeros above passband

3

OK

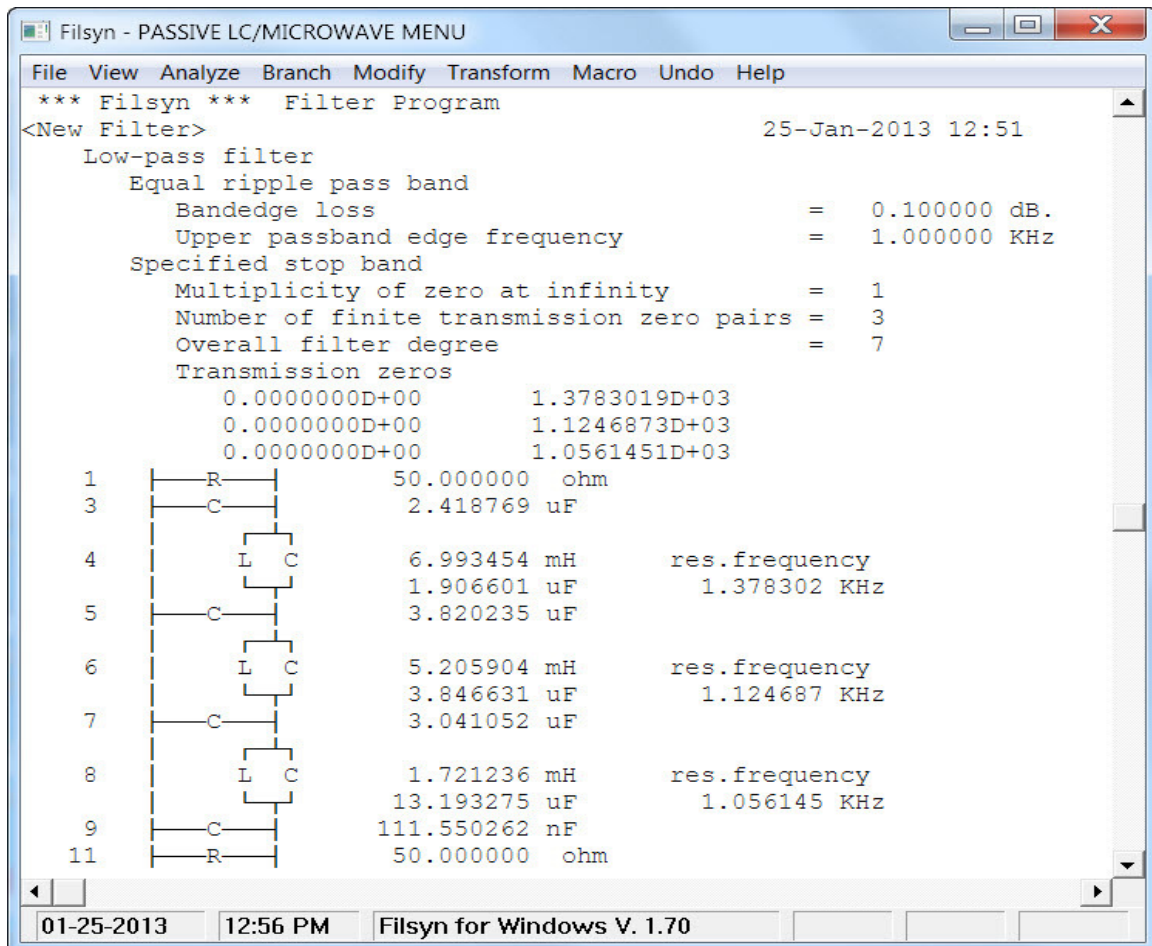
Cancel

The iterative procedure stopped here:

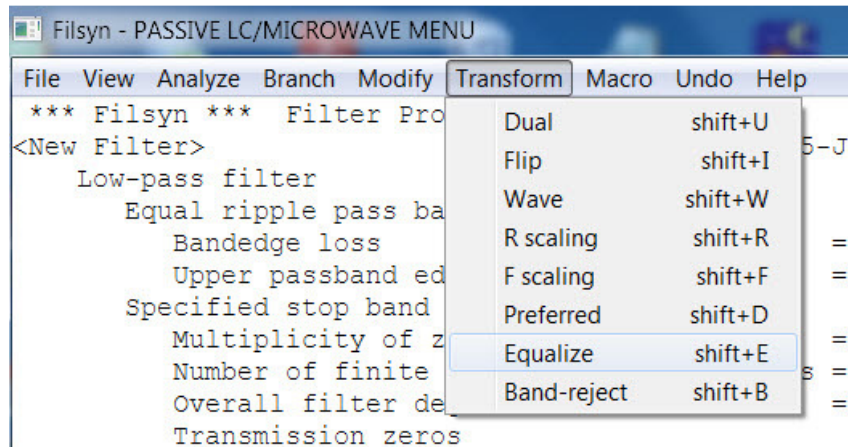


The requirements are not quite met, but the complex quadruplets of zeros we add later, will improve the stopband considerably.

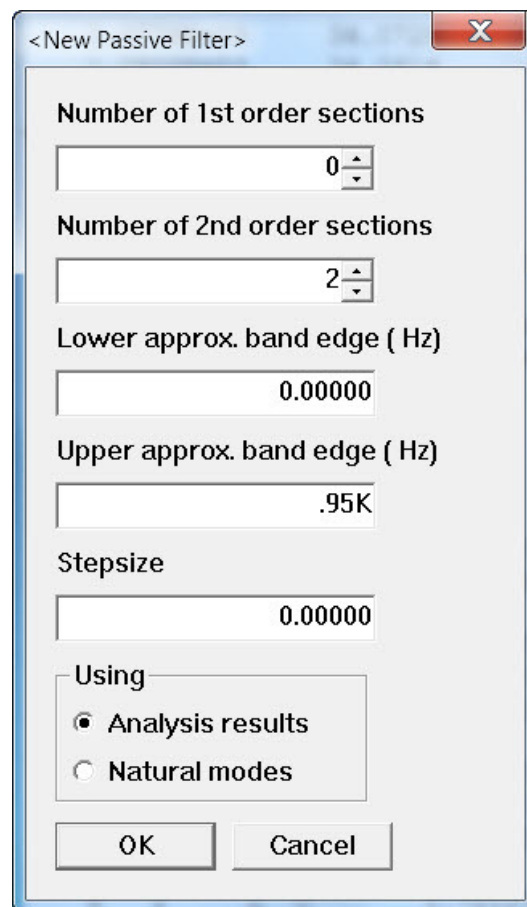
Proceeding with the synthesis, we reach the configuration as usual:



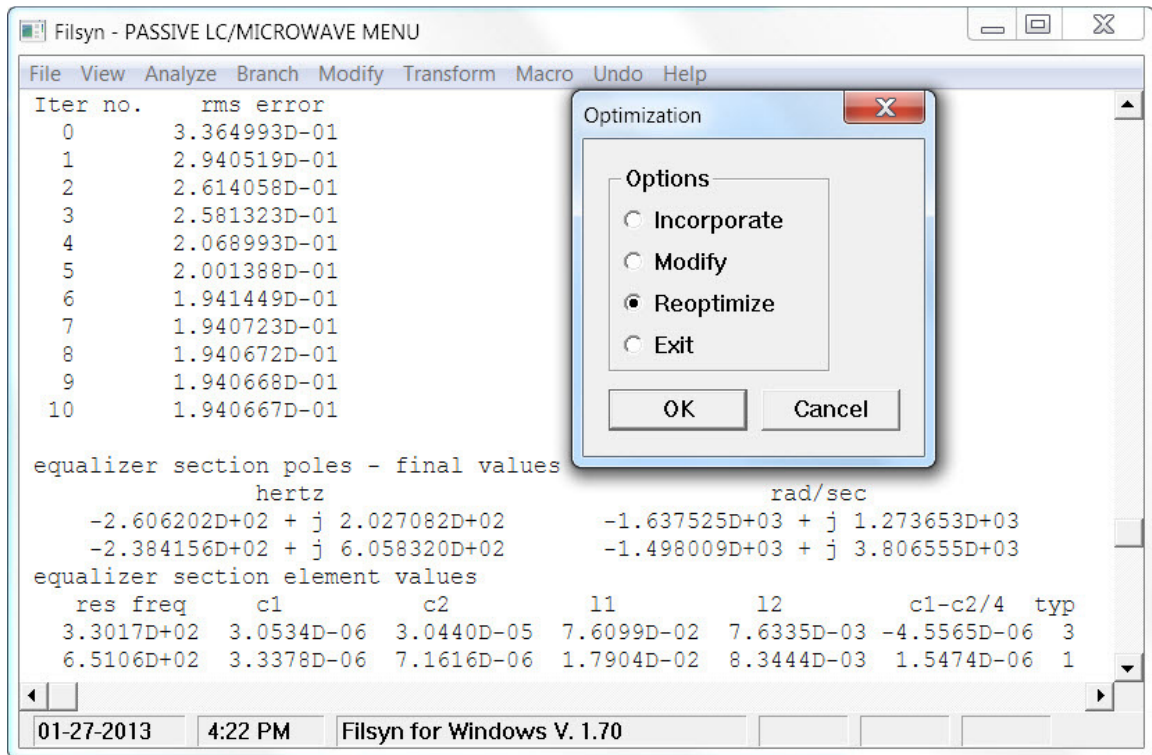
Next we wish to design a delay equalizer, therefore we first perform an analysis, followed by the delay equalization step:



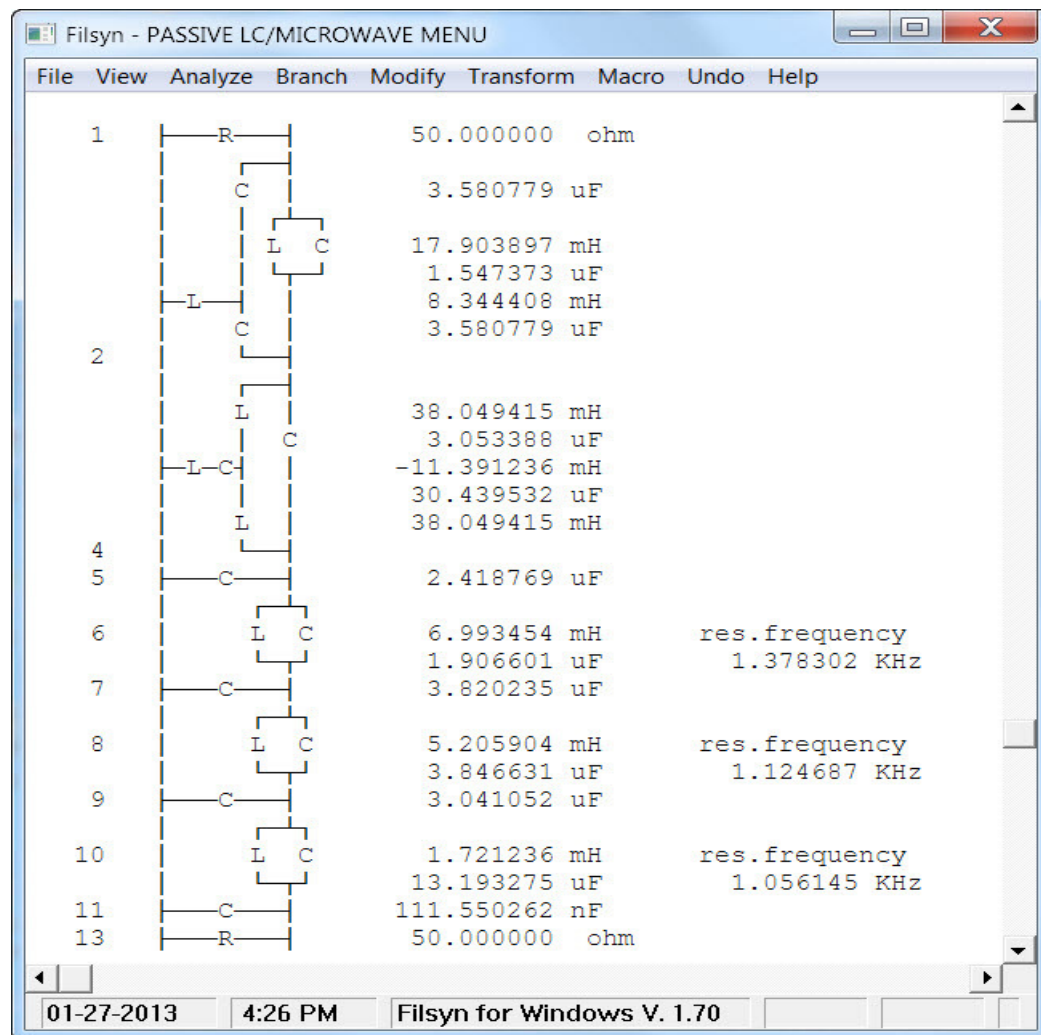
Which brings up the entry screen:



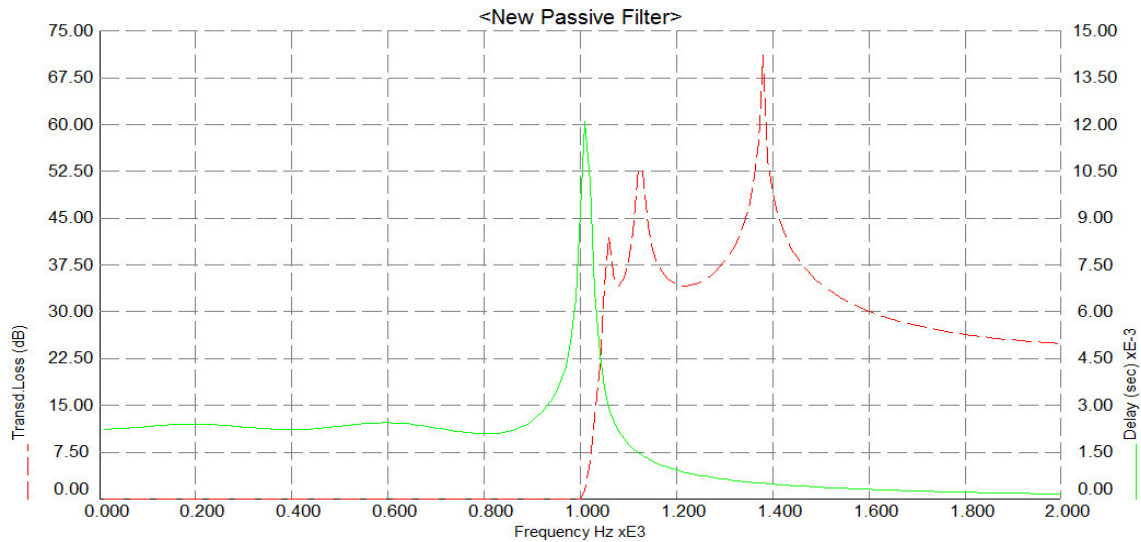
At this point we show the final equalizer data that we shall need later on:



For comparison purposes the complete filter-equalizer combination is here:



Next we performed a detailed analysis. The stopband loss and the passband delay are the characteristics of interest and they are displayed in graphical form below. The loss is as expected, the delay is reasonable and is close to equal ripple in 90% of the passband.

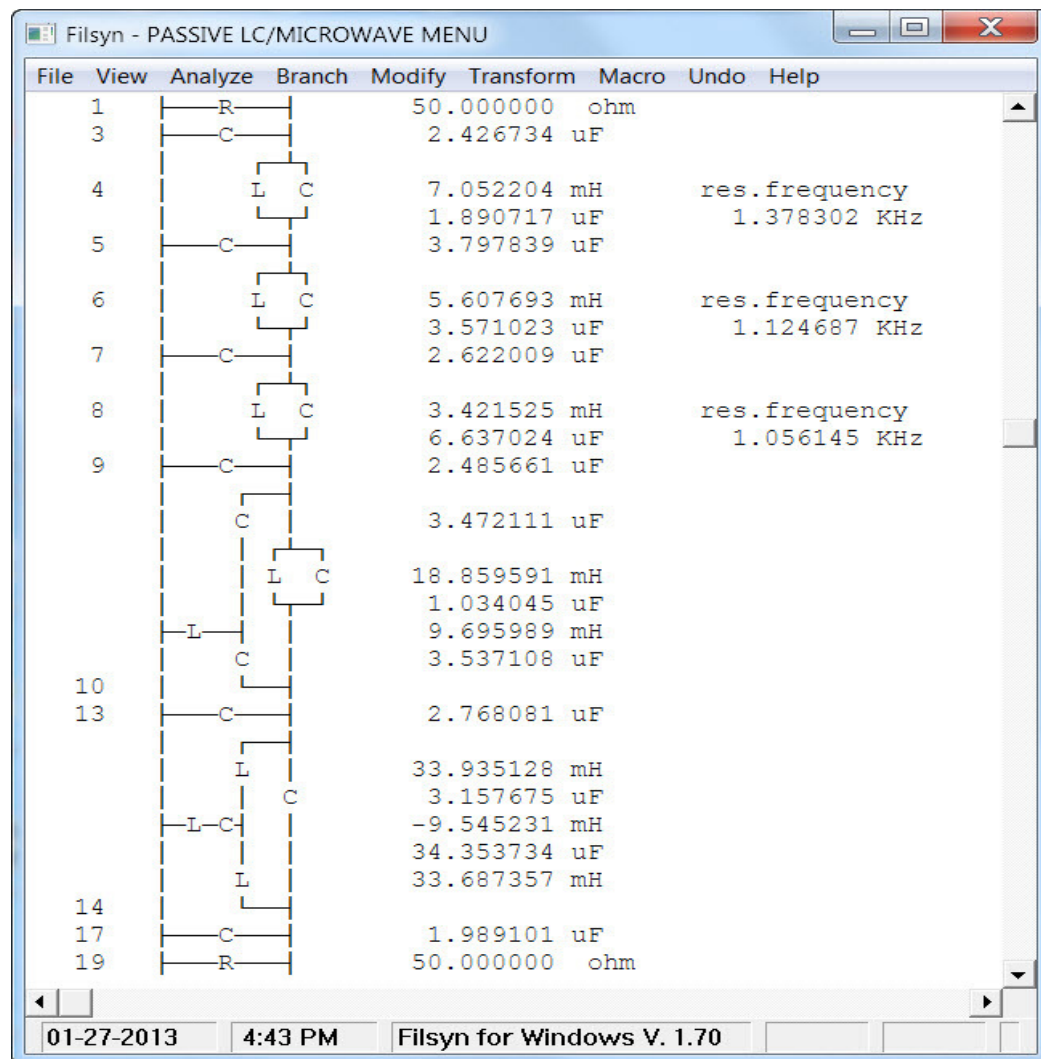


3. Combined Design

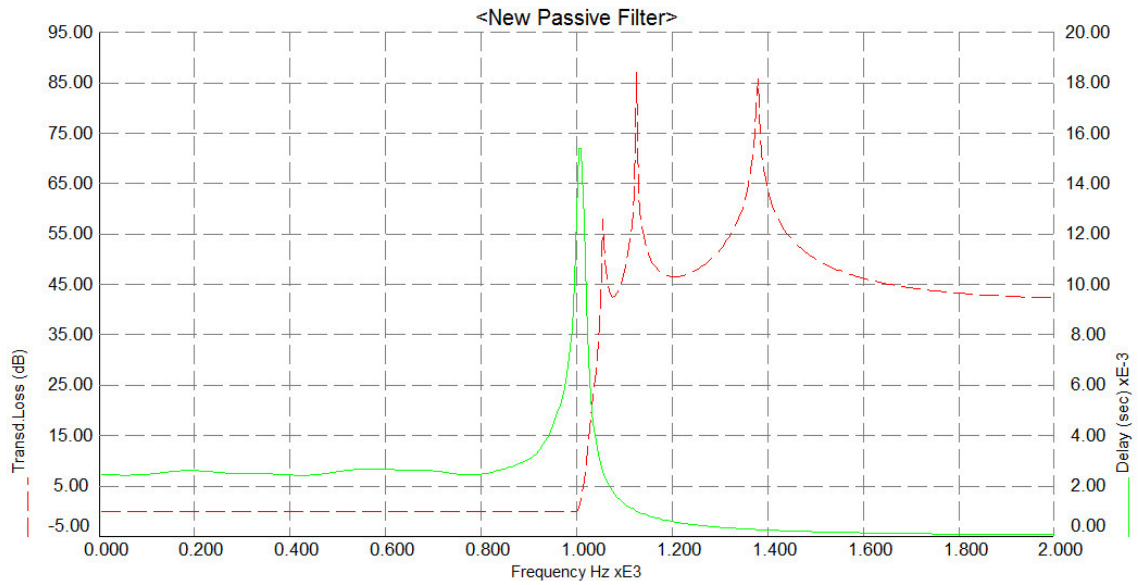
Next we redesign this filter using the **Specified** option, and combine the zeros of the original lowpass with the complex quadruplets obtained from the delay equalization step, as follows: the passband is from 0 to 1 KHz with 0.1 dB passband ripple, there is a single zero at infinity and the **Detail parameters** sub window contains all the transmission zeros of the previous design:

Transmission zeros (Hz)		
	Real Part	Imaginary Part
1	0.00000	1.056145E+03
2	0.00000	1.124687E+03
3	0.00000	1.378302E+03
4	260.620200E+00	202.708200E+00
5	238.415600E+00	605.832000E+00
6	0.00000	0.00000
7	0.00000	0.00000
8	0.00000	0.00000
9	0.00000	0.00000
10	0.00000	0.00000
11	0.00000	0.00000

The filter transmission zeros contain both the ones on the imaginary axis from the original lowpass as well as the two complex pairs from the delay equalizer. The resulting filter is now:



The last step is to perform an analysis to check the filter performance and show it below for comparison.



The stopband is considerably better, although not quite equal minima. The lowest minimum is about 42 dB, much better than the exact degree 7 elliptic design (30.4 dB). The surprise is the passband delay, which is very close to that of the original equalized lowpass.

The circuit configuration is similar. The second design contains 2 additional capacitors, which is a small price to pay for the substantially better stopband performance.

APPLICATION NOTE 3

LOWER SIDEBAND CRYSTAL FILTER

1. Introduction

Very narrow band crystal filters can be implemented in ladder form, and **Filsyn** can of course, do this easily enough. The resulting ladders must, however, be modified so that the proper three-element branches are formed and they have the proper minimum capacitance ratios. This has been possible as long as the crystals were in the shunt branch, i.e. represented transmission zeros *below* the passband. Using the other three-element branches, we can handle the cases, where the crystals are in the series branches, i.e. implement transmission zeros *above* the passband. Both the lower and upper sideband cases can be handled by the script files embedded in the **Filscript.exe** program. This application note will demonstrate the use of these branches in the lower sideband case and the corresponding **Modify->Convert** menu option, which converts them from one form to the other.

2. Lower sideband filter

Again the method is illustrated by an example. This is a single-sideband bandpass filter with passband from 2 MHz to 2.003 MHz with 0.1 dB ripple and we need to suppress everything from 600 Hz above the passband by at least 60 dB. Terminations tentatively selected to be 1 kohms and the initial **Filsyn** session is as follows. The **Detail parameters** window contains 2 breakpoints, one at 1.99 MHz of 50 dB (not significant, but required, since there must be at least one breakpoint in each stopband) and one at 2.0036 MHz of 60 dB. We also requested three movable transmission zeros in the upper stop band:

The screenshot shows the 'Filter Parameters' dialog box with three main sections: Requirements, Fixed zeros, and Movable zeros. Each section contains a table with columns for Frequency (Hz) and Loss (dB) or Multiplicity.

Requirements		Fixed zeros		Movable zeros	
	Frequency (Hz)	Loss (dB)		Frequency (Hz)	Multiplicity
1	1.990000E+06	50.00000000	1	0.000000	0
2	2.003600E+06	60.00000000	2	0.000000	0
3	0.000000	0.00000000	3	0.000000	0
4	0.000000	0.00000000	4	0.000000	0
5	0.000000	0.00000000	5	0.000000	0
6	0.000000	0.00000000	6	0.000000	0
7	0.000000	0.00000000	7	0.000000	0
8	0.000000	0.00000000	8	0.000000	0
9	0.000000	0.00000000	9	0.000000	0
10	0.000000	0.00000000	10	0.000000	0
11	0.000000	0.00000000			

Below the tables, there are two input fields:

or no. of zeros below passband:

and/or no. of zeros above passband:

At the bottom are 'OK' and 'Cancel' buttons.

<New Filter>

Filter kind

- ☒ LC
- ☐ Microwave
- ☐ Active RC
- ☐ IIR Digital

Quarter-wave/Sampling freq (Hz)
0.00000

Filter type

- ☐ Low pass
- ☐ High pass
- ☒ Band pass

Lower passband freq (Hz)
2.000000E+06

Upper passband freq (Hz)
2.003000E+06

Passband type

- ☐ Max. flat
- ☒ Equal ripple
- ☐ Functional
- ☐ Sloping

Band-edge loss/return loss (dB)
0.100000000

Loss Slope (dB/oct)
6.00000000

Flat loss (dB)
0.00000000

Function Type

- ☒ E
- ☐ F

Multiplier
0.00000000

Stopband

- ☐ Monotonic
- ☐ Equal min
- ☒ Placer
- ☐ Specified

Lower stopband freq (Hz)
0.00000

Loss (dB)
50.0000000

Upper stopband freq (Hz)
0.00000

Loss (dB)
50.0000000

Detail Parameters

of zeros at zero
3

of zeros at infinity
1

of unit elements
0

Parametric

- ☒ Conventional
- ☐ Parametric
- ☐ Matching

R1
1.000000E+03

R2
1.000000E+03

ZS parameter
-1

Q for predistortion
0.00000000

Predistortion

- ☒ Passband only
- ☐ Pass/stop bands

Shifted bandpass trans.
Center frequency (Hz)
0.00000

☐ Odd parametric

☐ Save design data

OK Cancel

These parameters, in particular the multiplicity of transmission zero at zero, were selected such, that the resulting structure will have shunt inductors at both ends, which will be useful for the practical implementation of the filter.

The pole-placer runs uneventfully and the last results are (after further optimization):

Filsyn - MAIN MENU

File Design Analysis Help

Iteration no. 1

transmission zeros	multiplicity
2.0057117D+06	1
2.0040441D+06	1
2.0036396D+06	1

fmin (hz)	amin (db)	dmin (db)
1.9900D+06	52.2869	2.2869
2.0088D+06	61.3627	1.3627
2.0046D+06	61.3627	1.3627
2.0038D+06	61.3626	1.3626
2.0036D+06	61.3627	1.3627

01-27-2013 5:12 PM Filsyn for Windows V. 1.70

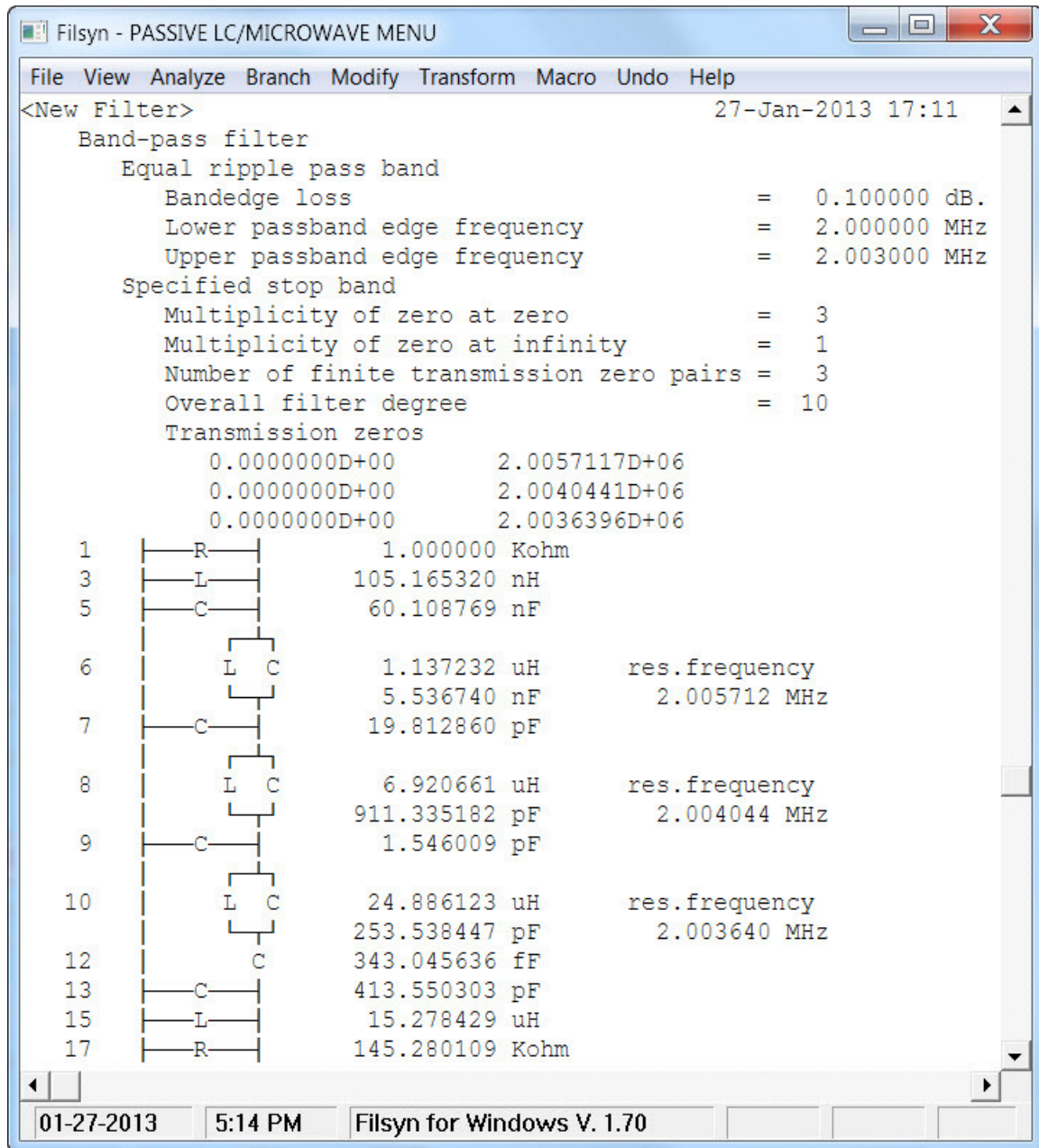
Optimization

Options

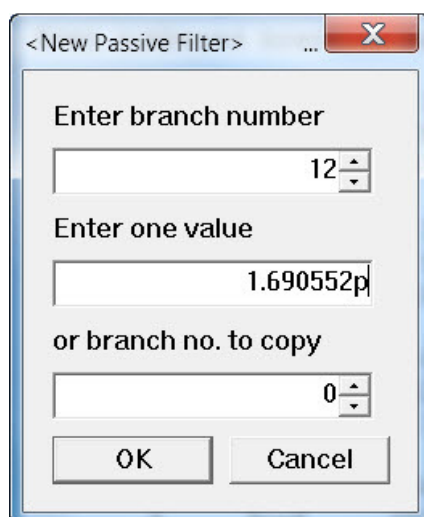
- ☐ Re-optimize
- ☒ Continue to synthesis
- ☐ Modify some parameters

OK Cancel

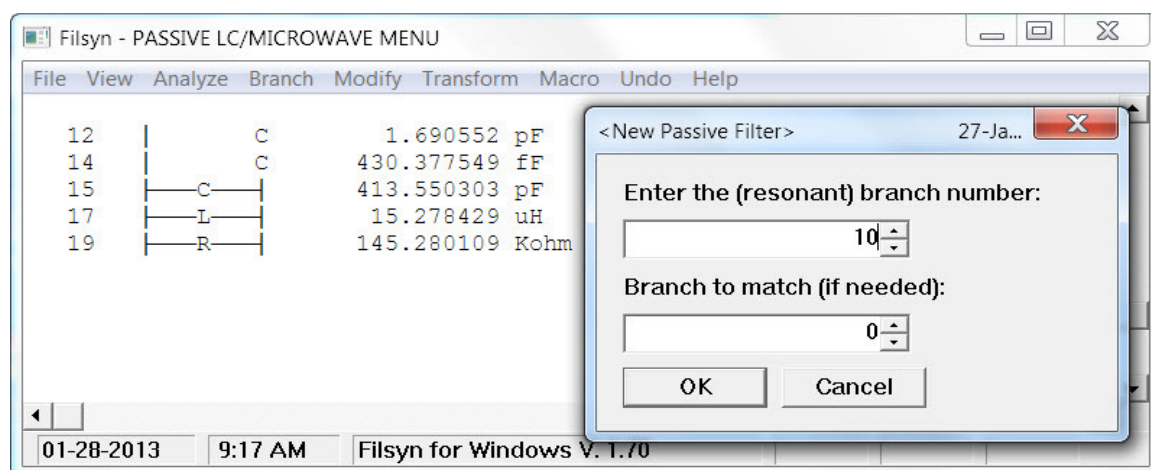
Next we perform the ladder synthesis, again uneventfully, using the computer-generated structure:



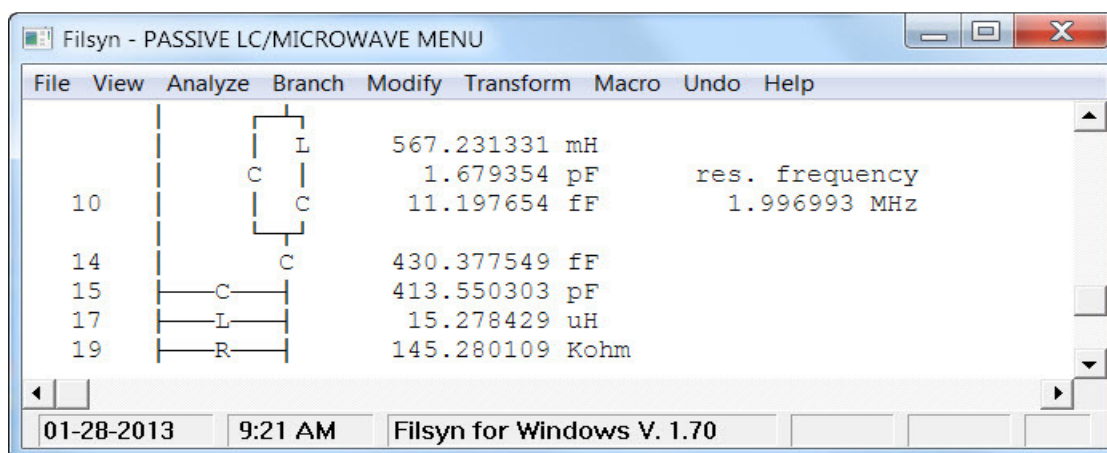
To convert this into a crystal ladder, the first step is to split C_{12} , such that one part of it, together with branch 10, forms a crystal with a specified capacitance ratio. We selected this ratio arbitrarily to be 150, hence one part of C_{12} must be the value of C_{10} divided by 150, i.e. 1.690552 pF. We use the **Branch->Split** menu item and declining the offer to split a value in half, we get:



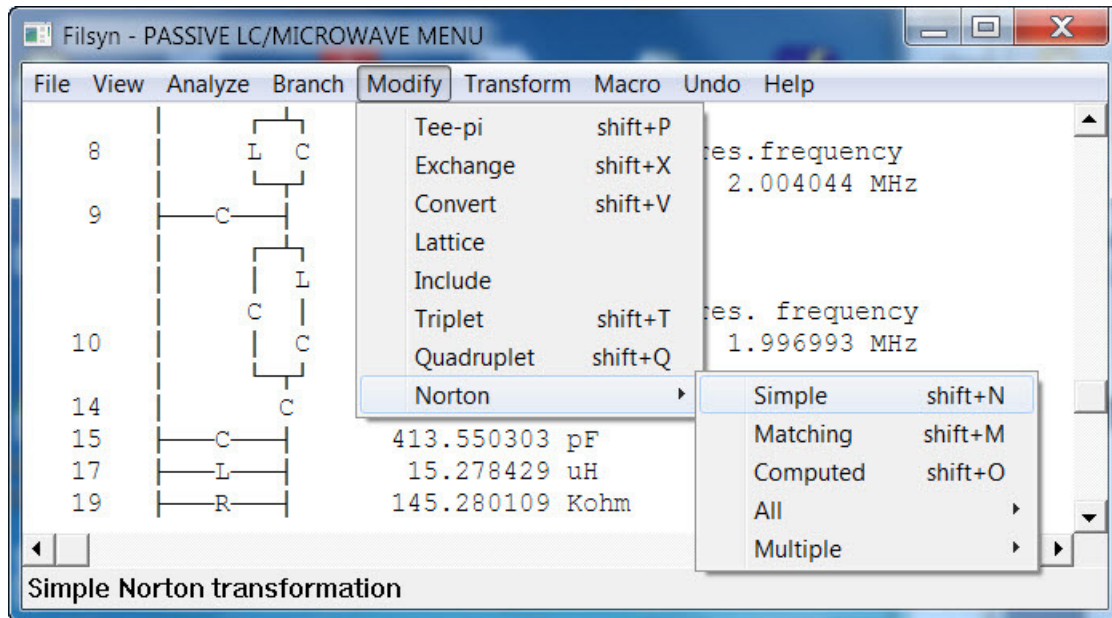
Next we use the **Modify->Convert** menu item to convert branches 10 and 12 into the other form, more appropriate for crystals. Note that the two branches forming the new three-element branch must be next to each other and the second branch need not be specified, the program will automatically pick up C_{12} .



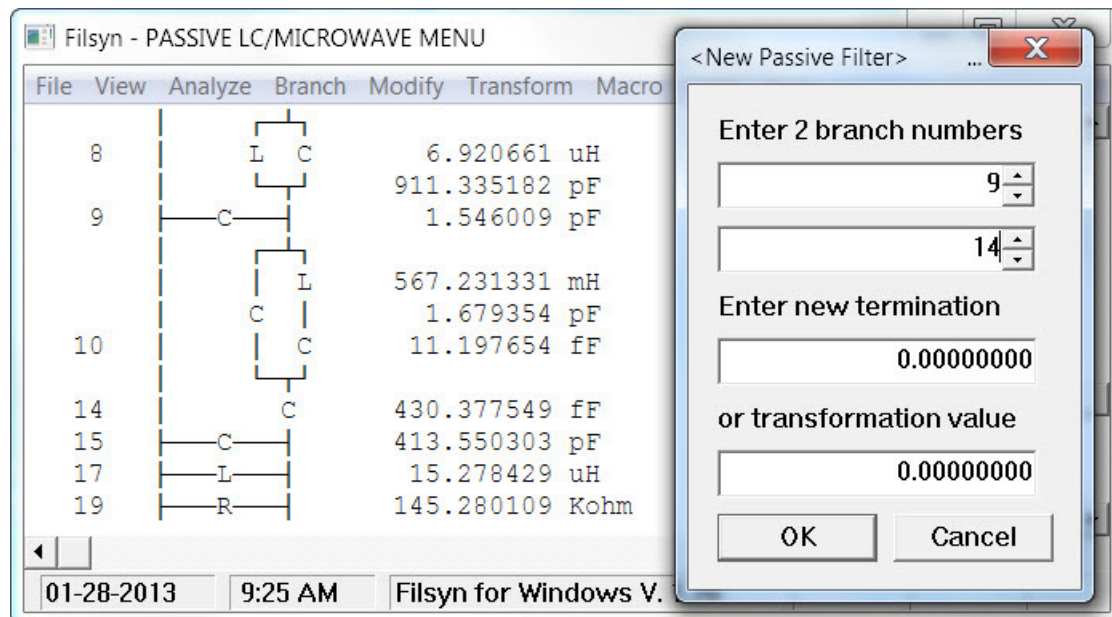
As expected, this is the right form and the capacitance ratio is 150.



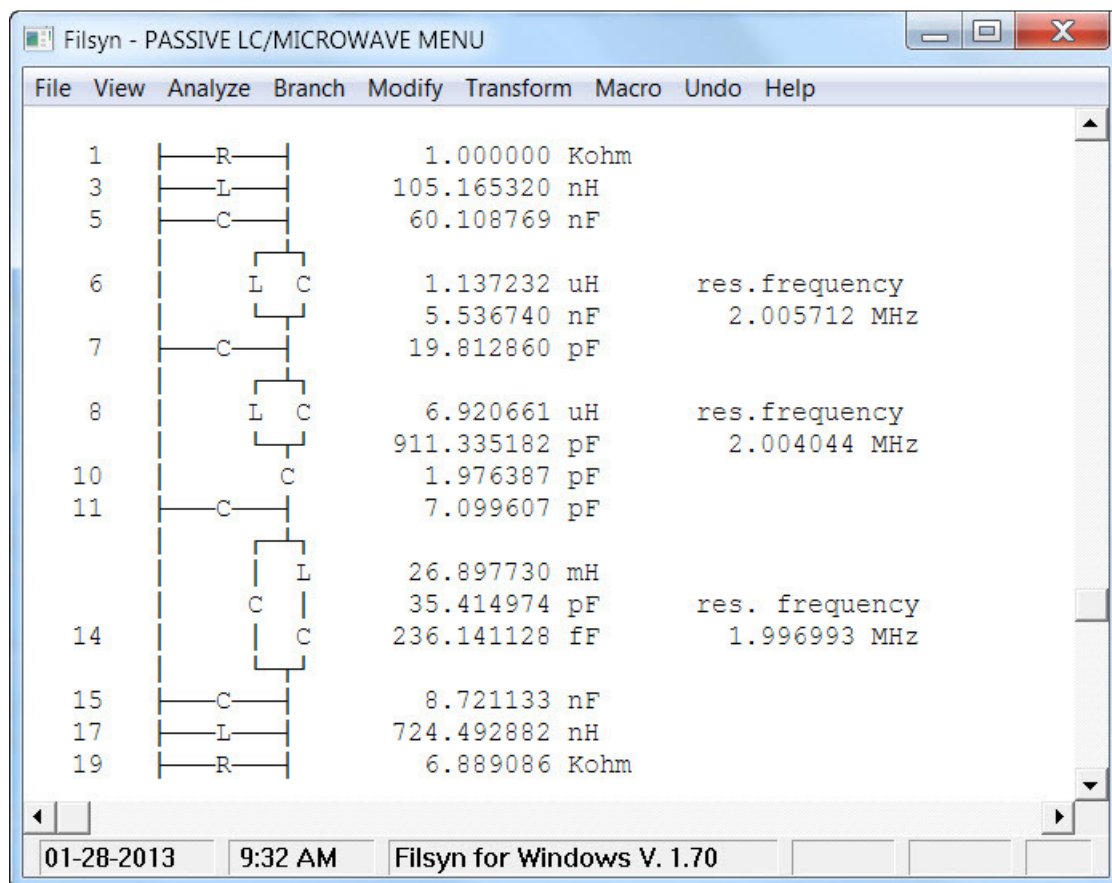
The next step is to shift the remaining series capacitor C_{14} next to branch 8. This needs **Modify->Norton->Simple** menu option to flip the capacitor L-section of C_9 and C_{14} around.



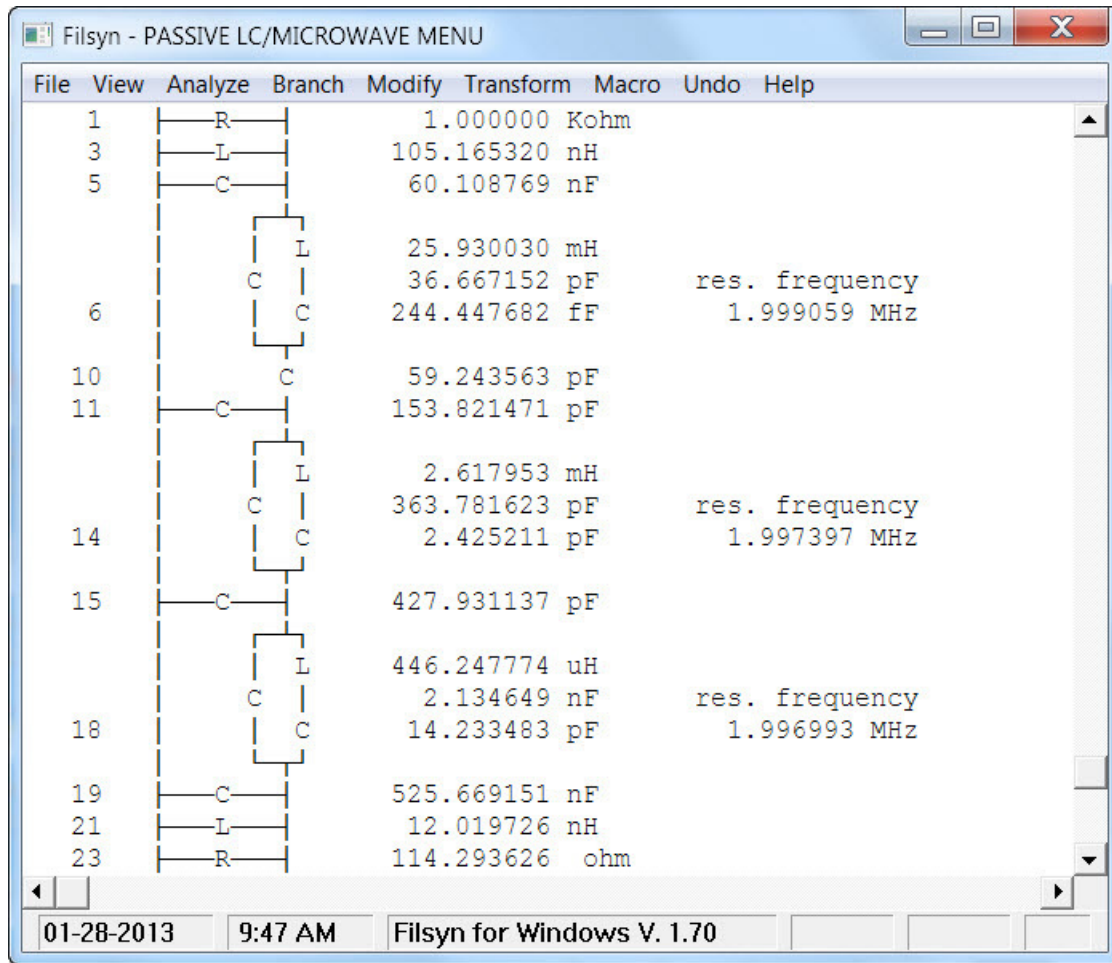
We only need to enter the two branch numbers and accept the maximum available transformation; again branches 10 and 14 need not be interchanged, the program does it when necessary.



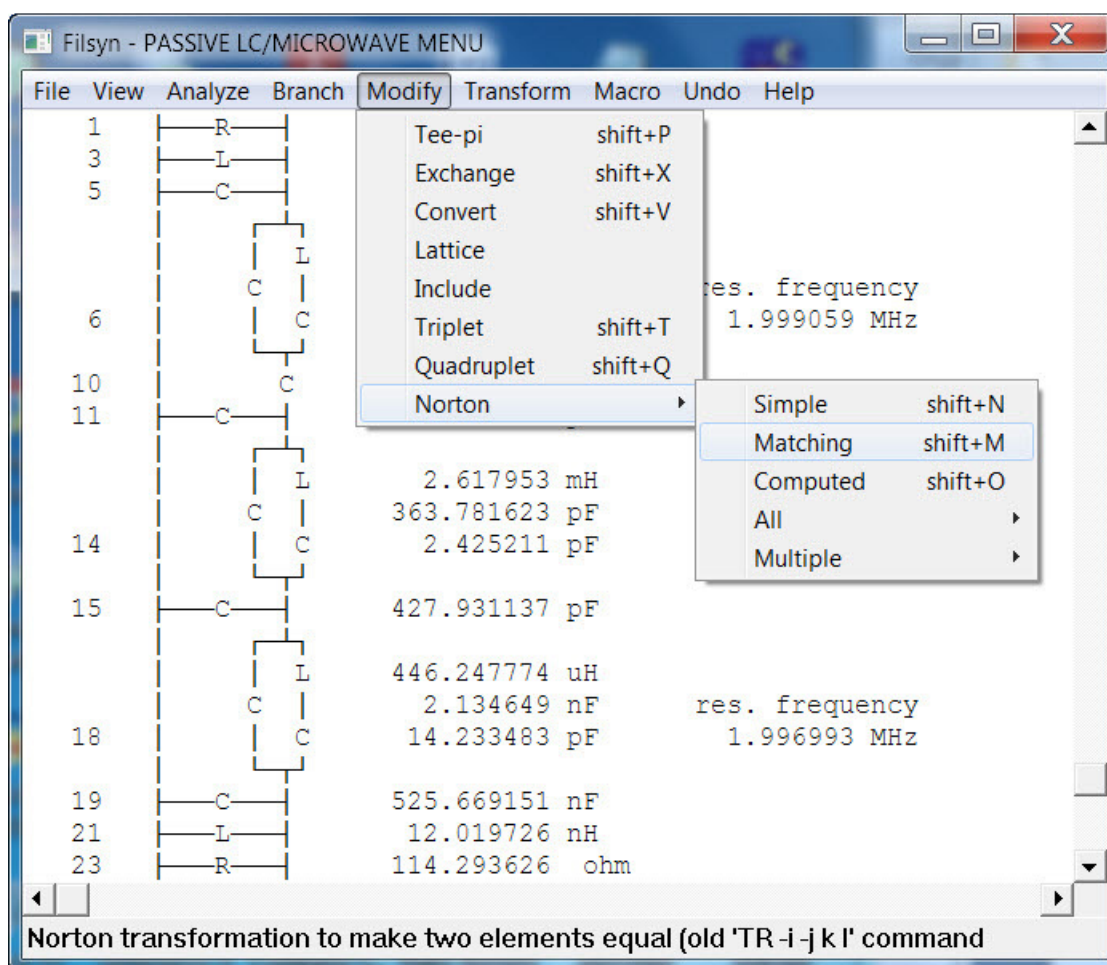
Next we print the circuit to see the complete structure.



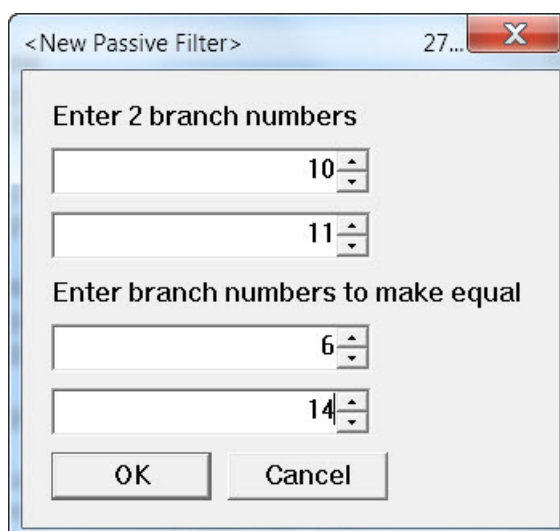
Exactly the same set of steps can be repeated twice more, yielding the following circuit:



We are close to the final circuit, all three crystal branches are in the proper form and they all have a capacitance ratio of 150. For the final touch-up, we will use the remaining series capacitance C_{10} , to make all crystal inductances equal. In order to do this, we can use the **Modify->Norton->Matching** menu item:

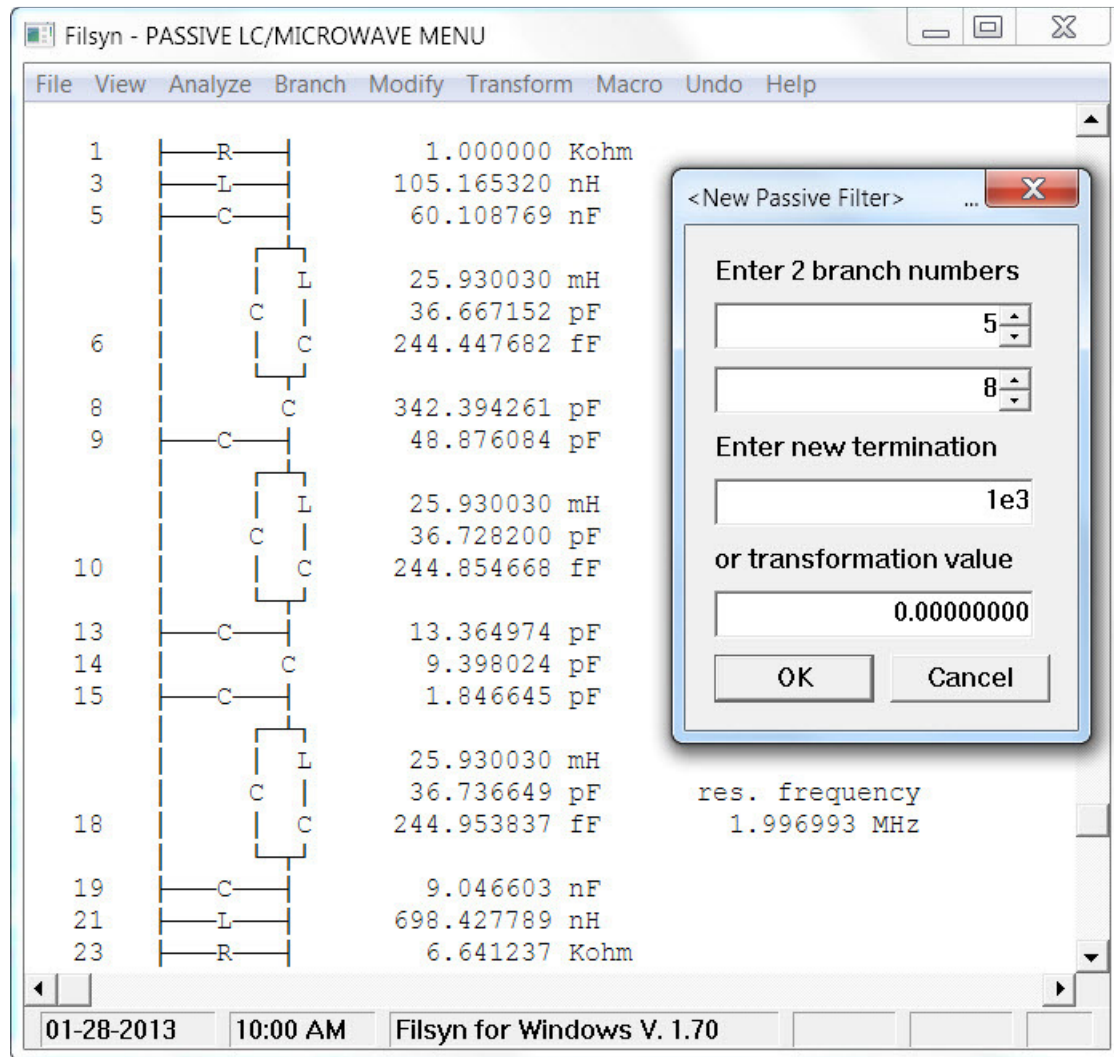


and enter the relevant branch numbers:

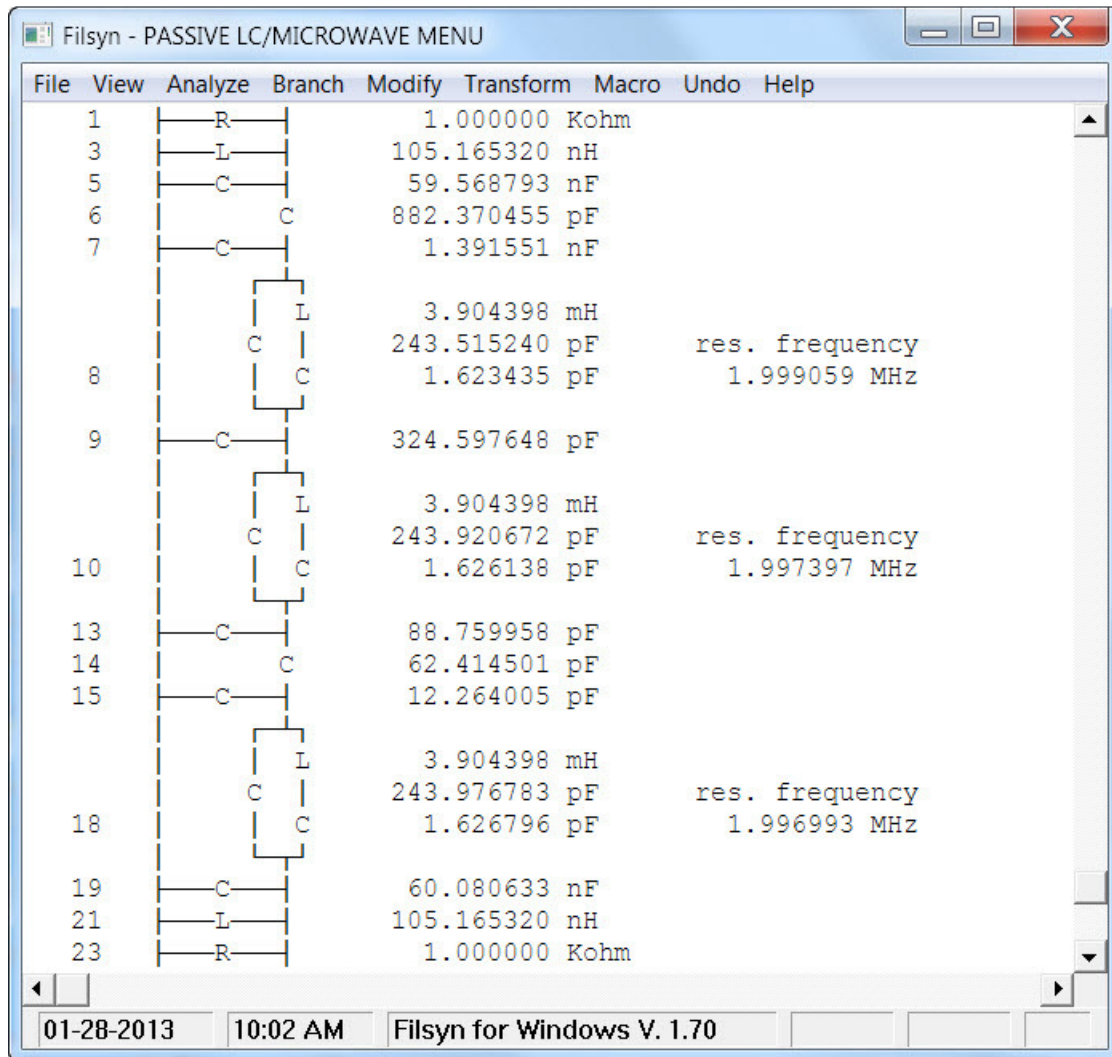


Next in the resulting circuit we convert the capacitive pi to a tee; followed by a repetition of the **Modify->Norton->Matching** step to make the last crystal branch inductance equal to

the others and finally, followed by one more **Modify->Norton->Simple** menu selection to make the two terminations equal:



This yields:

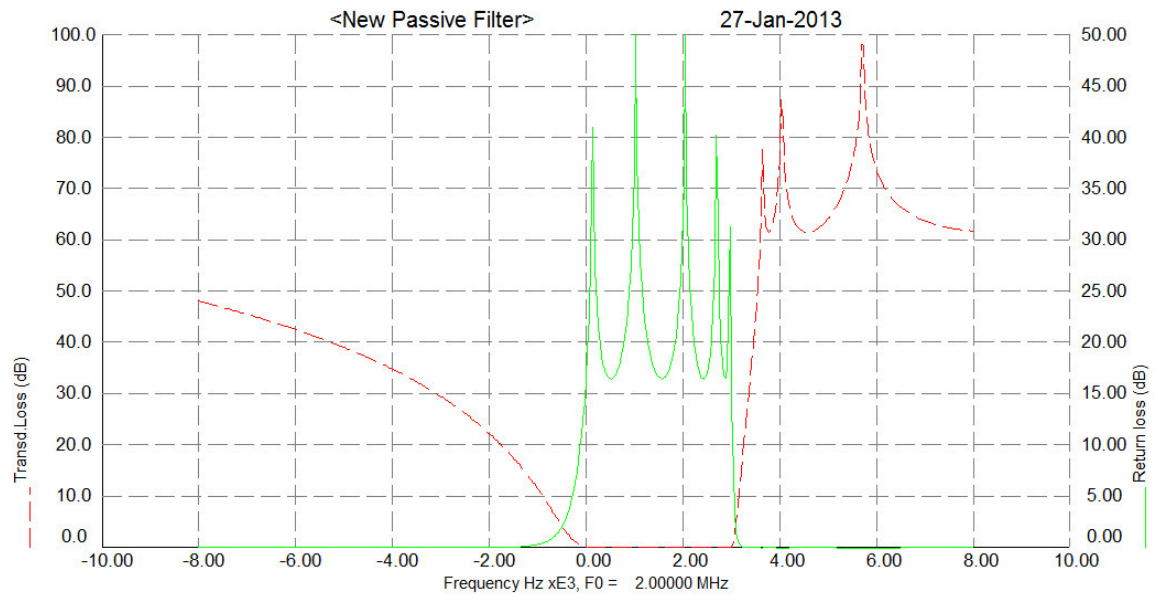


This circuit is now in the right form, contains three crystals in the series branches, with all crystal inductances equal and their resonant frequencies very close to each other.

The resonant frequencies of the parallel resonators at the end may also be computed if required, using the **Analyze->Resonance** menu item. These show that L_3C_5 resonate at 2.010823 MHz while $L_{21}C_{19}$ resonate at 2.002242 MHz. These end inductors may now be used to adjust the inside/outside impedance levels to any value, as well as to adapt to any crystal impedance we may be able to obtain.

The computed performance of this circuit is shown below:

Lower sideband crystal filter



APPLICATION NOTE 4

NARROW-BAND BAND-REJECT FILTERS

1. Introduction

Very narrow-band band-reject filters have two realizability problems: One is that the component Q-values need to be very high, the other is that the spread of component values, in particular those of the inductors, is very wide, proportional to the squared inverse of the bandwidth.

For very high frequencies (in the high MHz region) the attainable Q values are much higher, due to the use of superconducting materials. Here, our main problem is the very wide spread of the inductance values. We will describe a method here that solves this problem, assuming that the passbands on either side of the reject band are also relatively narrow.

2. Starting ladder design

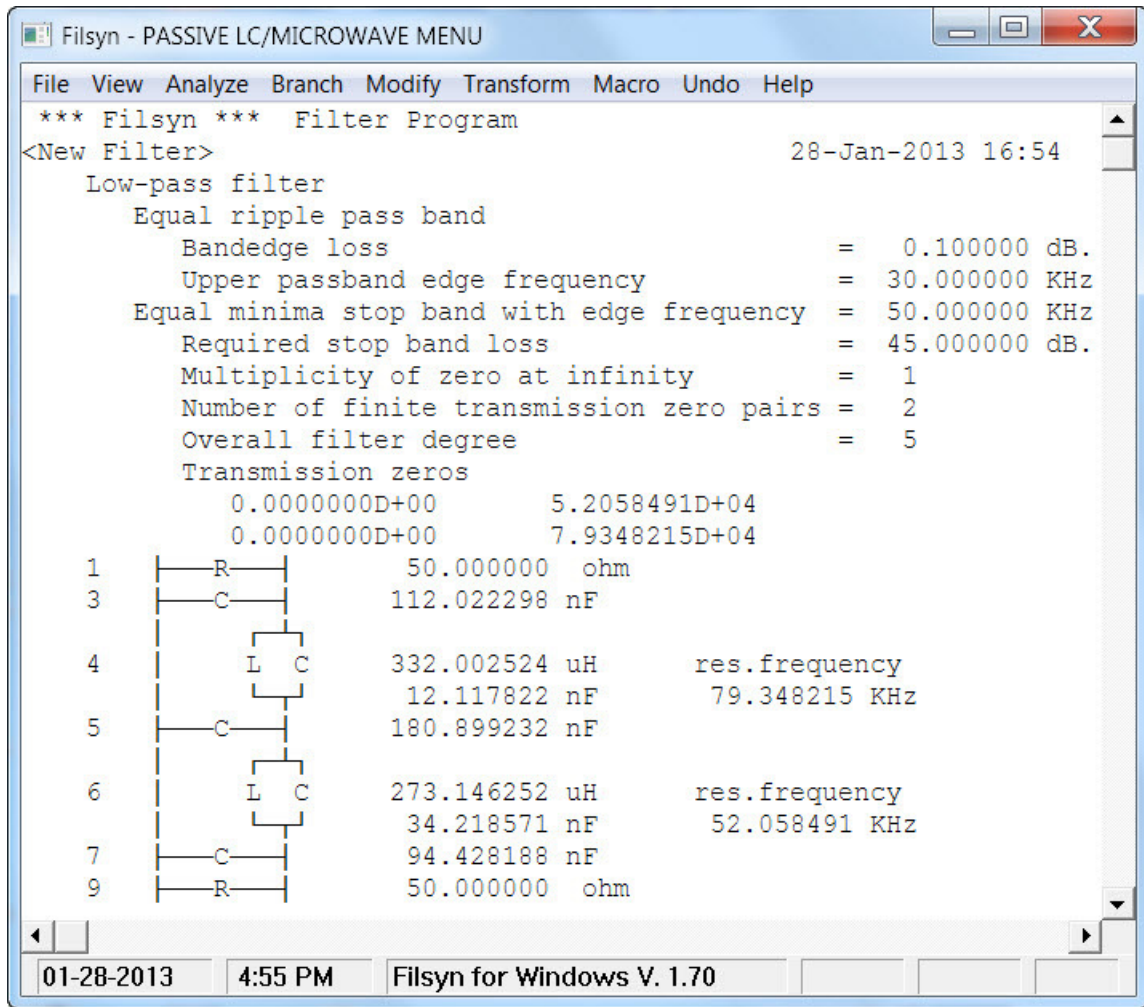
We shall start with the usual design technique of a ladder band-reject filter. It may be any one of the standard Butterworth, Chebyshev or even elliptic design. For illustrative purposes, we shall start with an elliptic filter as the most efficient.

Consider the following set of specifications:

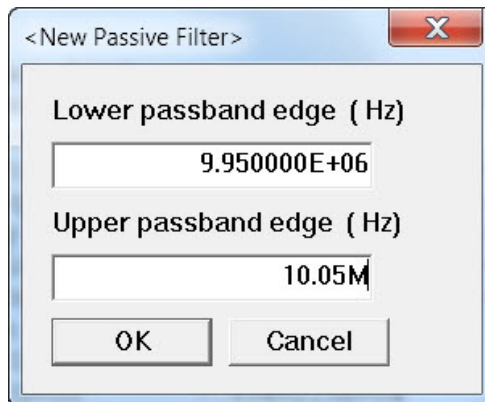
Reject a band of 0.6% width centered at 10 MHz, while the passband width is 1%. That is to say, the lower passband ends at 9.95 MHz, while the upper passband begins at 10.05 MHz. Passband ripple is 0.1 dB, while the stopband loss is required to be about 45 dB. Lastly we will note that the passbands need not be preserved all the way down to zero, or all the way up to infinity, the useful signal is, in fact, restricted to a band of about 10% around 10 MHz. What happens to the filter behavior beyond this band is immaterial.

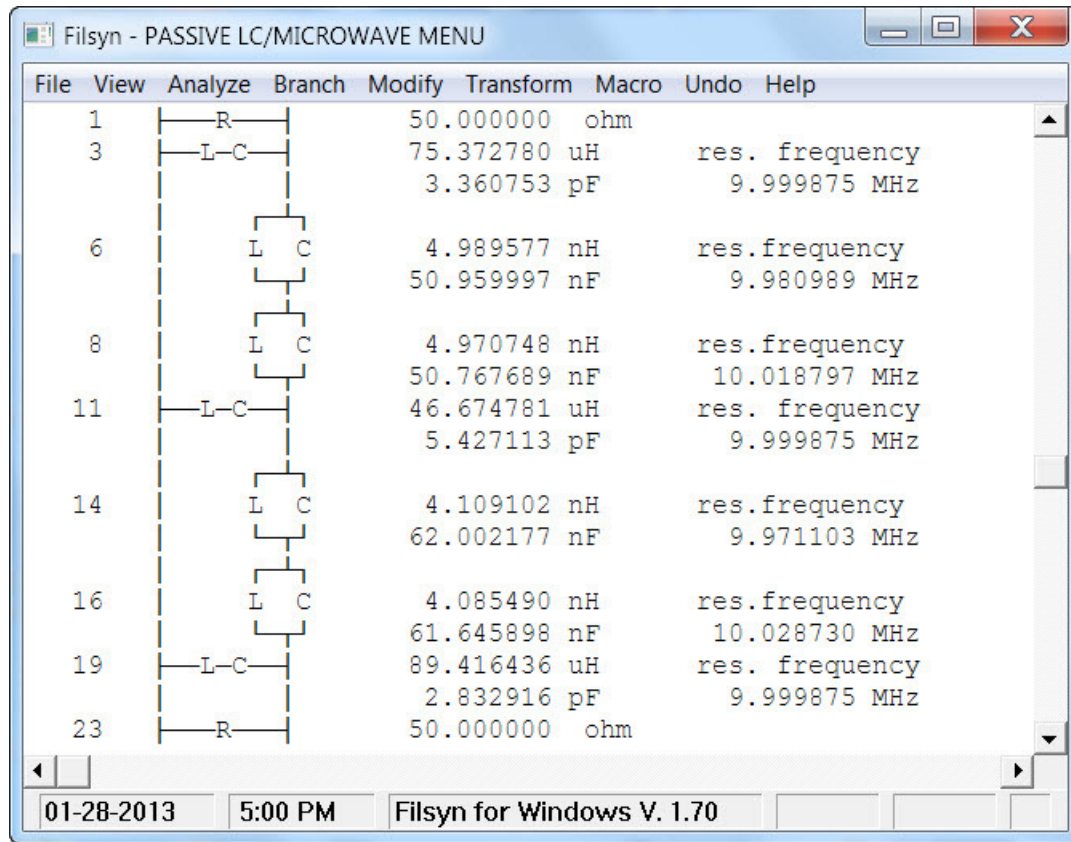
Converting these requirements to a reference lowpass filter, we can readily see, that a fifth order elliptic design will meet them.

Narrow band band-reject filter



Converting this to a band-reject filter, the resulting 10th order filter is shown below:





It is clear that the component values are unrealizable, in particular, those in the parallel resonant circuits are orders of magnitude different from those in the series resonant circuits.

This naturally raises the question whether one set, say, the parallel resonant circuits, could possibly be replaced by corresponding series resonant circuits. The answer is yes, using the equivalence:

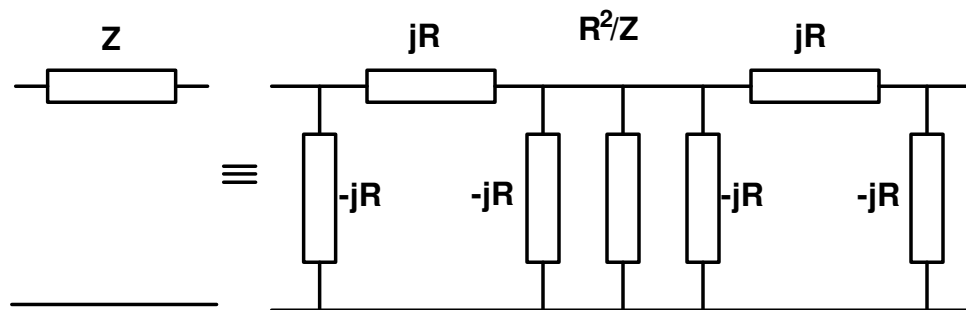
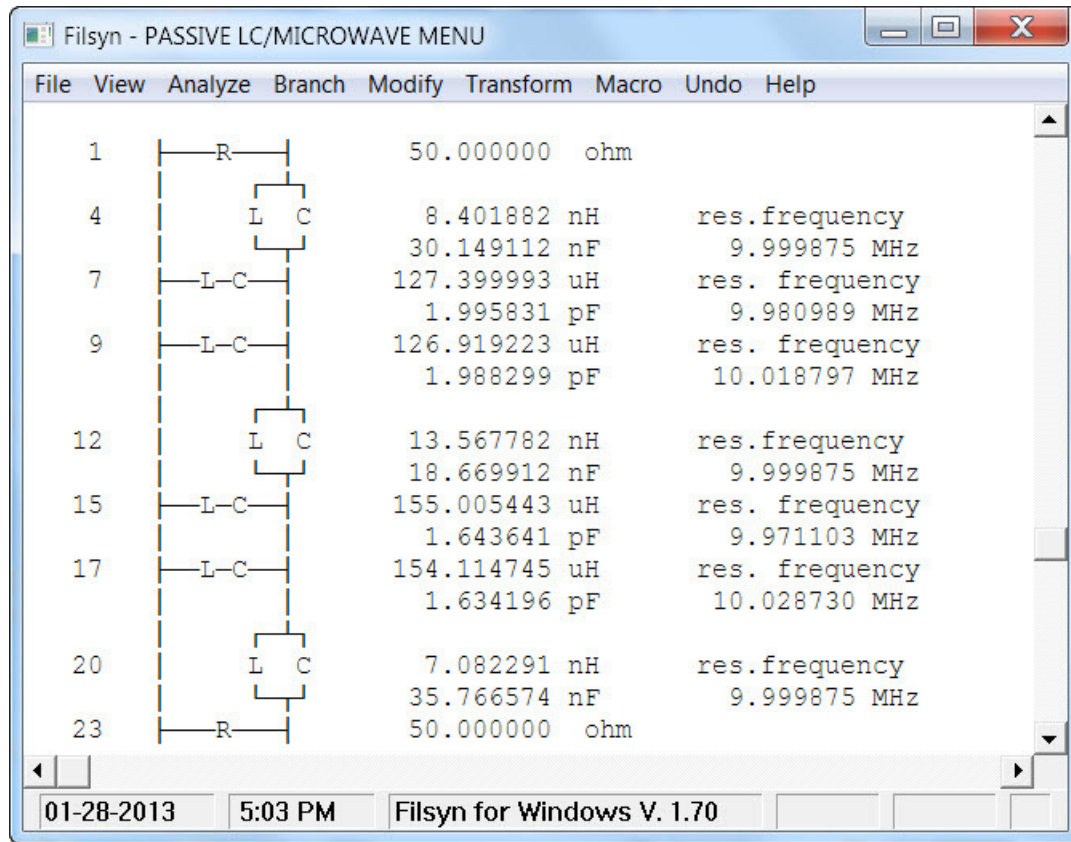


Fig. 40 Equivalence

The pi-section of three constant reactances is an impedance-inverter, and can naturally only be implemented for a narrow frequency band by a pair of inductors and a capacitor, or vice versa.

If we select the value of R to be 50 ohms (our terminating resistor), then the inverted branches (R^2/Z) can be simply picked up from the dual circuit, obtainable by the **Transform->Dual** menu item from the one above:



Actually, we will prefer to use the shunt resonant branches of the dual circuit. Consequently we shall replace branches 4, 12 and 20 in this circuit, by branches 3, 11 and 19 of the original filter. Using $R = 50$ ohms and the center frequency of 10 MHz, the impedance inverter will be a pi-section of two shunt capacitors with a series inductor in between, of values:

$$L_0 = R/2\pi f_0 = 795.7747 \text{ nH}$$

$$C_0 = 1/2\pi f_0 R = 318.3099 \text{ pF}$$

Using the **Branch->Delete** and **Branch->Insert** menu items, we can easily replace the series branches by the parallel branches and also insert these impedance inverters as well. Entering the same one- or two-element branch several times is greatly simplified by the **Branch->Copy** menu item. The resulting structure is as follows:

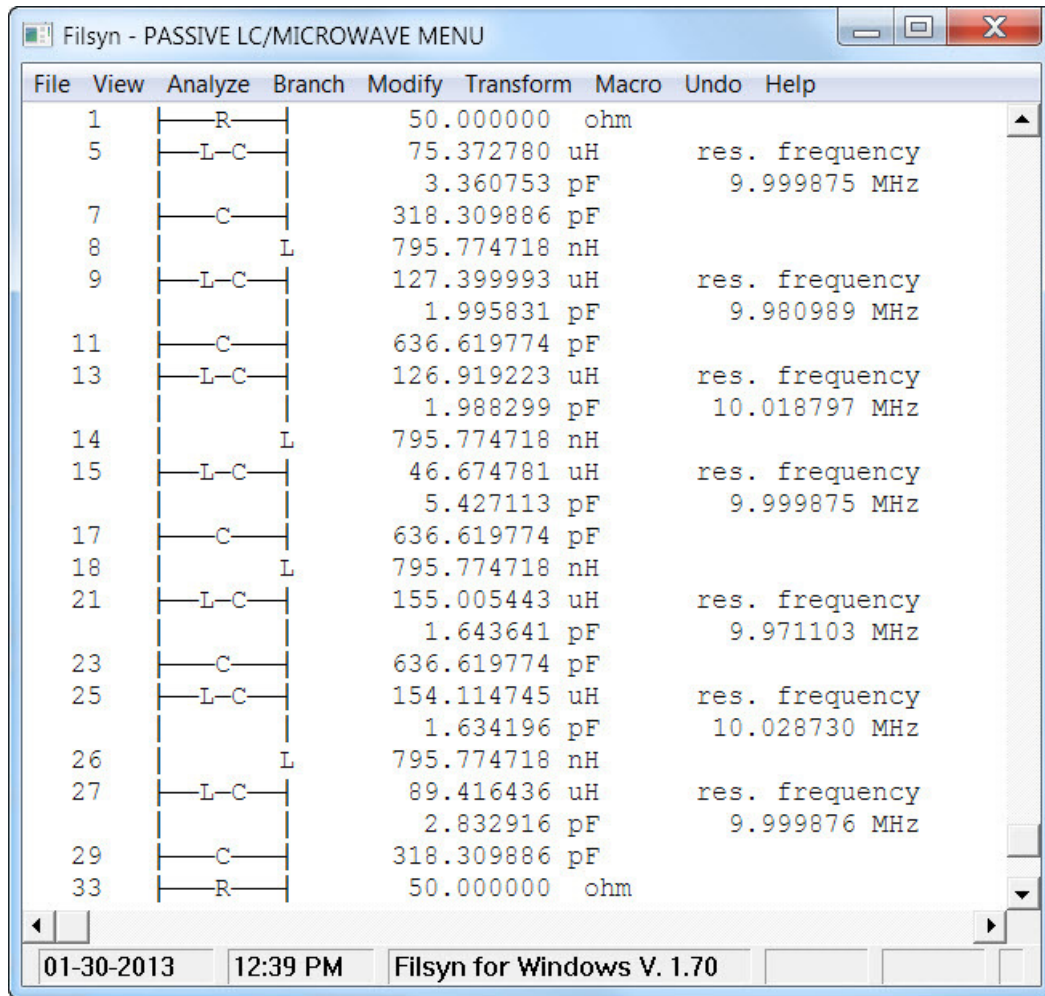
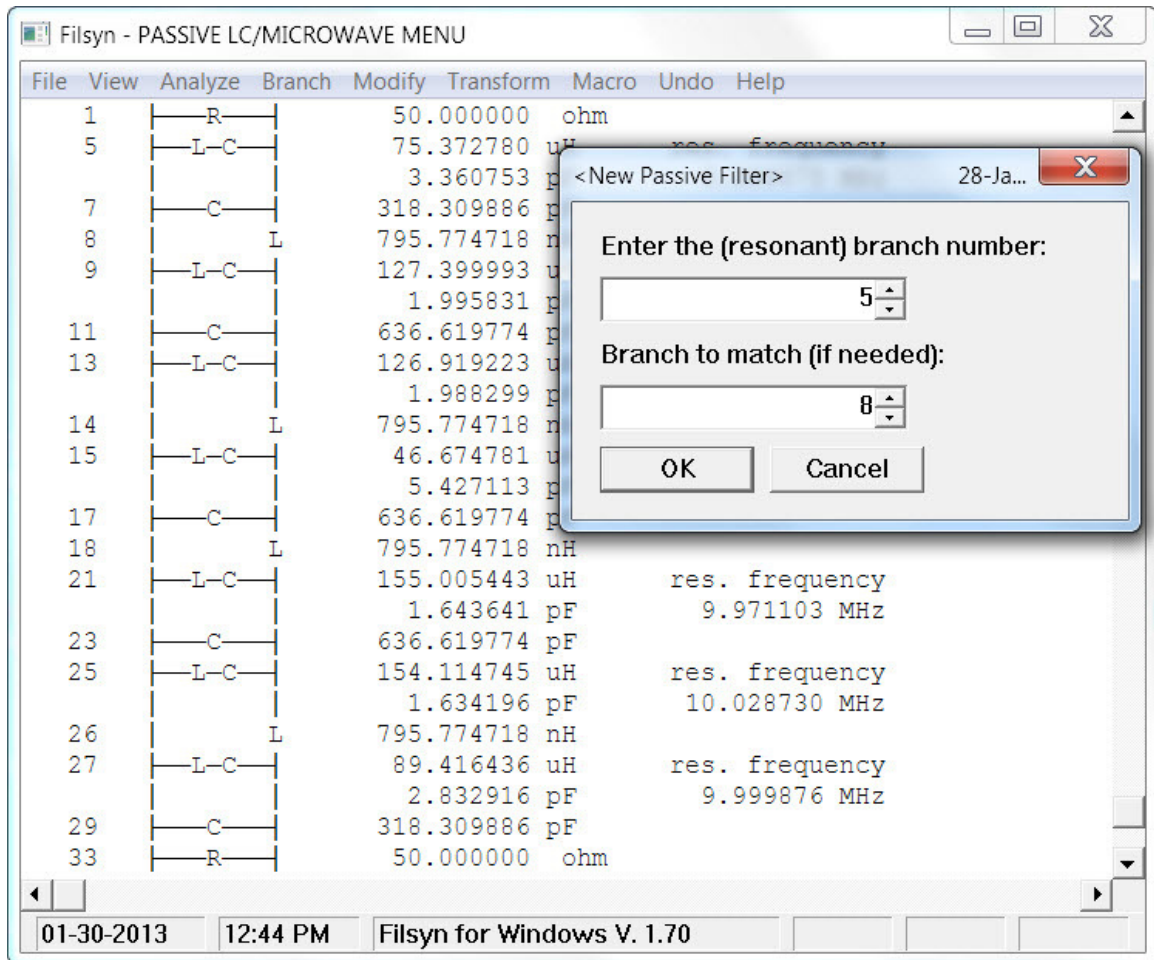


	Diagram	Value	Unit	Notes
1	R	50.000000	ohm	
5	L-C	75.372780	uH	res. frequency
		3.360753	pF	9.999875 MHz
7	C	318.309886	pF	
8	L	795.774718	nH	
9	L-C	127.399993	uH	res. frequency
		1.995831	pF	9.980989 MHz
11	C	636.619774	pF	
13	L-C	126.919223	uH	res. frequency
		1.988299	pF	10.018797 MHz
14	L	795.774718	nH	
15	L-C	46.674781	uH	res. frequency
		5.427113	pF	9.999875 MHz
17	C	636.619774	pF	
18	L	795.774718	nH	
21	L-C	155.005443	uH	res. frequency
		1.643641	pF	9.971103 MHz
23	C	636.619774	pF	
25	L-C	154.114745	uH	res. frequency
		1.634196	pF	10.028730 MHz
26	L	795.774718	nH	
27	L-C	89.416436	uH	res. frequency
		2.832916	pF	9.999876 MHz
29	C	318.309886	pF	
33	R	50.000000	ohm	

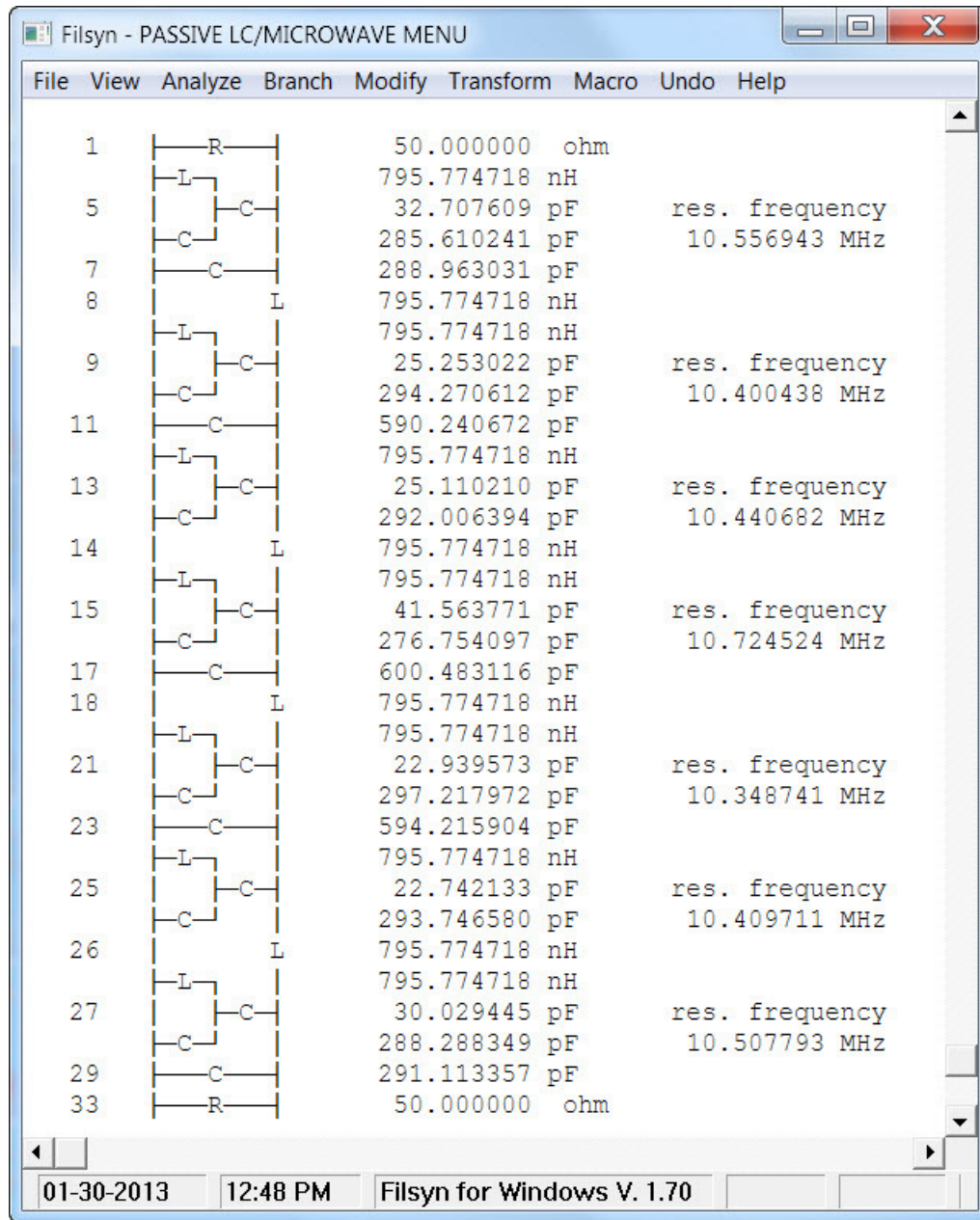
01-30-2013 12:39 PM Filsyn for Windows V. 1.70

The inductor values are now much closer to each other, but those in the shunt branches are still too large. This can be easily changed if we realize, that all shunt resonant branches are also shunted by capacitances and hence subject to the three-element branch conversion **Modify->Convert** menu item. We could, for instance, split off a part of C_7 and use that together with branch 5 to decrease the value of L_5 to anything we wish. A special form of the menu selection may be used to perform this set of steps to make L_5 equal to any other inductive branch in the filter. For instance, to make L_5 equal to L_8 , we simply use the menu as follows:



We just have to make sure that there is a shunt capacitor next to branch 5 in the circuit.

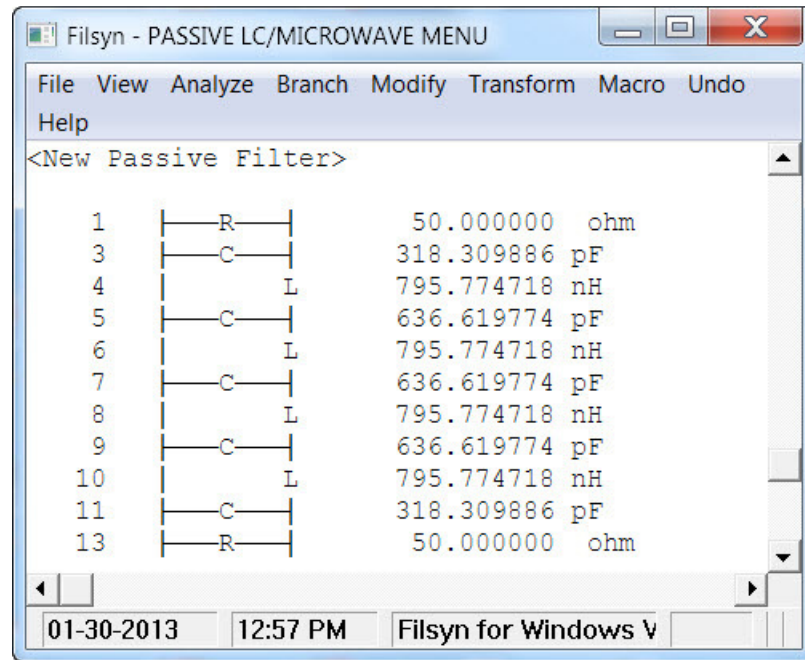
Performing this step on *all* shunt branches will make *all* inductors equal to L_8 and yields the following circuit:



Note that if we wish to set these inductors to be equal to some other value, we can do this by inserting an inductor of the appropriate value somewhere in the circuit, perform these steps and then delete the extra inductor.

These component values are now perfectly acceptable, the only remaining question is, what the overall filter performance is now?

Before doing a detailed frequency-domain analysis, consider what we have done to the circuit. We have introduced four impedance-inverter sections, which create the following lowpass structure:



This lowpass determines how our new band-reject filter will behave (relatively) far away from the reject band (i.e. everywhere, except very near 10 MHz). This lowpass is an *image-parameter* lowpass with a theoretical cutoff of $f_0 \cong 14.1$ MHz. Hence we can expect a reasonable behavior up to about 12 or 13 MHz. Let us analyze this circuit by itself to see what will happen:

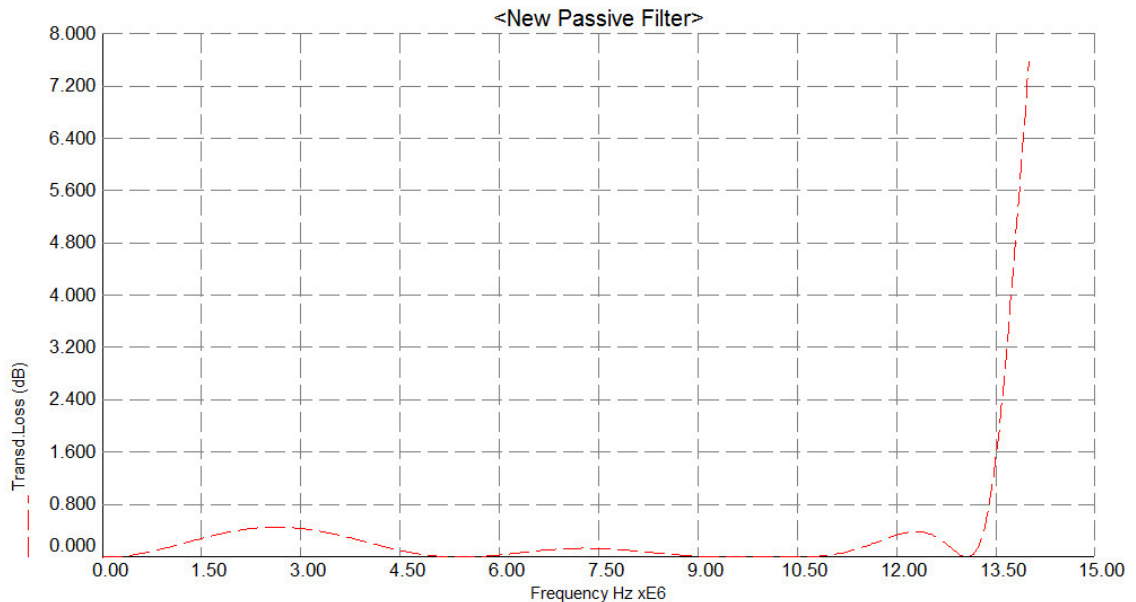


Fig. 41 Image-parameter lowpass

As we can see, up to about 13.4 MHz, there is less than 0.5 dB loss due to these extra elements and this is what we can expect from the band reject filter above.

Analyzing the complete circuit and plotting the results, we have the following, very satisfactory, results. The figure below shows the narrow-band response:

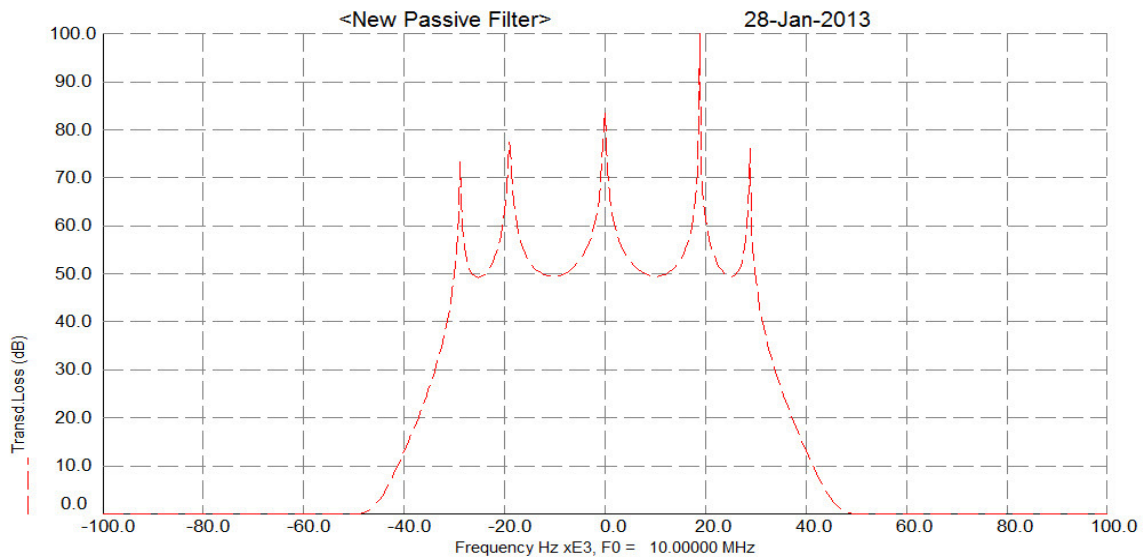


Fig. 42 Narrow-band results

The only visible effect in this narrow band response is the slight increase in passband ripple in the lower passband from 0.1 to 0.14 dB, and to 0.11 dB in the upper passband. Looking at the wideband response from 0 to 15 MHz, we see (Fig. 43 below), that the passbands are flat within less than 0.5 dB from zero frequency up to about 13.4 MHz, just as expected:

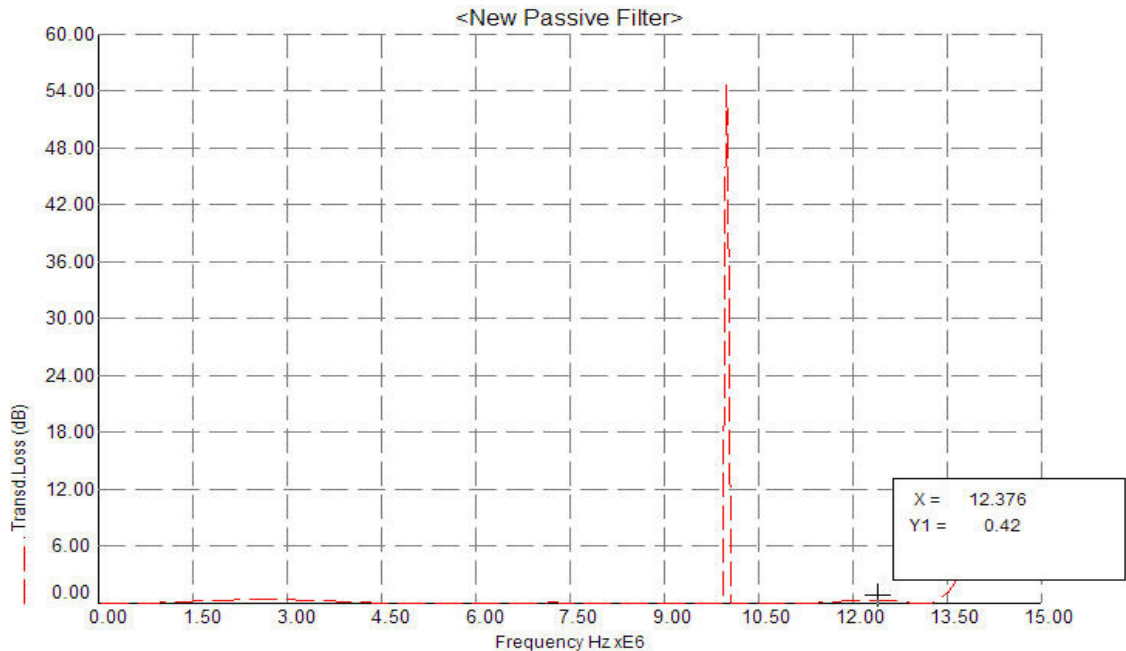


Fig. 43 Wide-band results

APPLICATION NOTE 5

CASCADED LATTICE CRYSTAL FILTER

1. Introduction

In chapter 4 of the manual, we have mentioned the **Modify->Lattice** (ladder-to-lattice conversion) menu option, that may be used to convert sections of a ladder to lattice form. To design more complex filters, we must learn additional details. We will accomplish this with the help of an example.

2. Filter Specification

Assume a very narrow band (3 kHz passband) single-sideband filter, centered at 1 MHz with 2 kHz transition region at the lower edge of the passband, but only 600 Hz at the upper edge. We estimate that for 70 dB stopband loss, we will need four finite transmission zeros – one below the passband and three above. These could be implemented by two cascaded lattice sections. To simplify the realization of this filter, we require a shunt inductor be placed between the lattices and at both ends. Additional capacitors will also be needed. We need therefore, a fifth order transmission zero at zero (three shunt L's with two series C's in between) and a single zero at infinity representing a shunt C. Terminations are set to 1 Kohms, but can be changed later. This circuit is a conventional structure and using the **Placer** option to obtain the optimum transmission zero locations, the data input screen is as follows:

The screenshot shows the 'New Filter' dialog box with the following settings:

- Filter kind:** LC (selected), Microwave, Active RC, IIR Digital.
- Quarter-wave/Sampling freq (Hz):** 0.00000
- Filter type:** Low pass, High pass, Band pass (selected).
- Lower passband freq (Hz):** 1.000000E+06
- Upper passband freq (Hz):** 1.003000E+06
- Passband type:** Max. flat, Equal ripple (selected), Functional, Sloping.
- Band-edge loss/return loss (dB):** 0.100000000
- Loss Slope (dB/oct):** 6.00000000
- Flat loss (dB):** 0.00000000
- Function Type:** E (selected), F.
- Multiplier:** 0.00000000
- Stopband:** Monotonic, Equal min, Placer (selected), Specified.
- Lower stopband freq (Hz):** 0.00000
- Loss (dB):** 50.0000000
- Upper stopband freq (Hz):** 0.00000
- Loss (dB):** 50.0000000
- Detail Parameters:** # of zeros at zero: 5, # of zeros at infinity: 1, # of unit elements: 0.
- Parametric:** Conventional (selected), Parametric, Matching.
- R1:** 1.000000E+03
- R2:** 1e3
- ZS parameter:** -1
- Q for predistortion:** 0.00000000
- Predistortion:** Passband only (selected), Pass/stop bands.
- Shifted bandpass trans. Center frequency (Hz):** 0.00000
- Odd parametric:** unchecked
- Save design data:** unchecked
- Buttons:** OK, Cancel.

where the **Detail parameters** window contains the breakpoints:

Filter Parameters

Requirements		Fixed zeros		Movable zeros	
	Frequency (Hz)	Loss (dB)		Frequency (Hz)	Multiplicity
1	998.000000E+03	70.00000000	1	0.000000	0
2	1.003600E+06	70.00000000	2	0.000000	0
3	0.000000	0.00000000	3	0.000000	0
4	0.000000	0.00000000	4	0.000000	0
5	0.000000	0.00000000	5	0.000000	0
6	0.000000	0.00000000	6	0.000000	0
7	0.000000	0.00000000	7	0.000000	0
8	0.000000	0.00000000	8	0.000000	0
9	0.000000	0.00000000	9	0.000000	0
10	0.000000	0.00000000	10	0.000000	0
11	0.000000	0.00000000			

or no. of zeros below passband:

and/or no. of zeros above passband:

OK Cancel

The optimization was uneventful converging in three iterations, and forcing one more step yielded the following printout:

Filsyn - MAIN MENU

File Design Analysis Help

Iteration no. 1

transmission zeros	multiplicity
9.9780734D+05	1
1.0077450D+06	1
1.0039940D+06	1
1.0036342D+06	1

fmin (hz)	amin (db)	dmin (db)
9.9800D+05	73.7713	3.7713
9.9708D+05	73.7705	3.7705
1.0100D+06	102.0559	32.0559
1.0045D+06	73.7824	3.7824
1.0038D+06	73.7701	3.7701
1.0036D+06	73.7703	3.7703

01-28-2013 1:41 PM Filsyn for Windows V. 1.70

Optimization

Options

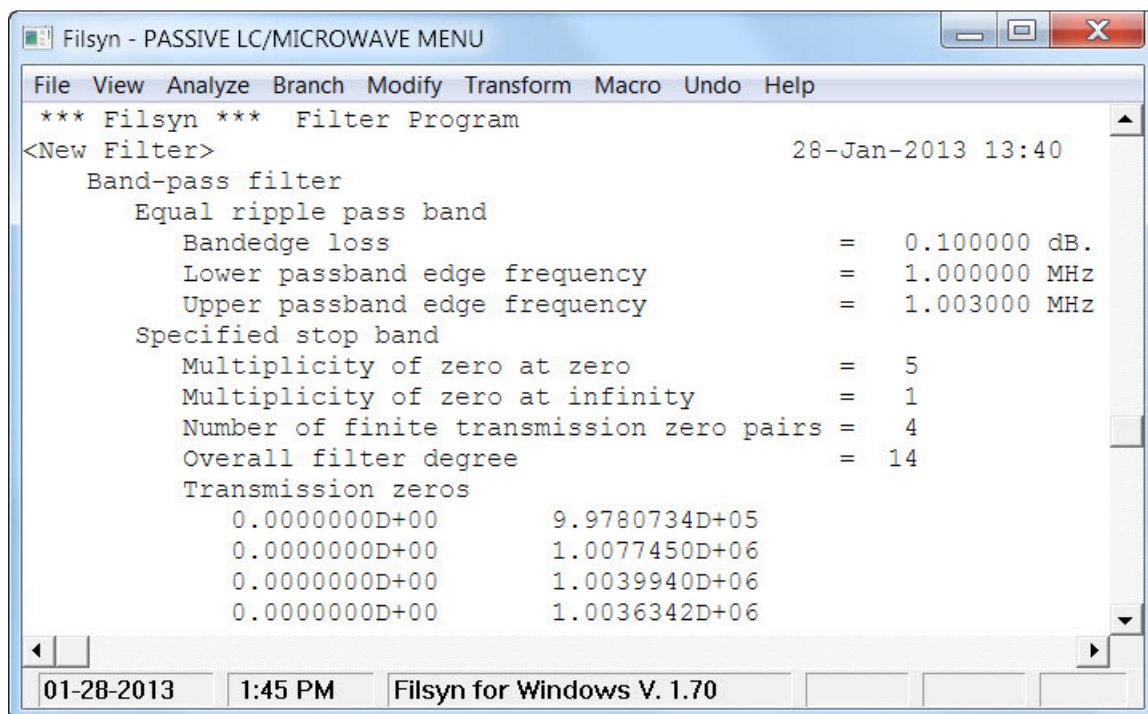
☐ Re-optimize

☒ Continue to synthesis

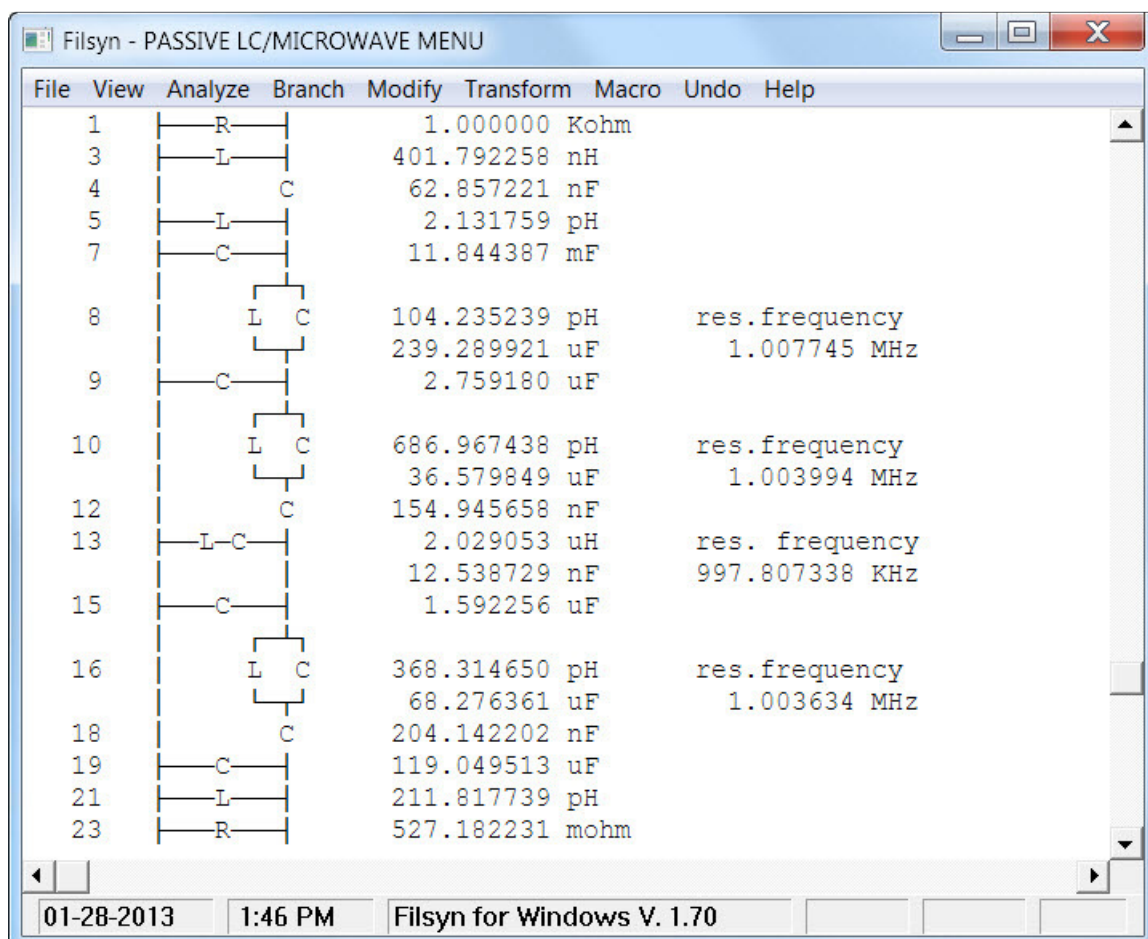
☐ Modify some parameters

OK Cancel

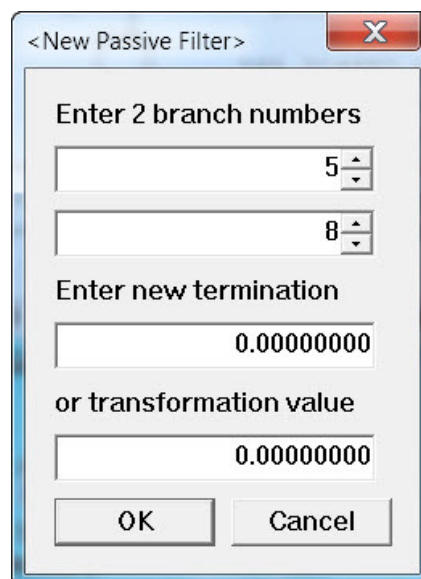
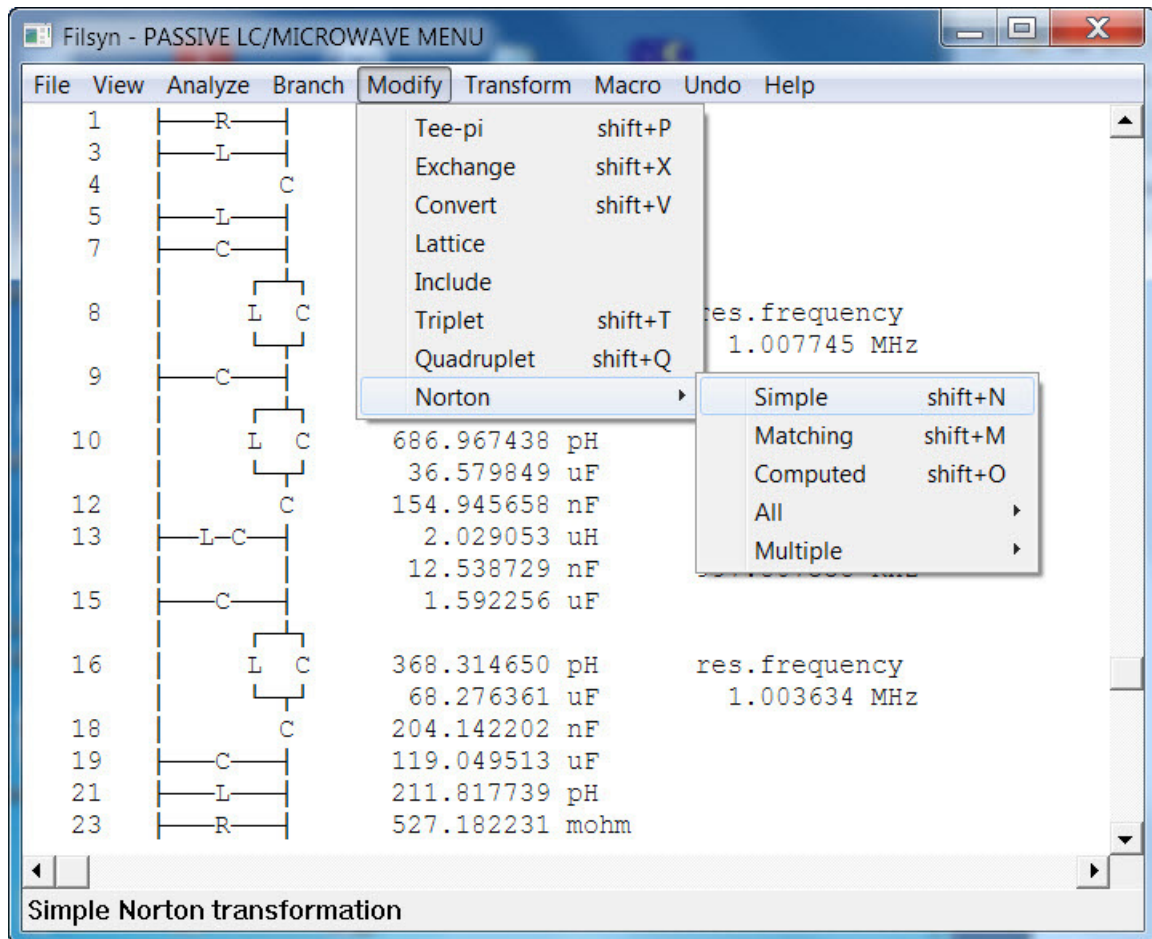
Entering the passive synthesis segment, even though we need a specific structure, we shall use the computer-generated configuration as the starting point. The reason for this is that it is much simpler than the manual synthesis and it gives us a configuration that is pretty close to what we need. Also we have all the tools to modify the configuration in any way we need. First the summary:



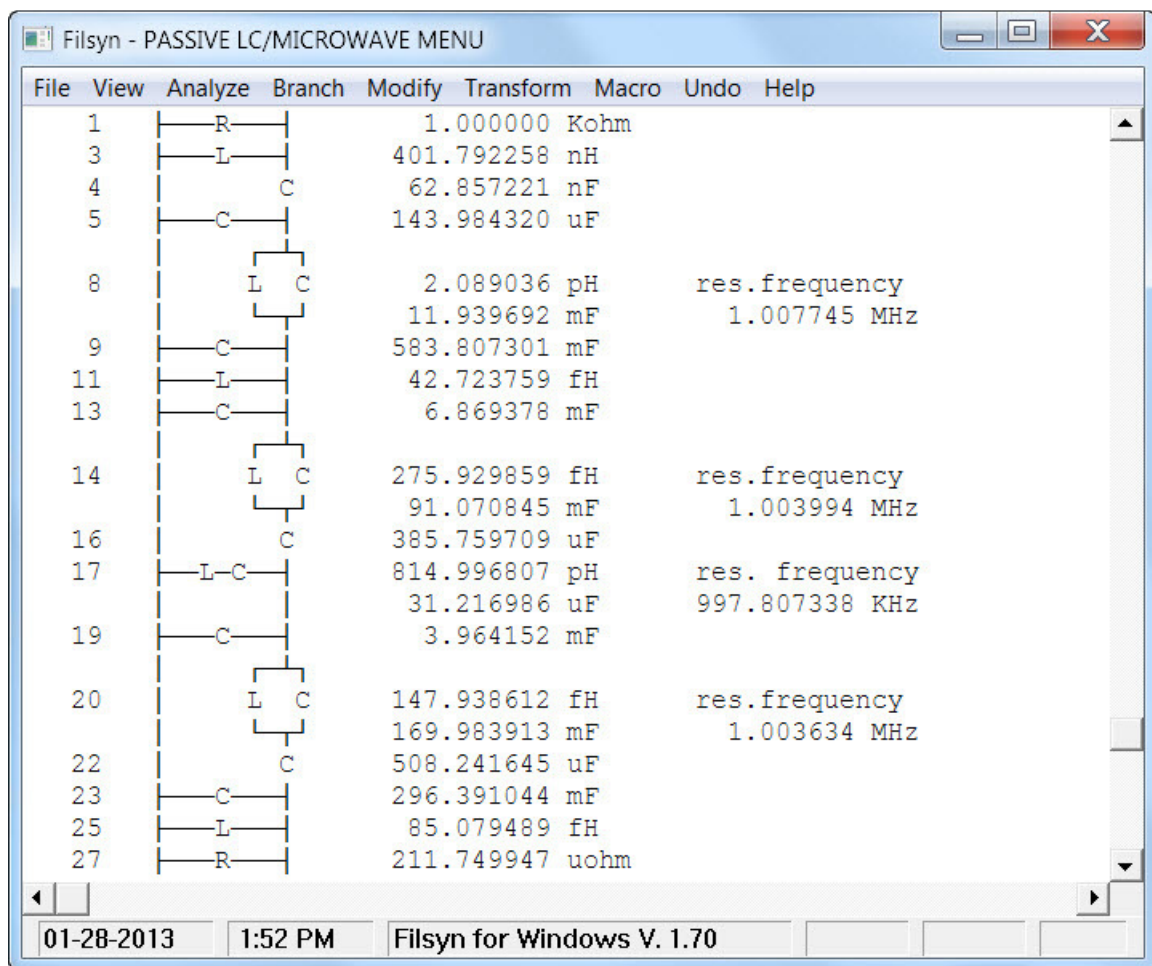
Then the structure and element values:



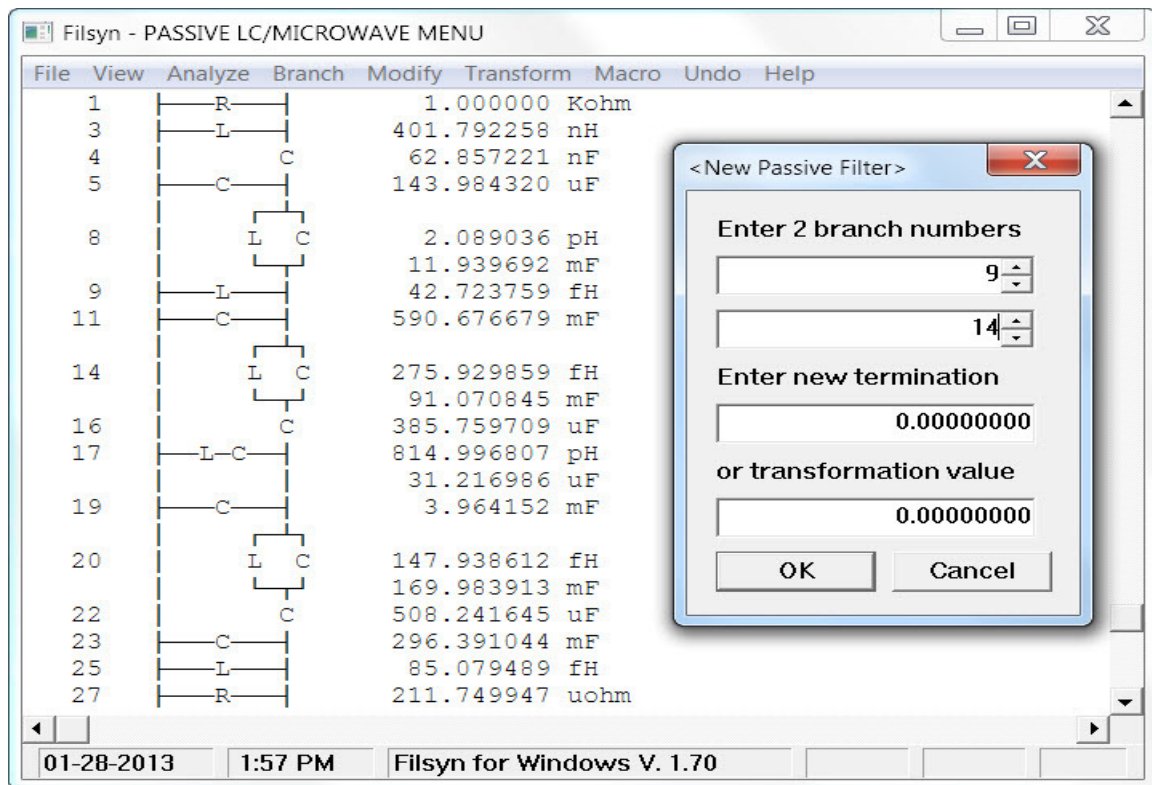
The only rearrangement needed is to move the shunt L_5 to the other side of the series branch 10. This need two steps together with some cleaning up. First we perform a simple Norton transformation using branches 5 and 8:



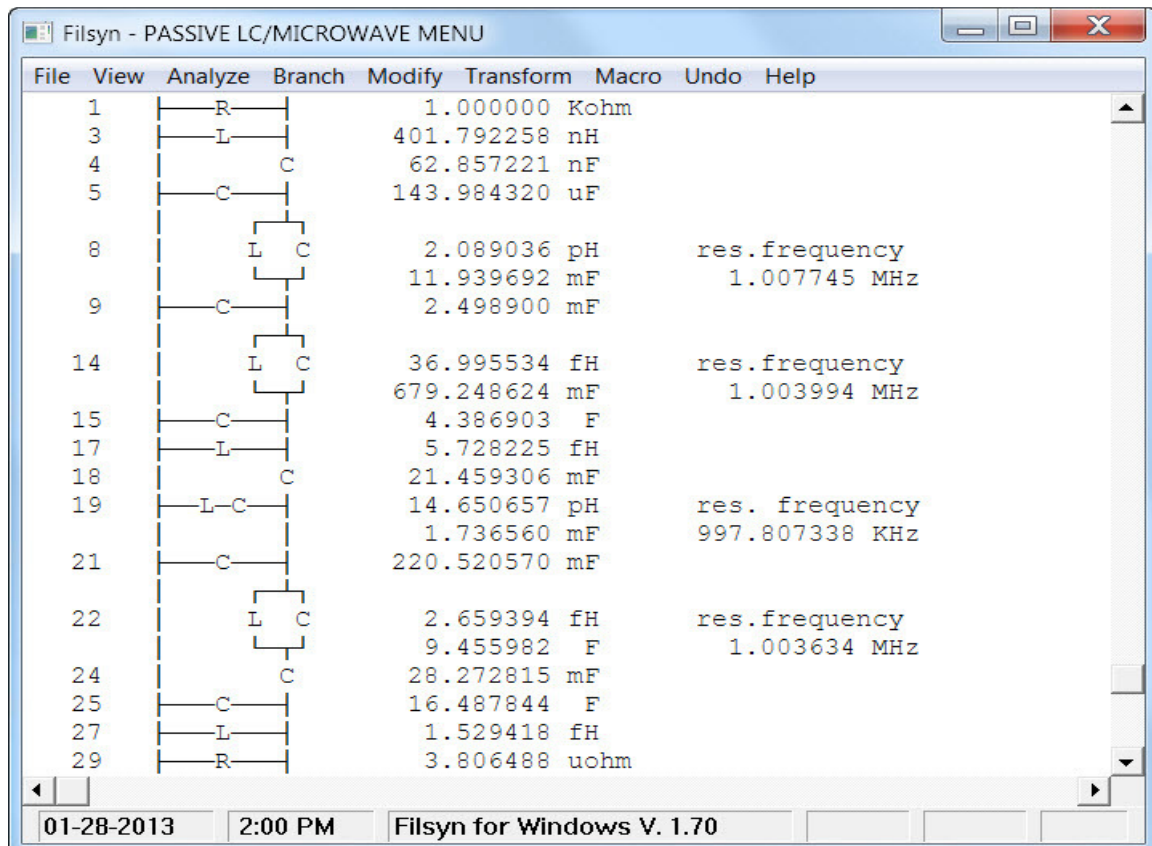
This yields:



This is fine but we need to interchange branches 9 and 11 and then combine the capacitors C_{11} and C_{13} . This is followed by the second Norton transformation step using branches L_9 and L_{14} :

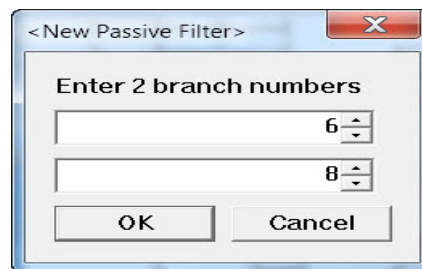
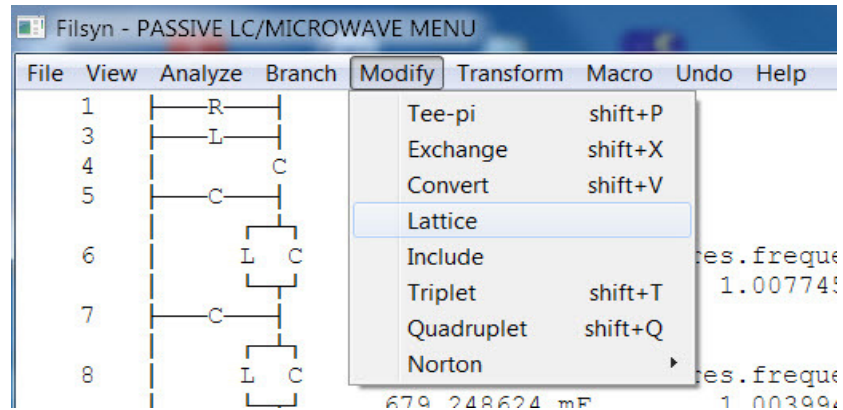


which gives us the configuration we need:

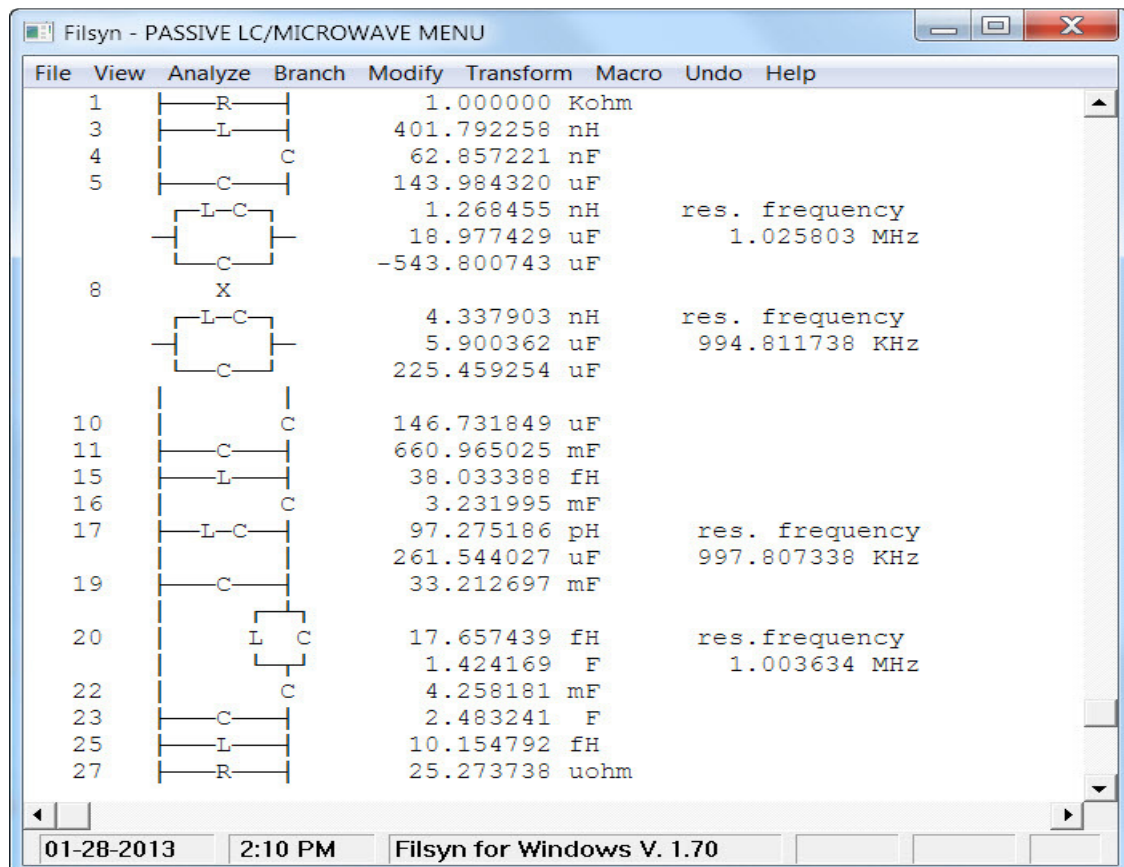


Cascaded lattice crystal filter

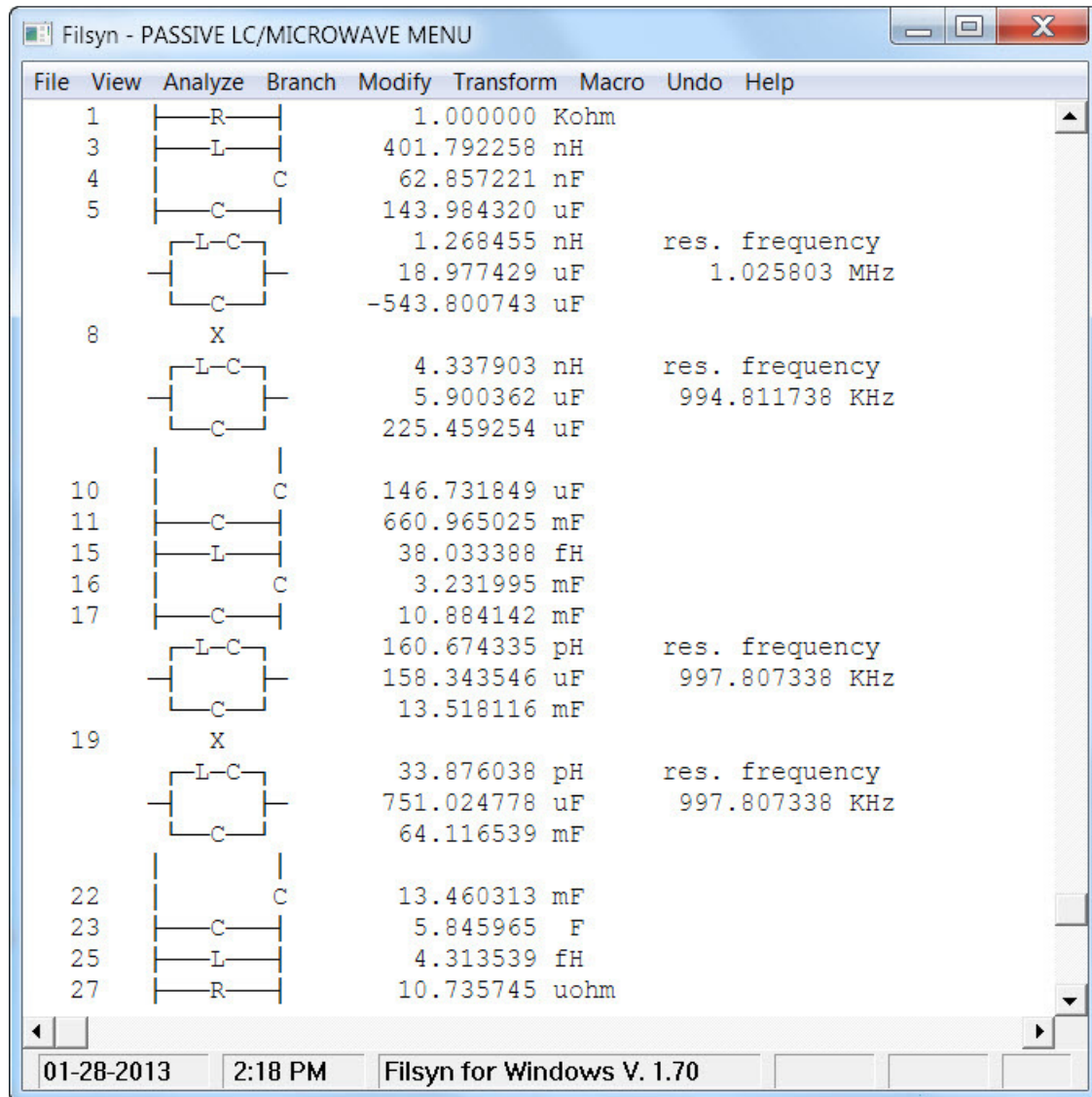
At this point we convert resonant branches 6 and 8 into a lattice, using the **Modify->Lattice** menu:



Yielding:

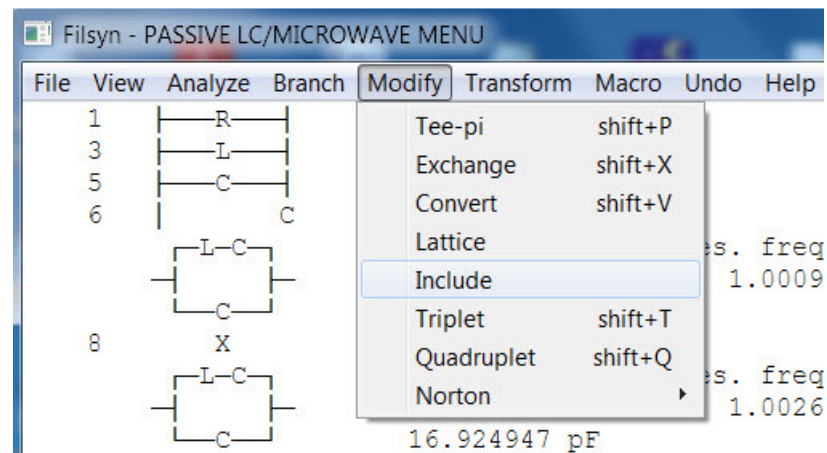


Next we do the same thing first interchanging branches 17 and 19 and then converting resonant branches 19 and 20, giving us a structure that is pretty close to the final one:

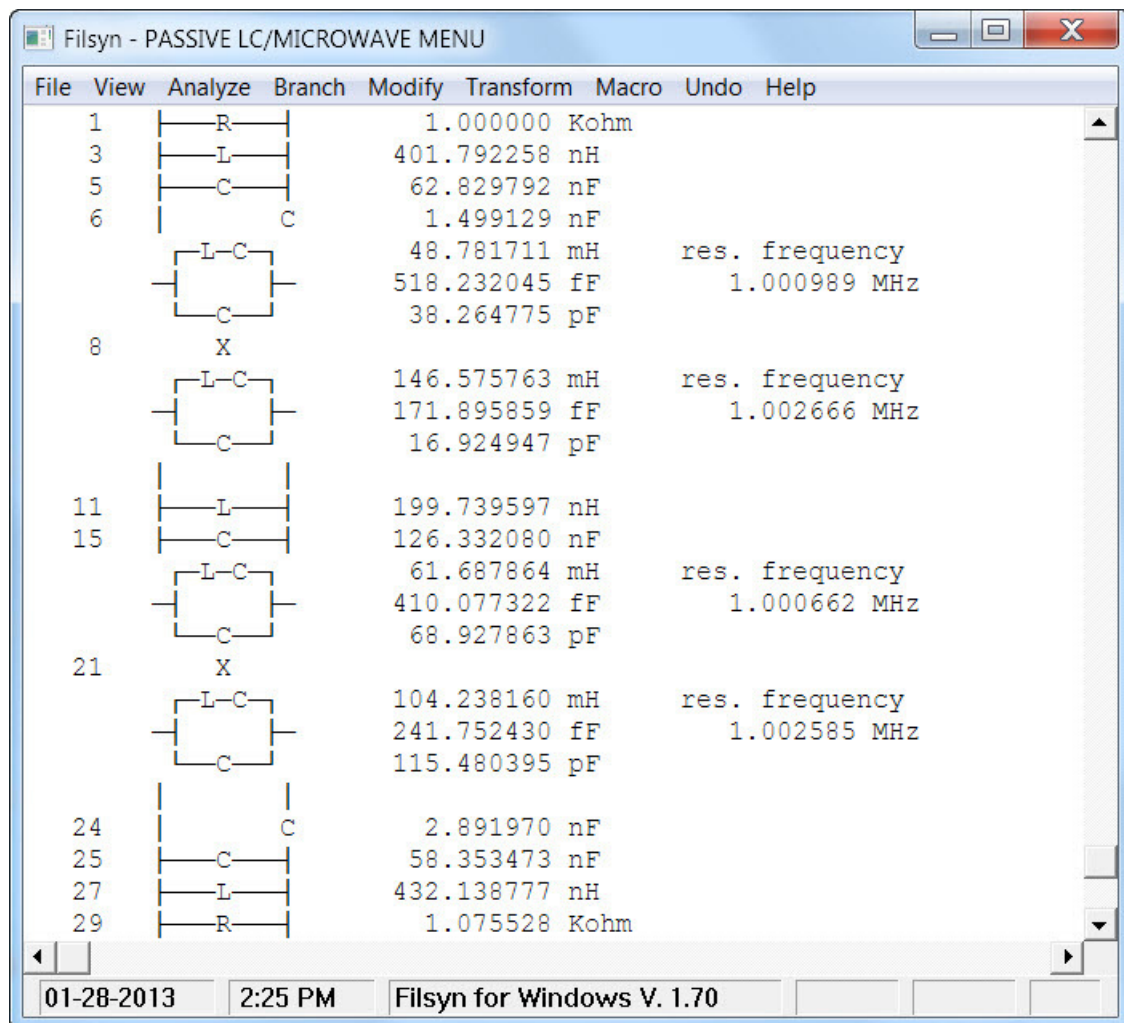


Let's not worry about the negative capacitor that will disappear in a minute. What we need to do are the following steps:

- Flipping the $C_4 - C_5 L$ section around using a simple Norton transformation step
- Do the same thing of the L section C_{16} and C_{17}
- Simplify the three shunt elements in the center of the structure using the **Branch->Interchange** and **Branch->Combine** menu items
- Finally use the **Modify->Include** menu to take the series capacitors on both sides of lattice number 8 and shift them inside the lattice:



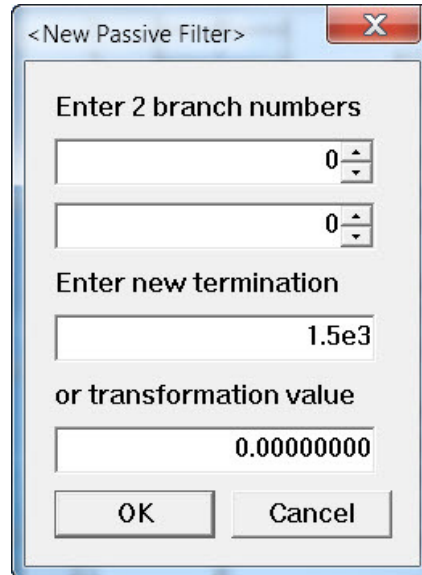
After having done that as well as repeating it for the capacitors flanking lattice 21 as well, we have the following circuit:



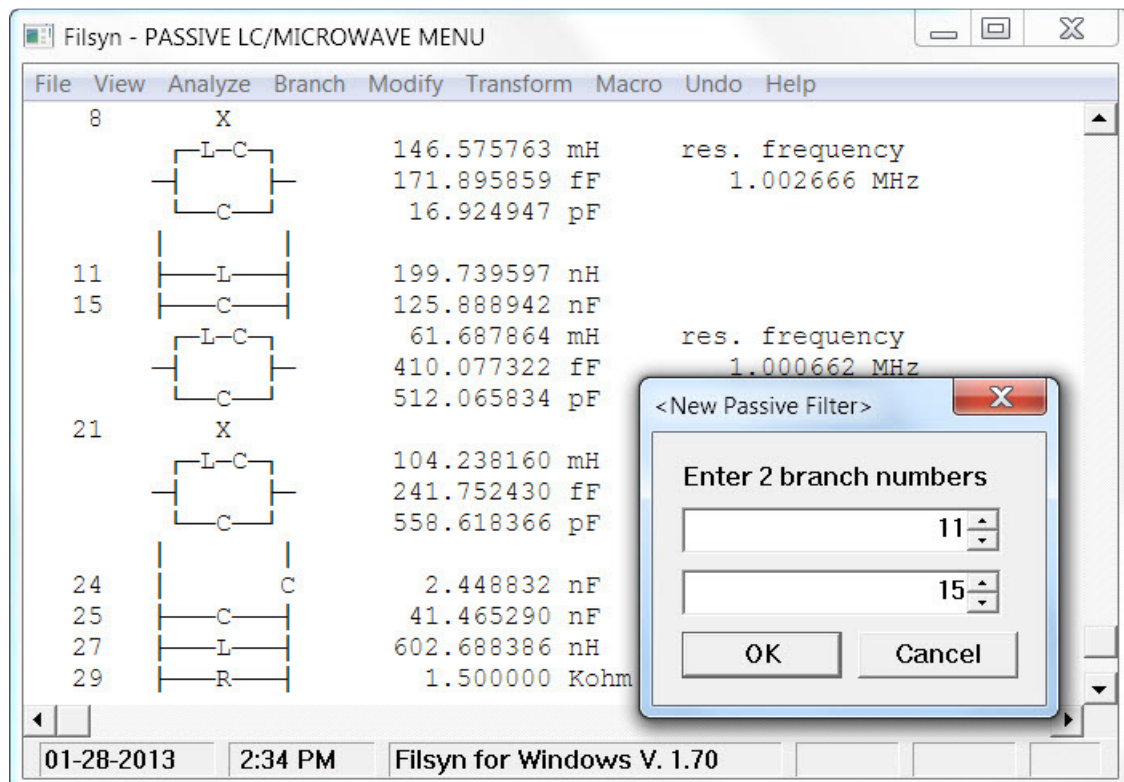
We are nearly done, we have just a couple of minor corrections to make. One is the slight deviation of the output termination, the other is that the crystal capacitance ratios are not

quite large enough, the smallest is about 74. This ratio, the ratio of the parallel C to the series one is a critical parameter and has to be over 200.

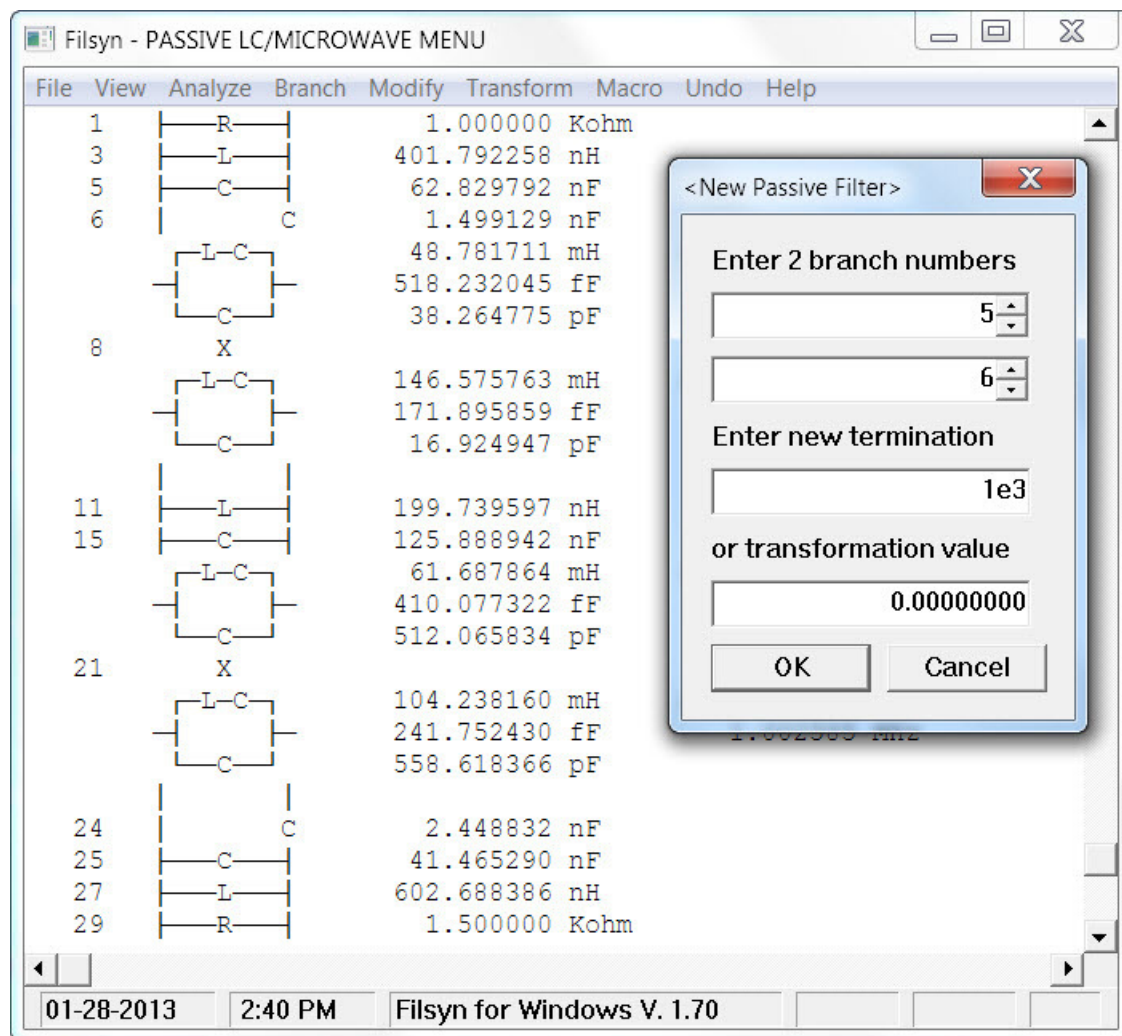
We can take care of both using a simple trick. First we use a Norton transformation step again to change the output termination to 1.5 Kohms:



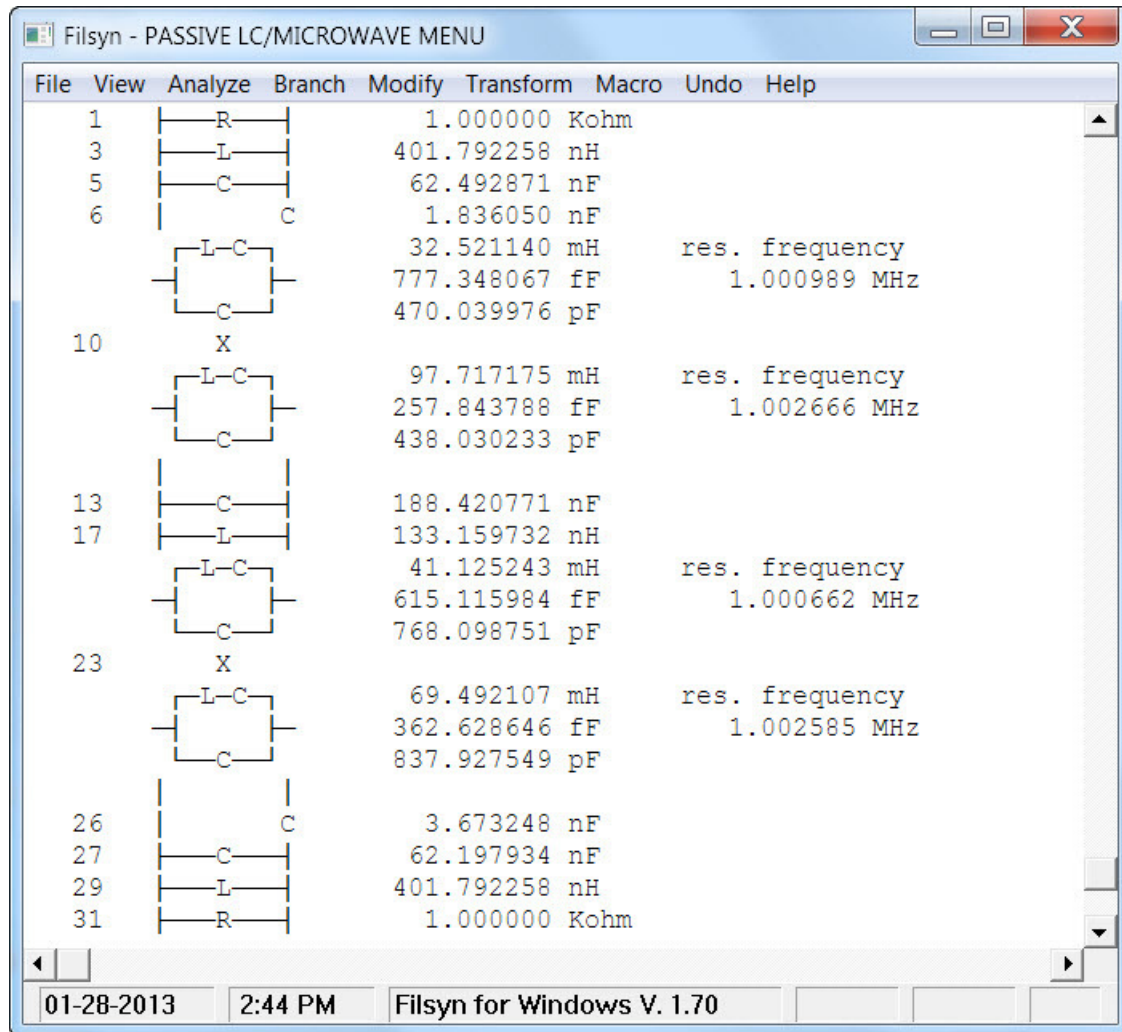
The resulting circuit shows, that we now have a shunt capacitor on both sides of lattice 21 that we can shift inside using the **Modify->Include** menu item again. Next we interchange branches 11 and 15 to have the capacitor next to lattice 8:



and finally we use branches 5 and 6 to change the output termination back to 1 Kohms:

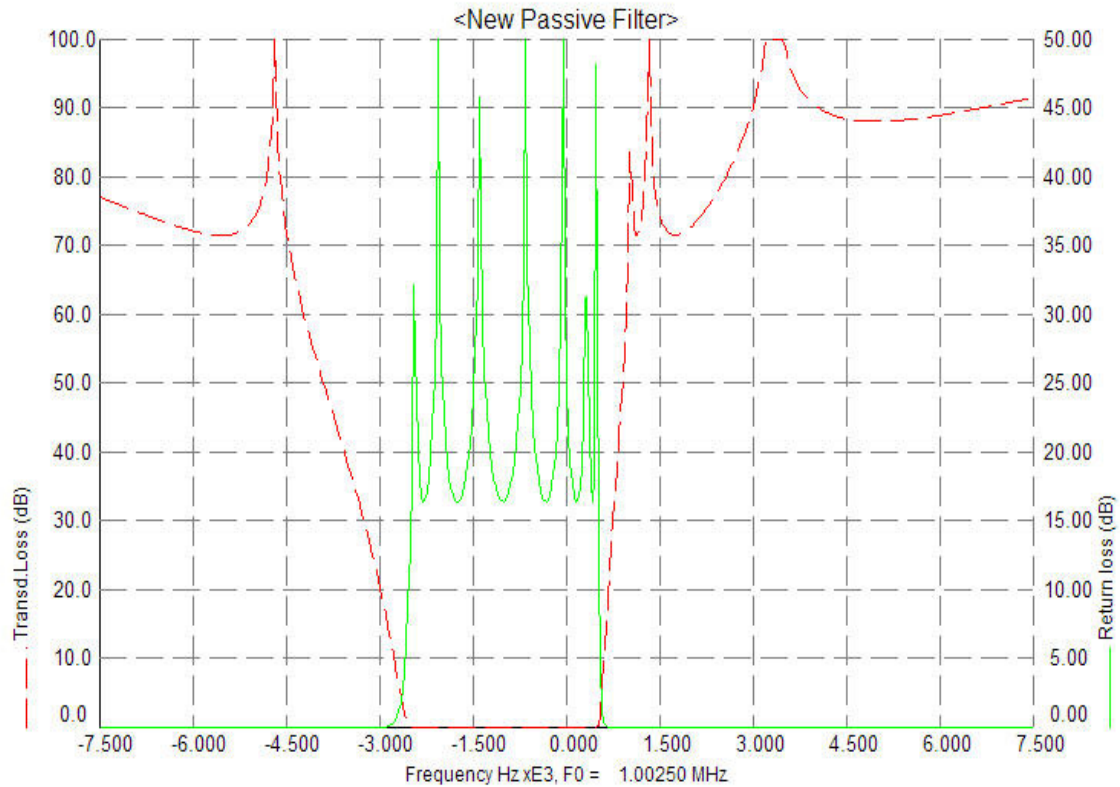


This puts a capacitor across the front end of lattice 8 (actually now 10) which we can now include into it and this yields our final result:



The last couple of steps have included some arbitrary numbers and this illustrates the fact that in the case of bandpass filters we usually have a number of degrees of freedom. One can use these to further manipulate the circuit if that is necessary. The results we obtained are very convenient, the smallest crystal capacitance ratio is a shade over 600, very satisfactory. The shunt L's may be used to modify the impedance levels of the crystal lattices.

Finally, we show below the results of an analysis, performed on this last circuit.



3. Additional Comments

In certain cases one might want to convert an internal ladder segment to a lattice, when one or both of the resonant branches are single inductors, instead of resonant circuits. The **Modify->Lattice** menu option will *not* perform this step directly, because it is looking for two resonant circuits. One can overcome this problem by replacing the single inductor with a resonant branch which has a very large series capacitor or a very small shunt capacitor and the same inductor. This will change the circuit performance minimally and will permit the menu item to operate properly.

APPLICATION NOTE 6

DELAY EQUALIZATION OF DIGITAL AND MICROWAVE HIGHPASS FILTERS

1. Introduction

In the case of analog filters (passive LC or active RC), low- and band-pass filters may be delay equalized, but highpass filters may not, since the extent of their passband is infinite and hence would need an infinite number of equalizer sections. However, this is not true for IIR digital or microwave filters where highpass implementations also have a finite passband width, therefore it *is* possible to apply delay equalization. Our version of the delay equalizer subprogram though is unable to perform this function directly, because it handles the problem by converting it to an analog problem with an infinite bandwidth. Nevertheless, the operation is still available to the users of **Filsyn**, using a simple trick.

For digital filters, the fact is that a highpass filter can be converted into a lowpass one by replacing every critical frequency f by $(f_s/2 - f)$, where f_s is the sampling frequency. One can then delay equalize this lowpass and finally reconvert the delay equalizer to the corresponding highpass form by performing the same frequency shift once again. It further turns out, that this shift corresponds to the very simple operation of flipping all z -plane singularities with respect to the imaginary (vertical) z -axis, that is to say, simply changing the signs of all the real parts of the poles and zeros. If we are using the factored form, we can simply change the signs of all the linear terms of the quadratic factors.

2. Digital example

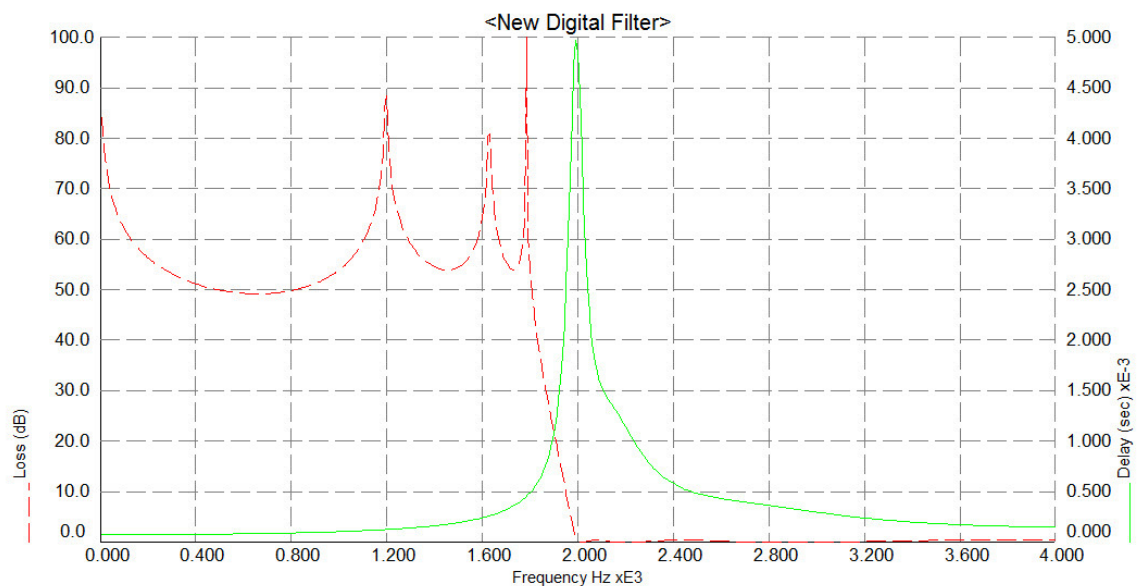
Consider, for example, a digital highpass filter with the following specification: 10 KHz sampling rate, passband starts at 2 KHz with 0.5 dB loss ripple and we need 50 dB suppression from 1 KHz to 1800 Hz. The **Placer** segment of the program will be needed and we will specify 25 dB loss below 1 KHz. The resulting filter is of 7th order and is shown below; the computed performance is in the next figure.

Delay equalization of highpass...

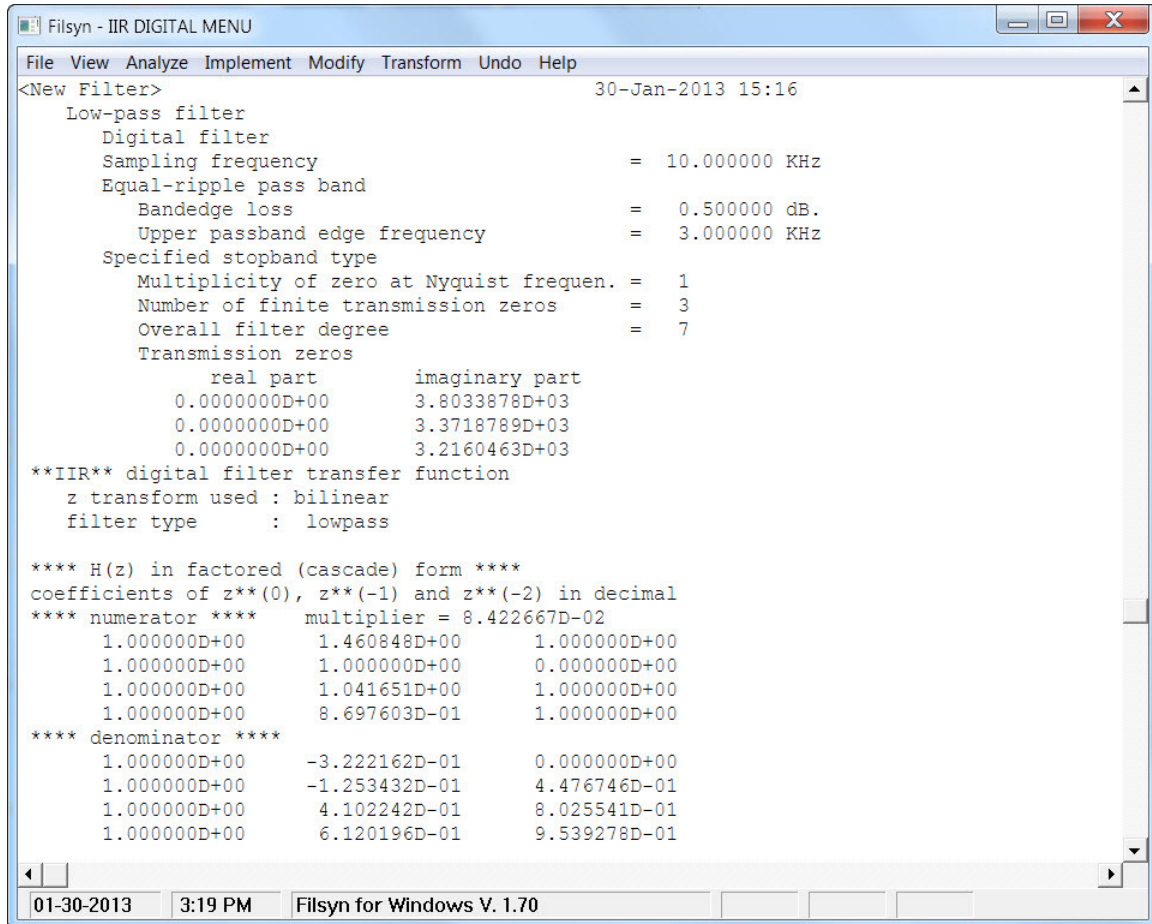
```

Filsyn - IIR DIGITAL MENU
File View Analyze Implement Modify Transform Undo Help
<New Filter> 30-Jan-2013 15:01
  High-pass filter
    Digital filter
    Sampling frequency = 10.000000 KHz
    Equal-ripple pass band
      Bandedge loss = 0.500000 dB.
      Lower passband edge frequency = 2.000000 KHz
    Specified stopband type
      Multiplicity of zero at zero = 1
      Number of finite transmission zeros = 3
      Overall filter degree = 7
      Transmission zeros
        real part      imaginary part
        0.000000D+00   1.1966122D+03
        0.000000D+00   1.6281211D+03
        0.000000D+00   1.7839537D+03
  **IIR** digital filter transfer function
    z transform used : bilinear
    filter type      : highpass

  **** H(z) in factored (cascade) form ****
  coefficients of z**(0), z**(-1) and z**(-2) in decimal
  **** numerator ****      multiplier = 8.422667D-02
    1.000000D+00   -1.460848D+00   1.000000D+00
    1.000000D+00   -1.000000D+00   0.000000D+00
    1.000000D+00   -1.041651D+00   1.000000D+00
    1.000000D+00   -8.697603D-01   1.000000D+00
  **** denominator ****
    1.000000D+00   3.222162D-01   0.000000D+00
    1.000000D+00   1.253432D-01   4.476746D-01
    1.000000D+00   -4.102242D-01   8.025541D-01
    1.000000D+00   -6.120196D-01   9.539278D-01
  
```



There are several ways to delay equalize this filter. One can convert this filter to a lowpass, by changing the signs of all linear terms in the quadratic factors of its transfer function, or one can simply redesign it as a lowpass as follows. We flip the requirements as outlined above, with respect to the 5 KHz Nyquist rate: the passband is from 0 to 3 KHz, while the 50 dB stopband is from 3.2 KHz to 4 KHz; otherwise the parameters are the same. Note please that the filter coefficients are exactly the same as in the highpass filter, except the linear coefficients have the opposite signs.



```

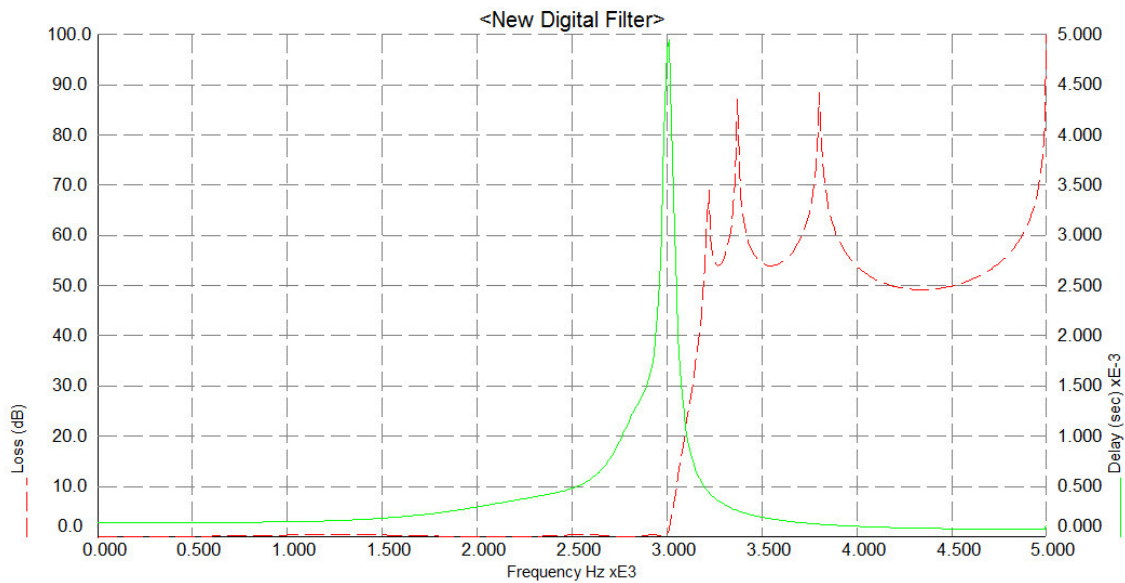
Filsyn - IIR DIGITAL MENU
File View Analyze Implement Modify Transform Undo Help
<New Filter> 30-Jan-2013 15:16
Low-pass filter
  Digital filter
  Sampling frequency = 10.000000 KHz
  Equal-ripple pass band
    Bandedge loss = 0.500000 dB.
    Upper passband edge frequency = 3.000000 KHz
  Specified stopband type
    Multiplicity of zero at Nyquist frequen. = 1
    Number of finite transmission zeros = 3
    Overall filter degree = 7
    Transmission zeros
      real part    imaginary part
      0.000000D+00 3.8033878D+03
      0.000000D+00 3.3718789D+03
      0.000000D+00 3.2160463D+03
  **IIR** digital filter transfer function
    z transform used : bilinear
    filter type      : lowpass

**** H(z) in factored (cascade) form ****
coefficients of z**(0), z**(-1) and z**(-2) in decimal
**** numerator **** multiplier = 8.422667D-02
  1.000000D+00 1.460848D+00 1.000000D+00
  1.000000D+00 1.000000D+00 0.000000D+00
  1.000000D+00 1.041651D+00 1.000000D+00
  1.000000D+00 8.697603D-01 1.000000D+00
**** denominator ****
  1.000000D+00 -3.222162D-01 0.000000D+00
  1.000000D+00 -1.253432D-01 4.476746D-01
  1.000000D+00 4.102242D-01 8.025541D-01
  1.000000D+00 6.120196D-01 9.539278D-01
01-30-2013 3:19 PM Filsyn for Windows V. 1.70

```

The computed response is also flipped around:

Delay equalization of highpass...



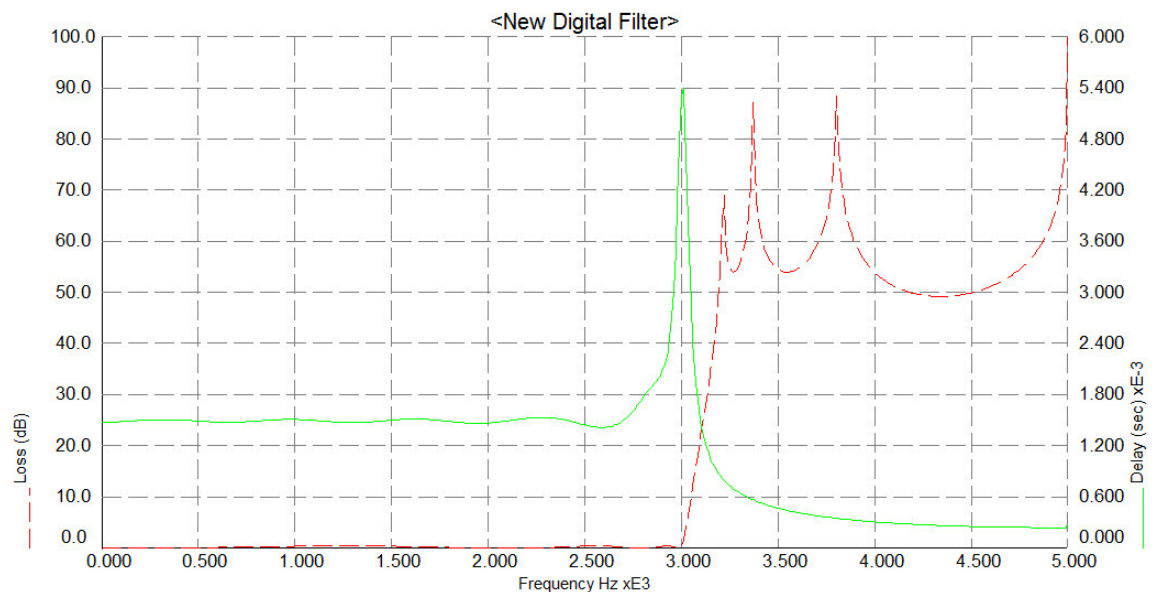
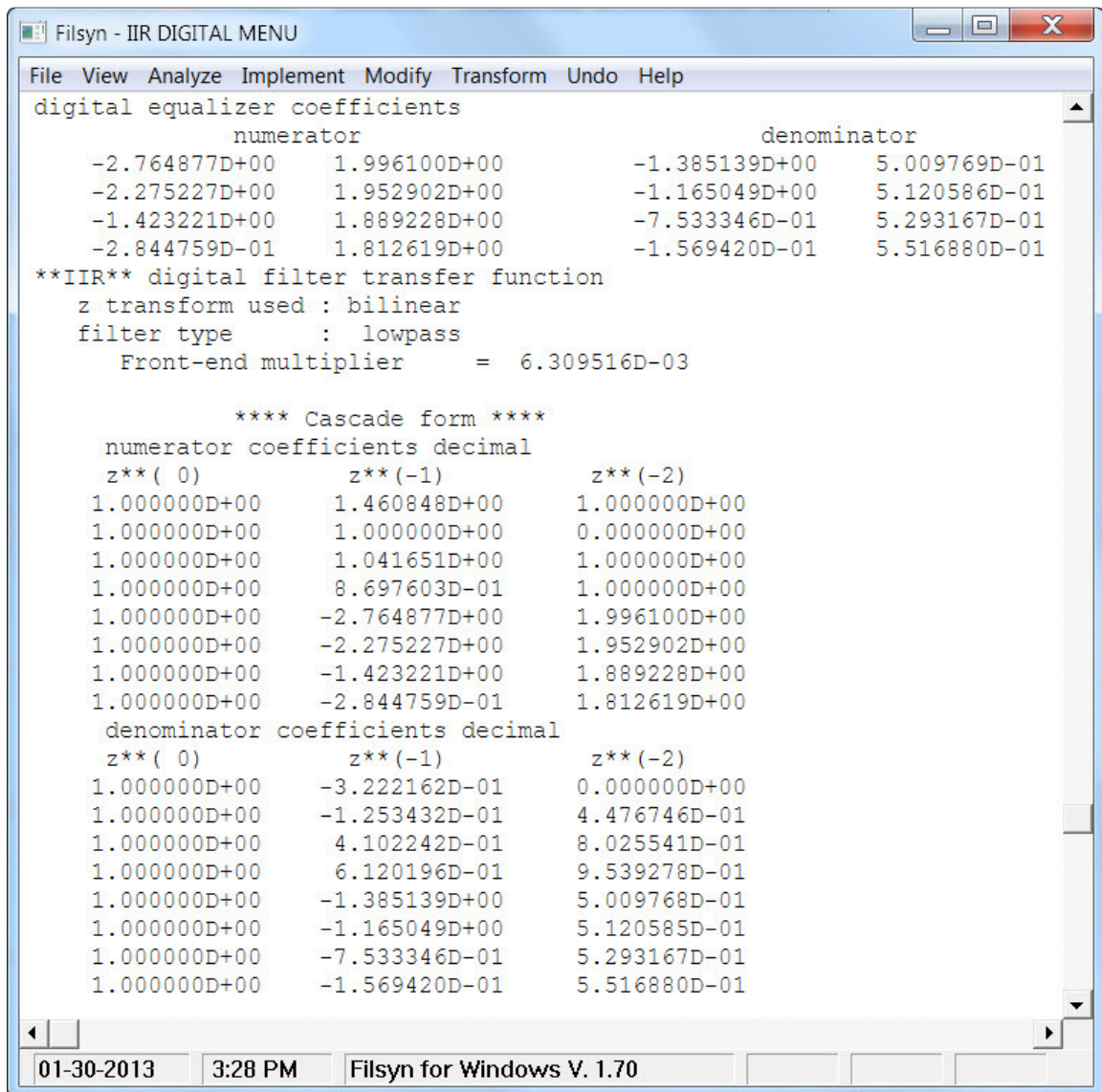
Having analyzed this filter we can perform a delay equalization step from 0 to 2.8KHz, using 4 second order sections:

The dialog box titled "<New Digital Filter>" contains the following settings:

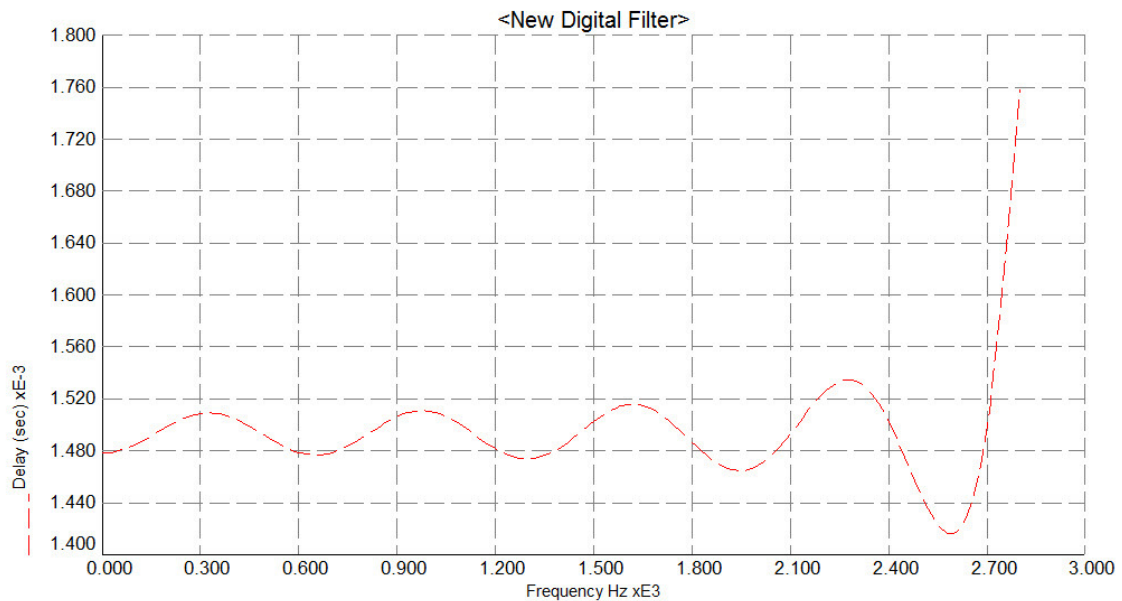
- Number of 1st order sections: 0
- Number of 2nd order sections: 4
- Lower approx. band edge (Hz): 0.00000
- Upper approx. band edge (Hz): 2.8K
- Stepsize: 0.00000

Buttons for OK and Cancel are at the bottom.

Incorporating these delay sections into the lowpass, we get the final form of the filter, and an analysis yields the overall performance shown below.



The passband details of the equalized delay are shown next:



As the last step to get the equalized highpass, we can either add the delay equalizer sections (the last four quadratic factors in both the numerator and denominator above, changing the signs of the linear terms) to the original highpass, or simply enter all the above singularities into the **Analysis->Digital** segment, again changing the signs of all linear coefficients. This is done as follows. First we use the **File->Pole-zero** menu item to write a text file that contains all the poles and zeros of this equalized lowpass filter. Next we edit this file to change the signs of all the real parts from positive to negative and vice versa. Finally we call the **Filsyn** program again and use the **Analysis->Filter->New** menu and specify a digital highpass filter with 10 KHz sampling frequency and pole-zero data entry:

Entering filter data

Filter kind

- ☐ Passive LC
- ☐ Microwave
- ☐ Active RC
- ☐ Switched capacitor
- ☒ IIR digital

Filter type

- ☐ Lowpass
- ☒ Highpass
- ☐ Bandpass
- ☐ Band reject
- ☐ Delay equalizer

Structure

- ☒ Cascade
- ☐ Follow-the-leader
- ☐ Leapfrog

Form

- ☒ Roots
- ☐ Factors

Sampling/Q.W. freq (Hz)

10.000000E+03

Lower passband freq (Hz)

0.00000

(Upper) passband/norm. (Hz)

2.000000E+03

Poles Zeros

Numerator Denominator

Feedback coeff.

No. of zeros at infinity

0

☐ Save analysis data

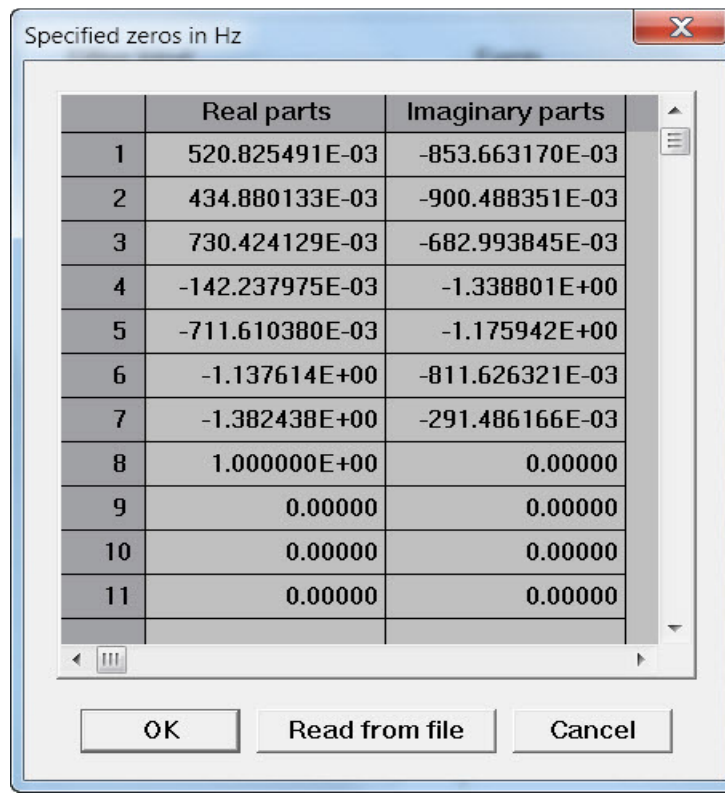
OK Cancel

Clicking now on the **Poles** button, we can direct the program to read the data in from the file we have just modified, yielding:

Specified poles (normalized)

	Real parts	Imaginary parts
1	62.671618E-03	666.143258E-03
2	692.569651E-03	-146.027820E-03
3	582.524693E-03	-415.600178E-03
4	376.667292E-03	-622.445579E-03
5	78.470985E-03	-738.600241E-03
6	-205.112114E-03	-872.056817E-03
7	-306.009819E-03	-927.515934E-03
8	322.216166E-03	0.00000
9	0.00000	0.00000
10	0.00000	0.00000
11	0.00000	0.00000

OK Read from file Cancel



When we then click on the **Zeros** button and select the **Read from file** option, we do not need to specify the file, the program knows to use the same file. The resulting filter is now:

The screenshot shows the 'Filsyn - IIR DIGITAL MENU' window. The menu bar includes File, View, Analyze, Implement, Modify, Transform, Undo, and Help. The main text area displays the following information:

```

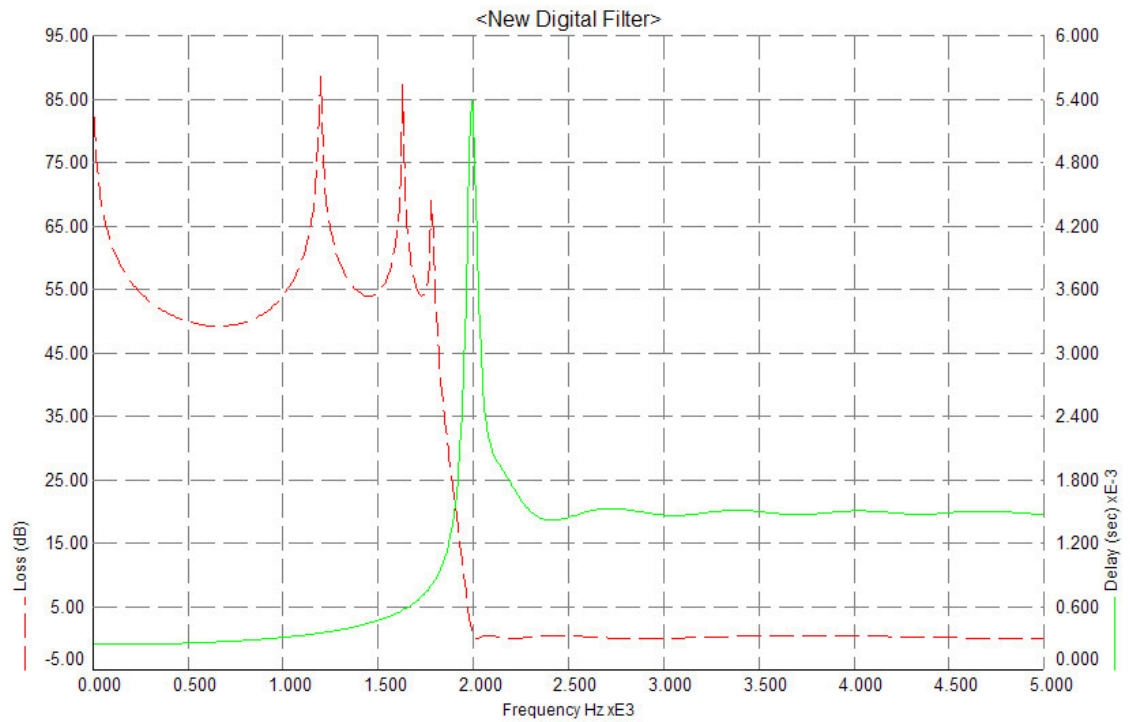
**IIR** digital filter transfer function
z transform used : bilinear
filter type      : lowpass
Front-end multiplier = 6.309516D-03

**** Cascade form ****
numerator coefficients decimal
z**( 0)      z**(-1)      z**(-2)
1.000000D+00  1.460848D+00  1.000000D+00
1.000000D+00  1.000000D+00  0.000000D+00
1.000000D+00  1.041651D+00  1.000000D+00
1.000000D+00  8.697603D-01  1.000000D+00
1.000000D+00 -2.764877D+00  1.996100D+00
1.000000D+00 -2.275227D+00  1.952902D+00
1.000000D+00 -1.423221D+00  1.889228D+00
1.000000D+00 -2.844759D-01  1.812619D+00
denominator coefficients decimal
z**( 0)      z**(-1)      z**(-2)
1.000000D+00 -3.222162D-01  0.000000D+00
1.000000D+00 -1.253432D-01  4.476746D-01
1.000000D+00  4.102242D-01  8.025541D-01
1.000000D+00  6.120196D-01  9.539278D-01
1.000000D+00 -1.385139D+00  5.009768D-01
1.000000D+00 -1.165049D+00  5.120585D-01
1.000000D+00 -7.533346D-01  5.293167D-01
1.000000D+00 -1.569420D-01  5.516880D-01

```

The status bar at the bottom shows the date 01-30-2013, time 4:03 PM, and version Filsyn for Windows V. 1.70.

The only step we need to perform now is to use the **Modify->Scale** menu item, which will change the front-end multiplier to adjust the flat loss in the passband to be zero. The subsequent analysis shows the correct results and the delay details show up as exactly the inverted for of the lowpass on page 476 above:



3. Microwave filter case

The case of microwave highpass filters is a bit different. First, we must realize, that while mathematically, digital and microwave filters behave identically, their physical and behavioral properties are quite different. In the case of digital filters, if the input contains a signal with frequency f that is greater than $f_s/2$, the output will contain a signal of frequency $(f_s - f)$ instead (the *aliasing*). In the microwave case, however, an input signal with frequency above the quarter-wave frequency will be simply transmitted to the output and will appear there without any transposition. Consequently, a microwave highpass filter is, in fact, a bandpass one with a passband centered at the quarter-wave frequency, and theoretically, additional identical passbands will appear at every odd multiples of the quarter-wave frequency.

Nevertheless, the same basic idea can be used to equalize these microwave bandpass filters. However, the operation of shifting the highpass to a lowpass is equivalent here to replacing the transformed singularities by their *inverses*. Hence, if we are given the highpass transfer function in terms of quadratic factors, we must replace a quadratic factor:

$$(S^2 + pS + q)$$

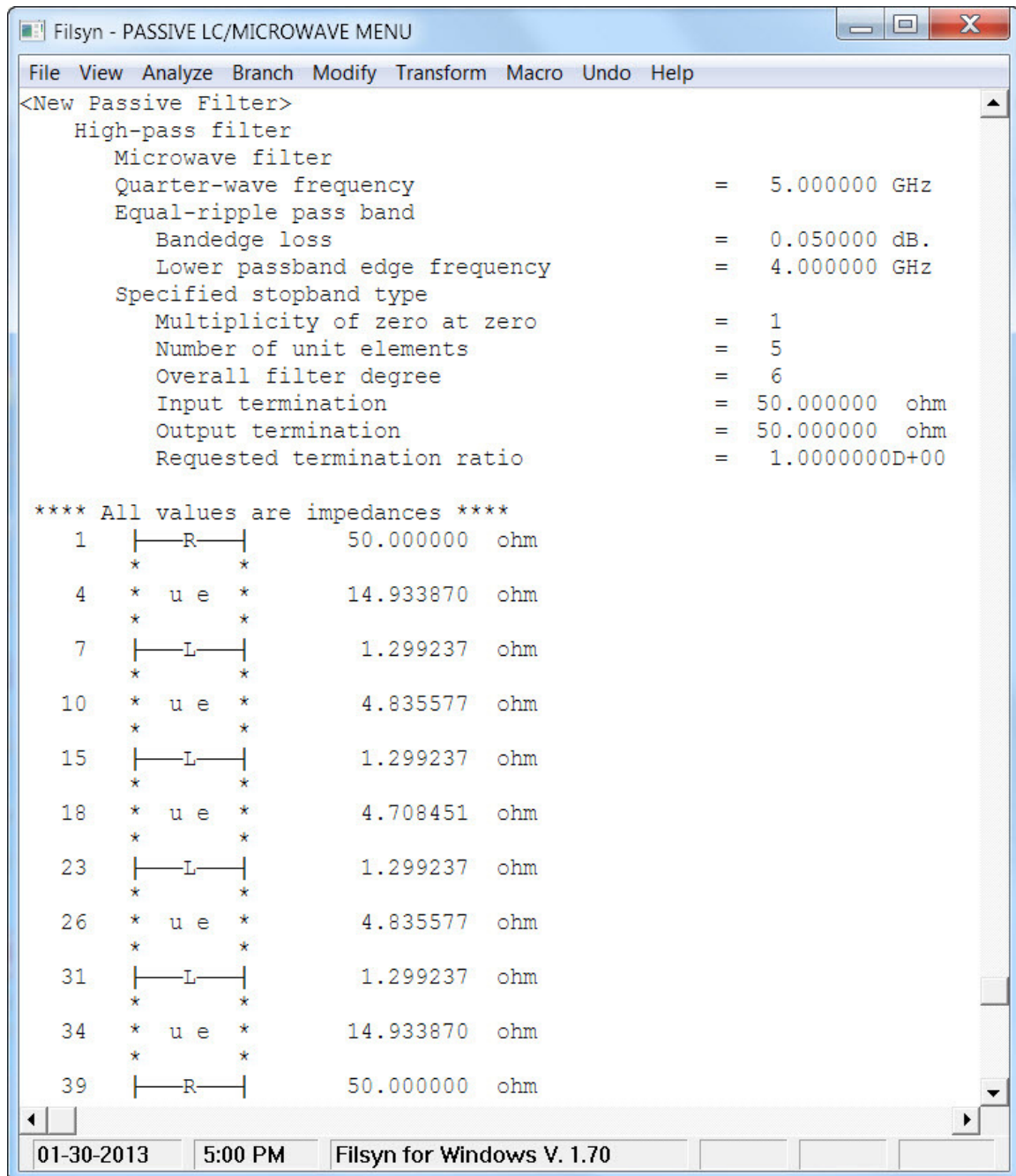
by the factor:

$$(S^2 + (p/q)S + 1/q)$$

Consequently, the conversion is not quite as simple as in the digital case. Furthermore, when we have the equalizer and it is reconverted to the highpass form, one cannot simply append their singularities to the filter transfer function as we did in the digital case.

Nevertheless, the equalizer can be readily designed, but as a separate network. The procedure is again best described by an example.

For simplicity, let us design a simple highpass filter containing 5 unit elements and one zero at zero frequency only, with a 5 GHz quarter-wave frequency and a passband starting at 4 GHz (i.e. a bandpass from 4 GHz to 6 GHz). The resulting shunt shorted stub is subsequently redistributed as identical stubs along the string of unit elements. This yields a structure with convenient element values:



We wish to delay equalize this structure, and towards that objective, design the corresponding lowpass version as follows:

<New Filter>

Filter kind

- ☐ LC
- ☒ Microwave
- ☐ Active RC
- ☐ IIR Digital

Quarter-wave/Sampling freq (Hz)

5.000000E+09

Filter type

- ☒ Low pass
- ☐ High pass
- ☐ Band pass

Lower passband freq (Hz)

0.00000

Upper passband freq (Hz)

1.000000E+09

Passband type

- ☐ Max. flat
- ☒ Equal ripple
- ☐ Functional
- ☐ Sloping

Band-edge loss/return loss (dB)

0.500000000E-01

Loss Slope (dB/oct)

6.00000000

Flat loss (dB)

0.00000000

Function Type

- ☒ E
- ☐ F

Multiplier

0.00000000

Stopband

- ☐ Monotonic
- ☐ Equal min
- ☐ Placer
- ☒ Specified

Lower stopband freq (Hz)

0.00000

Loss (dB)

50.0000000

Upper stopband freq (Hz)

0.00000

Loss (dB)

50.0000000

of zeros at zero

0

of zeros at FQ or FS/2

1

of unit elements

5

Parametric

- ☒ Conventional
- ☐ Parametric
- ☐ Matching

R1

50.000000E+00

R2

50.01

ZS parameter

-1

Q for predistortion

0.00000000

Predistortion

- ☒ Passband only
- ☐ Pass/stop bands

Shifted bandpass trans. Center frequency (Hz)

0.00000

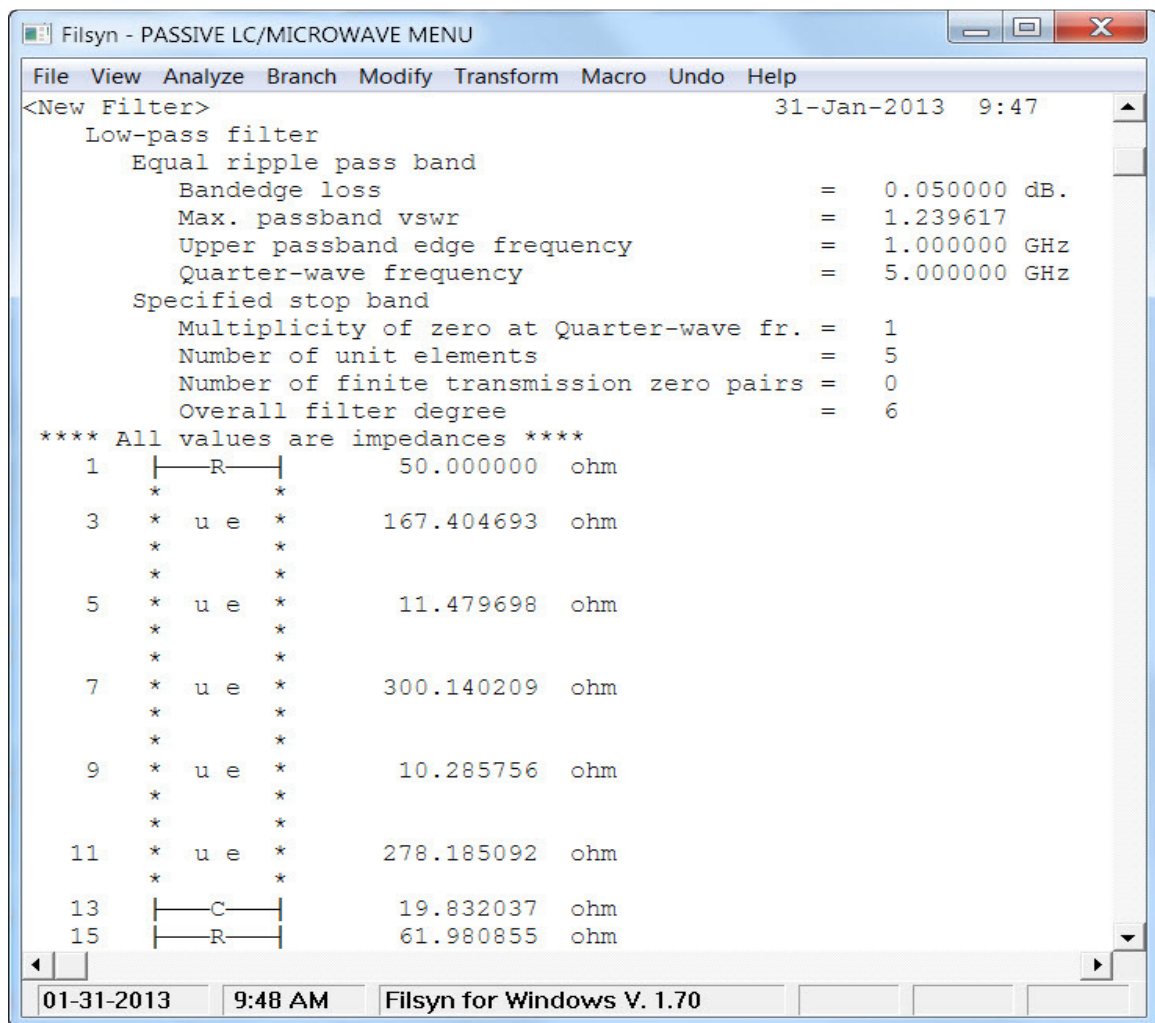
☐ Odd parametric

☐ Save design data

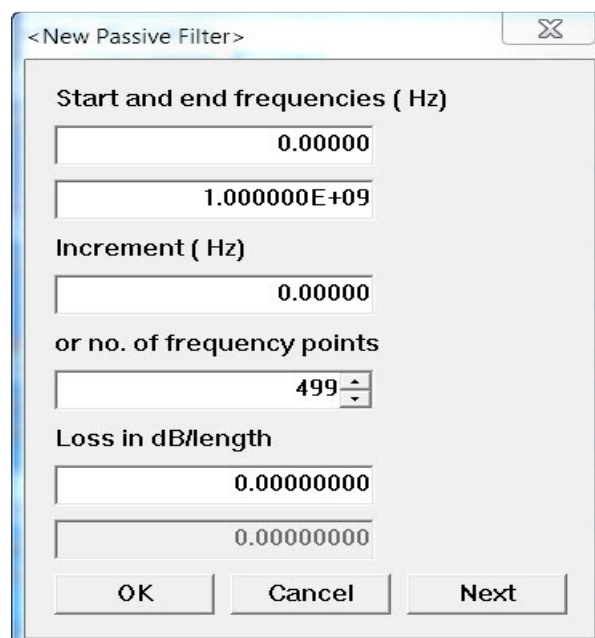
OK Cancel

Note that we had to specify an output termination slightly different from the input and a ZS parameter of -1 (or 0) in order to obtain the class B solution (see *Appendix D* for details).

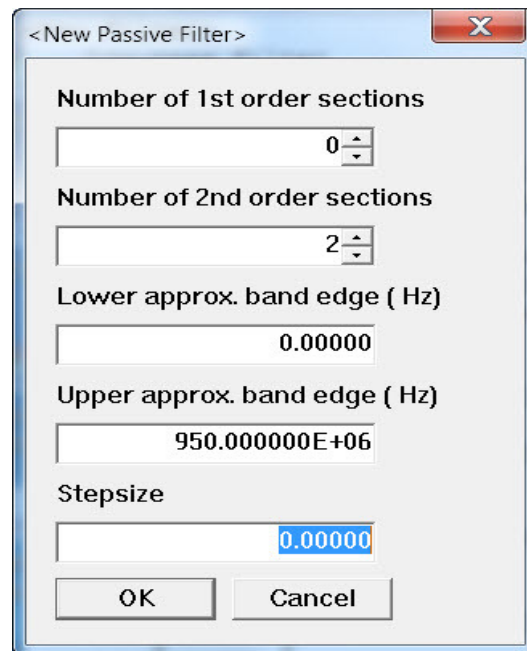
By printing out the poles and zeros of both of these filters we can easily check that the singularities of this lowpass are exactly the inverses of the singularities of the corresponding highpass above. Next we synthesize the filter but without doing any modifications:



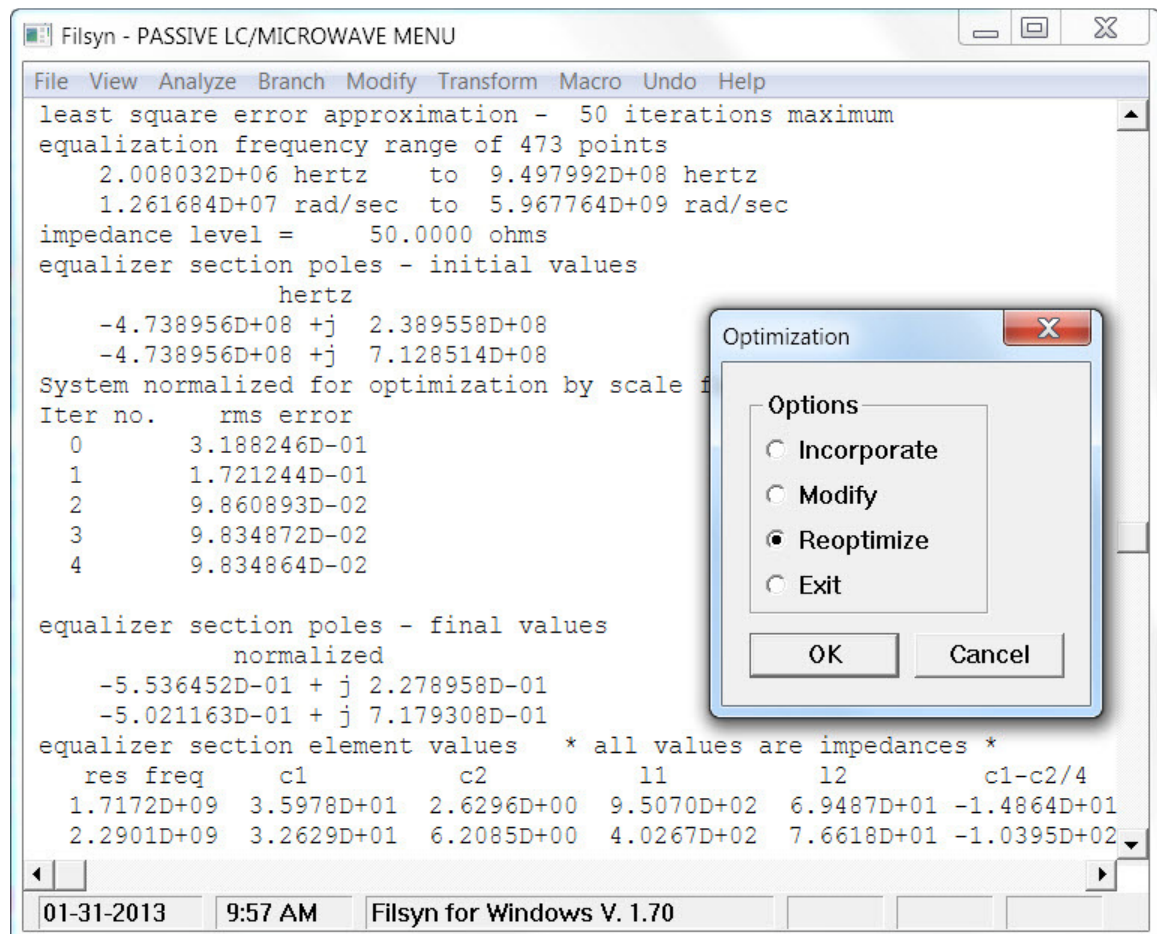
We do a frequency domain analysis in the passband only:



Finally we do a delay equalization step using two, second order sections:



and stop when the natural modes are shown:



This is all we need. These poles must now be inverted and the corresponding delay sections constructed. The inversion is simple and yields the roots:

$$\begin{aligned} &-1.59481889 + j 0.70642022 \\ &-0.63814863 + j 0.95929275 \end{aligned}$$

Finally, the implementation of these equalizer sections is done as follows using the **Analysis->Filter->New** menu option and selecting the **Delay equalizer** radio button:

The screenshot shows the 'Entering filter data' dialog box with the following settings:

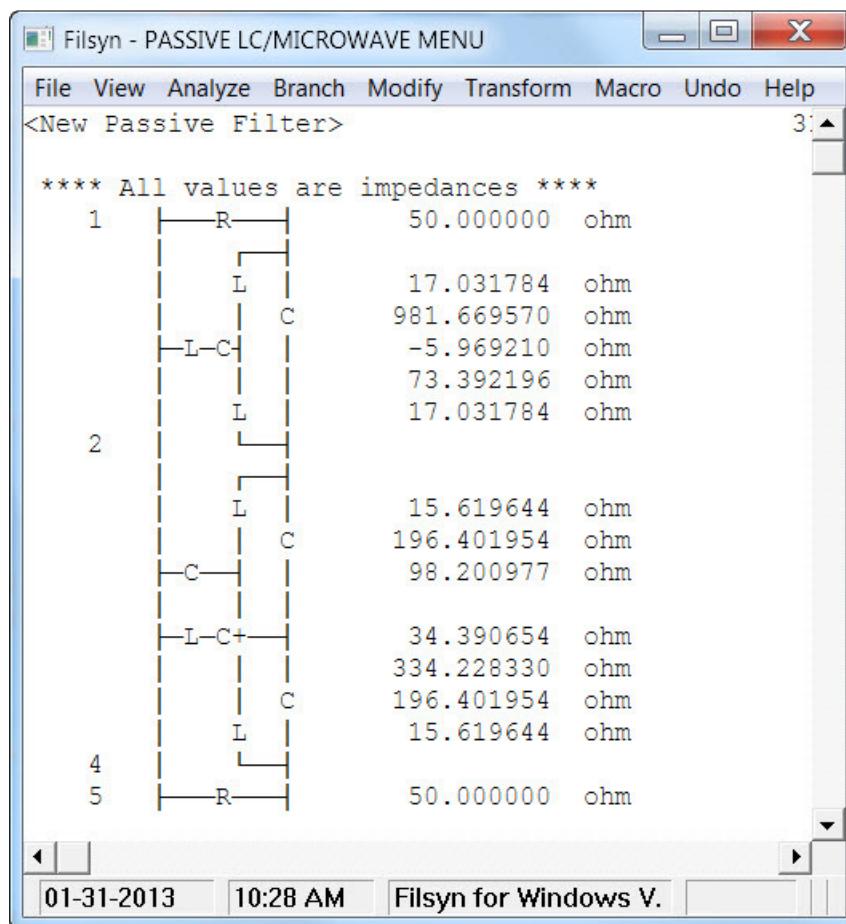
- Filter kind:** Microwave (selected)
- Form:** Roots (selected)
- Sampling/Q.W. freq (Hz):** 10.000000E+09
- Lower passband freq (Hz):** 0.00000
- (Upper) passband/norm. (Hz):** 4.000000E+09
- Filter type:** Delay equalizer (selected)
- Structure:** Cascade (selected)
- No. of zeros at infinity:** 0
- Save analysis data:** (unchecked)

The 'Specified poles (normalized)' sub-dialog is open, showing a table of 11 poles:

	Real parts	Imaginary parts
1	-1.594819E+00	706.420220E-03
2	-638.148630E-03	959.292750E-03
3	0.00000	0.00000
4	0.00000	0.00000
5	0.00000	0.00000
6	0.00000	0.00000
7	0.00000	0.00000
8	0.00000	0.00000
9	0.00000	0.00000
10	0.00000	0.00000
11	0.00000	0.00000

The status bar at the bottom shows: 01-31-2013 10:31 AM Filsyn for Windows V. 1.70

The normalization frequency requested by the program above is the edge of the passband. Clicking on the **Poles** button the poles can then be entered manually as shown above. The resulting equalizer sections follow:



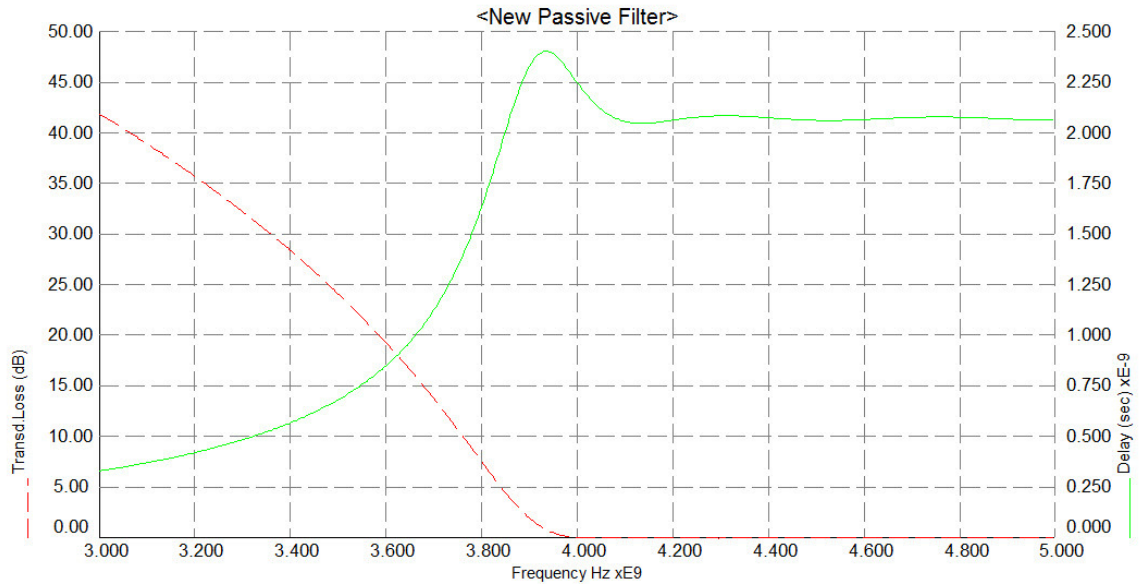
This structure cannot be entered into the highpass filter above, but we can certainly enter instead the filter itself into this structure using the **Branch->Insert** menu item yielding:

The screenshot shows the Filsyn - PASSIVE LC/MICROWAVE MENU software interface. The main window displays a circuit schematic with components numbered 1 through 25. The components are arranged in a ladder-like structure. The values and units for each component are listed in the table below.

Component Number	Symbol	Value	Unit
1	R	50.000000	ohm
2	L	17.031784	ohm
3	C	981.669570	ohm
4	L-C	-5.969210	ohm
5	L	73.392196	ohm
6	L	17.031784	ohm
7	L	15.619644	ohm
8	C	196.401954	ohm
9	C	98.200977	ohm
10	L-C+	34.390654	ohm
11	C	334.228330	ohm
12	L	196.401954	ohm
13	L	15.619644	ohm
14	*		
15	* u e *	14.933870	ohm
16	*		
17	L	1.299237	ohm
18	*		
19	* u e *	4.835577	ohm
20	*		
21	L	1.299237	ohm
22	*		
23	* u e *	4.708451	ohm
24	*		
25	L	1.299237	ohm
26	*		
27	* u e *	4.835577	ohm
28	*		
29	L	1.299237	ohm
30	*		
31	* u e *	14.933870	ohm
32	*		
33	R	50.000000	ohm

The bottom status bar shows the date 01-31-2013, time 10:45 AM, and version Filsyn for Windows V. 1.70.

The analysis of this filter, shown below, yields perfect agreement with our expectations.



Computed performance of delay-equalized microwave highpass

Comment: Since completing this *Application Note* we have added a new submenu item **Convert** to the **Transform** menu item in the IIR digital analysis segment. This menu item, that requires no other data, performs the conversion of a lowpass filter to a highpass and vice versa. This makes the first example of this note much simpler. When we have the highpass to equalize we simply call this new submenu, that converts it to a lowpass that can be equalized and the result converted back to a highpass and we are done.

APPLICATION NOTE 7

LINEAR-PHASE BANDPASS FILTER

1. Introduction

Filsyn is capable of designing linear-phase lowpass as well as bandpass filters directly. However, the bandpass case only yields accurate results for small and moderate bandwidths (bandwidths of about 25% or less).

As an example, consider the following requirements. We need an equal ripple delay of 0.1 msec in a 20 KHz wide band, centered at 200 KHz (10% bandwidth). The bandwidth-delay product is 4π , or about 12.6, hence we estimate the necessary degree to be about 12.

2. Design example

Since the bandpass case is derived from a lowpass by shifting the band linearly, we start with an equivalent linear-phase lowpass design with the same 0.1 msec delay over a band from 0 to 10 KHz (half the bandwidth), and specifying a 6th order design. The delay-bandwidth product is about 6.28 and from equation 10 in *Appendix G* we see, that a 6th order section will yield about a 2%, or a 2 μ sec ripple. Hence the data input session goes as follows, starting with the **Filsyn** program and selecting the **Design->Delay line->New** menu option:

<New Delay-Filter/-Line/-Equalizer>

Delay type

☐ Max. flat

☒ Equal ripple

☐ Equalizer

Implementation

☒ LC elements

☐ Active RC

☐ Microwave

☐ IIR digital

Data

FQ or FS (Hz)

0.000000

Normalization freq (Hz)

0.000000

Network type

☒ Filter

☐ Delay line

Delay value (sec)

100.000000E-06

Stopband

☐ Equal min.

☒ Specified

End of passband (Hz)

10.000000E+03

Start of stopband (Hz)

0.000000

Degree

0

Zeros at infinity

6

Transm. zeros

Terminations

50.000000E+00

50.000000E+00

Q-value

0.00000000

Hurwitz type

☒ Hurwitz

☐ Non Hurwitz

Selection

☒ Auto

☐ Manual

Indicators

Shifted bandpass trans.

Center frequency (Hz)

200K

☐ Odd parametric

☐ Save design data

OK **Cancel**

We specified equal ripple delay; which requires an iterative approximation that stops in this case after two iterations, but we request two more and then stop:

Filsyn - MAIN MENU

File Design Analysis Help

*** Filsyn *** Filter Program

<New Delay-Filter/-Line/-Equalizer> 31-Jan-2013 15:43

Linear phase low-pass filter

Equal ripple delay passband

Low frequency delay = 100.000000 usec

Upper passband edge frequency = 10.000000 KHz

at iteration no. 2 delay ripple: 1.8022D-06 rms error: 4.5840D-04

at iteration no. 2 delay ripple: 1.8022D-06 rms error: 5.5316D-12

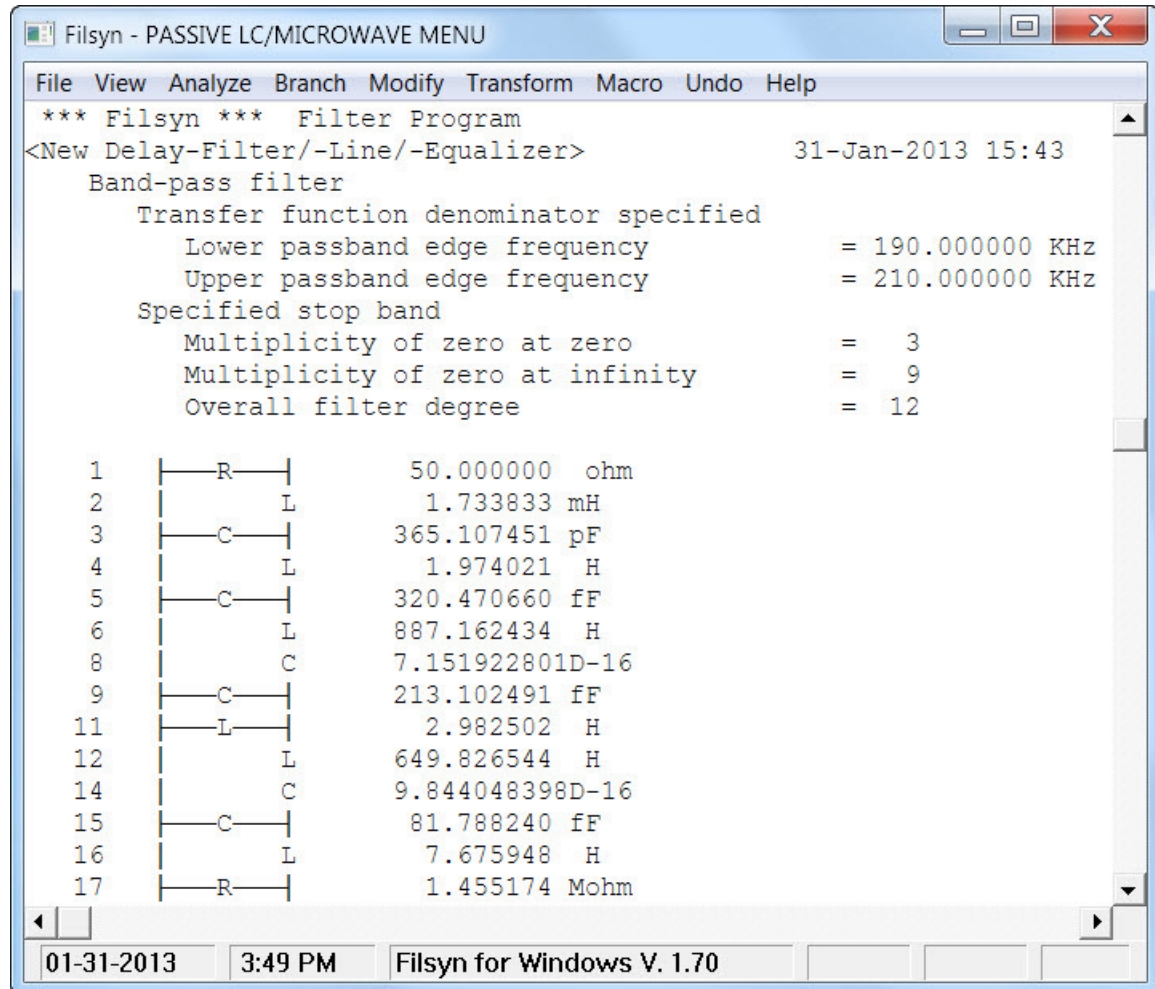
Question

More iterations?

Yes **No**

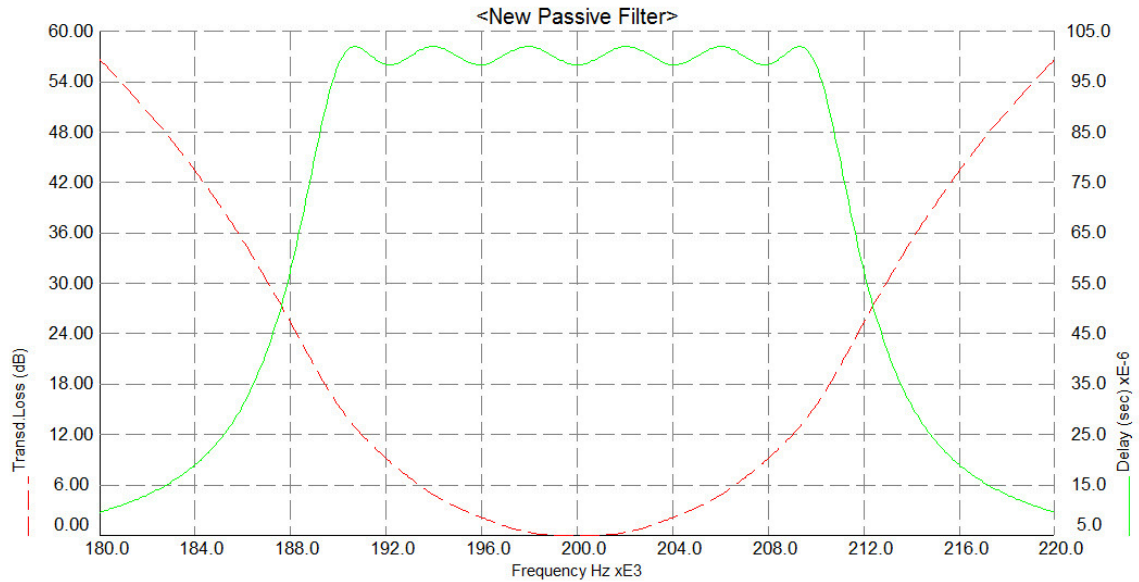
An rms error of 4.58D-04 would have already been quite satisfactory and the delay error is $\pm 1.8 \mu\text{sec}$, close to what we have predicted.

Since we have specified a center frequency of 200 KHz, the program automatically converts this lowpass into a bandpass centered at the indicated 200 KHz yielding the circuit:



One item requires some explanation. The 6th order transmission zero of the lowpass filter at infinity is not converted to the same order zero of the bandpass and zero and infinite frequencies. Instead, the 12 zeros are divided into three at zero and 9 at infinity. This is because making $n_{\infty} \approx 3n_0$ makes the filter loss characteristic as close to arithmetically symmetrical as possible:

Linear-phase bandpass filter



Clearly the element values are not very convenient, but a few applications of the **Modify->Norton->Matching** menu options can yield all equal inductances, for instance:

Filsyn - PASSIVE LC/MICROWAVE MENU				
File View Analyze Branch Modify Transform Macro Undo Help				
1	—R—	50.000000	ohm	
2	—L—	1.733833	mH	
3	—C—	354.095102	pF	
4	—C—	11.348684	pF	
5	—C—	371.579887	pF	
6	—L—	1.733833	mH	
7	—C—	7.734966	nF	
8	—L—	1.733833	mH	
9	—C—	361.002900	pF	
10	—C—	22.219795	pF	
11	—L—	1.733833	mH	
12	—C—	370.760841	pF	
13	—C—	5.101946	nF	
14	—L—	1.733833	mH	
15	—C—	352.620285	pF	
16	—C—	42.997487	pF	
17	—C—	323.764839	pF	
18	—L—	1.733833	mH	
19	—R—	328.692781	ohm	
01-31-2013 4:31 PM Filsyn for Windows V.				

The same filter can, of course, be implemented in active RC form as well, but microwave or IIR digital filters currently may not have linear-phase forms.

In preparation for some additional possibilities, we have saved the pole-zero data of this filter, using the **File->Pole-zero** menu option, which generated the following text file:

```
! Pole-zero data of:
! <New Passive Filter>                                5-oct-2005 11:51
! Poles (in rad/sec):
-2.013533653E+04   1.269666646E+06
-2.013533653E+04   1.243607477E+06
-1.916356063E+04   1.295171363E+06
-1.916356063E+04   1.218102760E+06
-1.530450392E+04   1.318692199E+06
-1.530450392E+04   1.194581924E+06
! Zeros (in Hz):
0.000000000E+00   0.000000000E+00
0.000000000E+00   0.000000000E+00
0.000000000E+00   0.000000000E+00
```

3. Equal-minima design

If we need more discrimination, we may request an equal minima type stopband beginning at, say, 15 KHz:

<New Delay-Filter/-Line/-Equalizer>

Delay type

- ☐ Max. flat
- ☒ Equal ripple
- ☐ Equalizer

Implementation

- ☒ LC elements
- ☐ Active RC
- ☐ Microwave
- ☐ IIR digital

Data

FQ or FS (Hz)

0.00000

Normalization freq (Hz)

0.00000

Network type

- ☒ Filter
- ☐ Delay line

Delay value (sec)

100.000000E-06

Stopband

- ☒ Equal min.
- ☐ Specified

End of passband (Hz)

10.000000E+03

Start of stopband (Hz)

15.000000E+03

Degree

6

Zeros at infinity

0

Transm. zeros

Terminations

50.000000E+00

50.000000E+00

Q-value

0.00000000

Hurwitz type

- ☒ Hurwitz
- ☐ Non Hurwitz

Selection

- ☒ Auto
- ☐ Manual

Indicators

Shifted bandpass trans.

Center frequency (Hz)

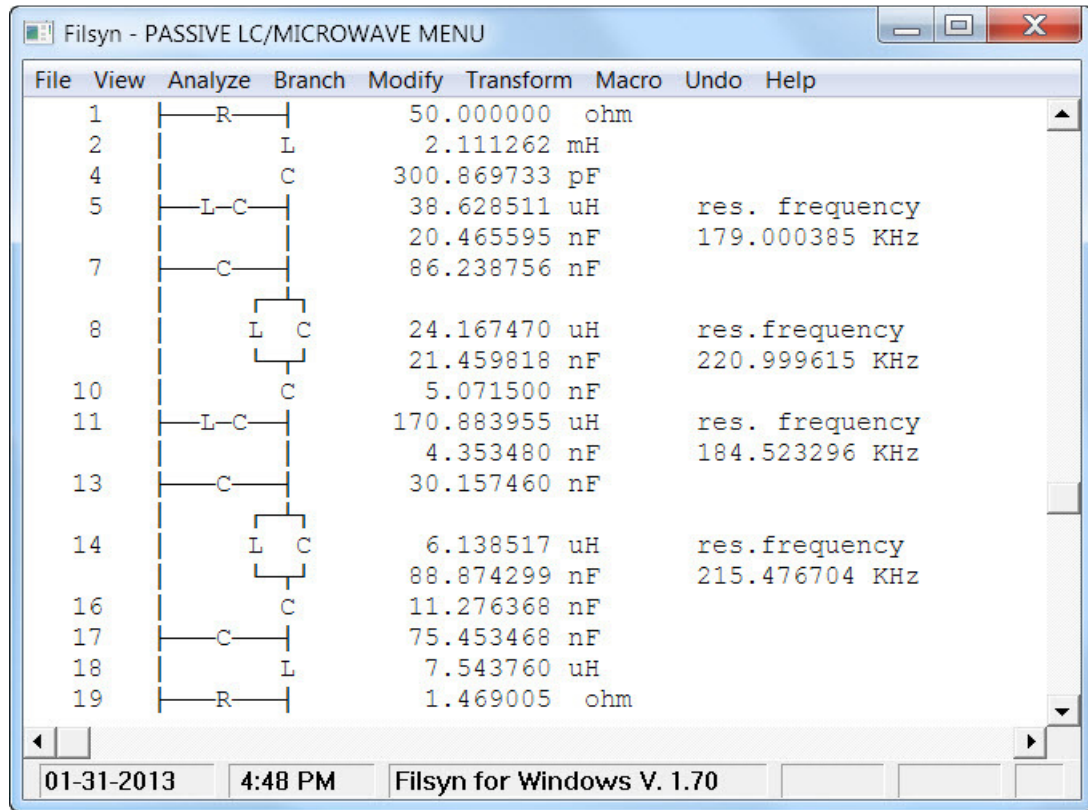
200K

☐ Odd parametric

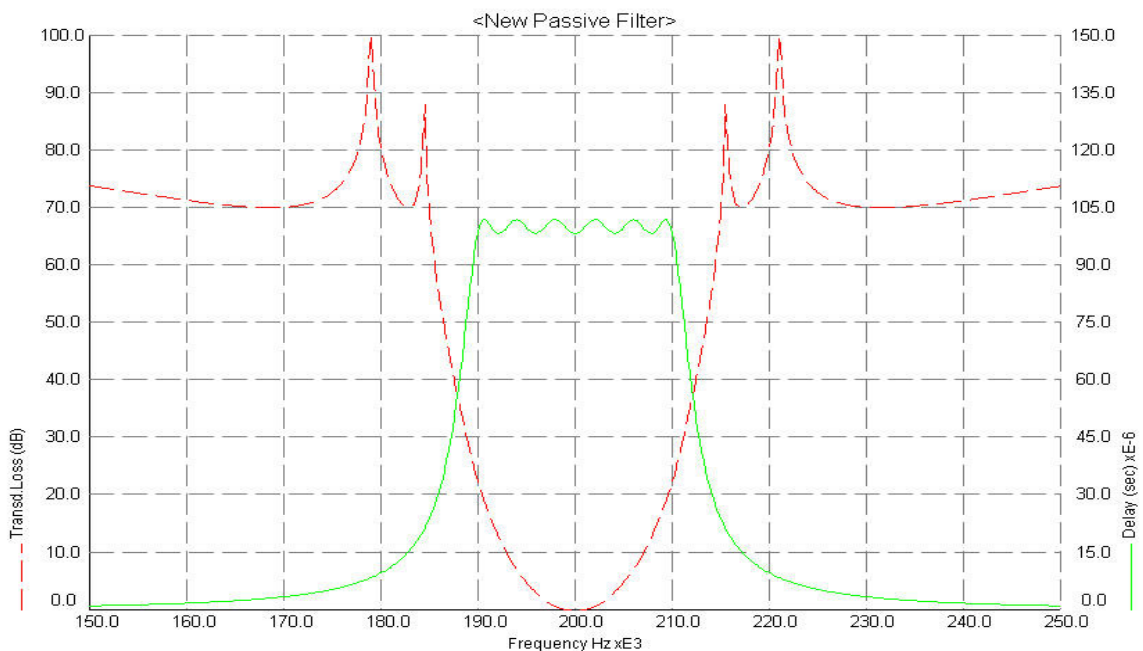
☐ Save design data

OK Cancel

resulting in a circuit shown below. The element values could also be adjusted here, but we leave that to the reader. Note also that the extreme zeros were again distributed in a 1:3 fashion.



The frequency domain behavior is quite impressive also:



One more comment is in order here. For passive LC filter implementation, please remember to specify an even degree lowpass. If you specify an odd degree (with the equal-minima type stopband), the resulting filter structure will contain more inductors and it will not be of the zig-zag type, with the result that no impedance transformation can be applied and therefore no element value adjustments are available. For instance, specifying degree 5, will yield a structure containing 7 inductors instead of the 6 above for degree 6. One could avoid this situation by selecting the Odd parametric option offered in the data entry screen above. This adds a degree to the filter making it of a parametric type, without affecting the performance in any noticeable way. The resulting filter has a zig-zag form, hence has fewer inductors and will be amenable to element value adjustments.

4. Specified stopband

If you wish to exercise more control over the stopband behavior of the filter, the following procedure makes that possible. We shall use the main data entry window **Design->Filter->New** and **Functional input**, using the pole-zero data we have obtained before. However, we shall only use the poles, specifying the zeros manually:

The screenshot shows the 'Filter Parameters' dialog box with two tables for pole-zero data. The left table, 'Function zeros (in rad/sec)', contains 11 rows of data. The right table, 'Transmission zeros (in Hz)', contains 11 rows of data, with the value '0.00000' highlighted in the Real Part column for row 5. At the bottom are buttons for 'OK', 'Read from file', and 'Cancel'.

Function zeros (in rad/sec)			Transmission zeros (in Hz)		
	Real Part	Imaginary Part		Real Part	Imaginary Part
1	-20.135337E+03	1.269667E+06	1	0.00000	170.000000E+03
2	-20.135337E+03	1.243607E+06	2	0.00000	180.000000E+03
3	-19.163561E+03	1.295171E+06	3	0.00000	220.000000E+03
4	-19.163561E+03	1.218103E+06	4	0.00000	230.000000E+03
5	-15.304504E+03	1.318692E+06	5	0.00000	0.00000
6	-15.304504E+03	1.194582E+06	6	0.00000	0.00000
7	0.00000	0.00000	7	0.00000	0.00000
8	0.00000	0.00000	8	0.00000	0.00000
9	0.00000	0.00000	9	0.00000	0.00000
10	0.00000	0.00000	10	0.00000	0.00000
11	0.00000	0.00000	11	0.00000	0.00000

<New Filter>

Filter kind

- ☒ LC
- ☐ Microwave
- ☐ Active RC
- ☐ IIR Digital

Quarter-wave/Sampling freq (Hz)

0.00000

Filter type

- ☐ Low pass
- ☐ High pass
- ☒ Band pass

Lower passband freq (Hz)

190.000000E+03

Upper passband freq (Hz)

210.000000E+03

Passband type

- ☐ Max. flat
- ☐ Equal ripple
- ☒ Functional
- ☐ Sloping

Function Type

- ☒ E
- ☐ F

Multiplier

0.00000000

Stopband

- ☐ Monotonic
- ☐ Equal min
- ☐ Placer
- ☒ Specified

Lower stopband freq (Hz)

0.00000

Loss (dB)

50.0000000

Band-edge loss/return loss (dB)

0.500000000

Loss Slope (dB/oct)

6.00000000

Flat loss (dB)

0.00000000

Upper stopband freq (Hz)

0.00000

Loss (dB)

50.0000000

Detail Parameters

of zeros at zero

1

of zeros at infinity

3

of unit elements

0

Parametric

- ☒ Conventional
- ☐ Parametric
- ☐ Matching

R1

50.000000E+00

R2

50.000000E+00

ZS parameter

-1

Q for predistortion

0.00000000

Predistortion

- ☒ Passband only
- ☐ Pass/stop bands

Shifted bandpass trans. Center frequency (Hz)

0.00000

☐ Odd parametric

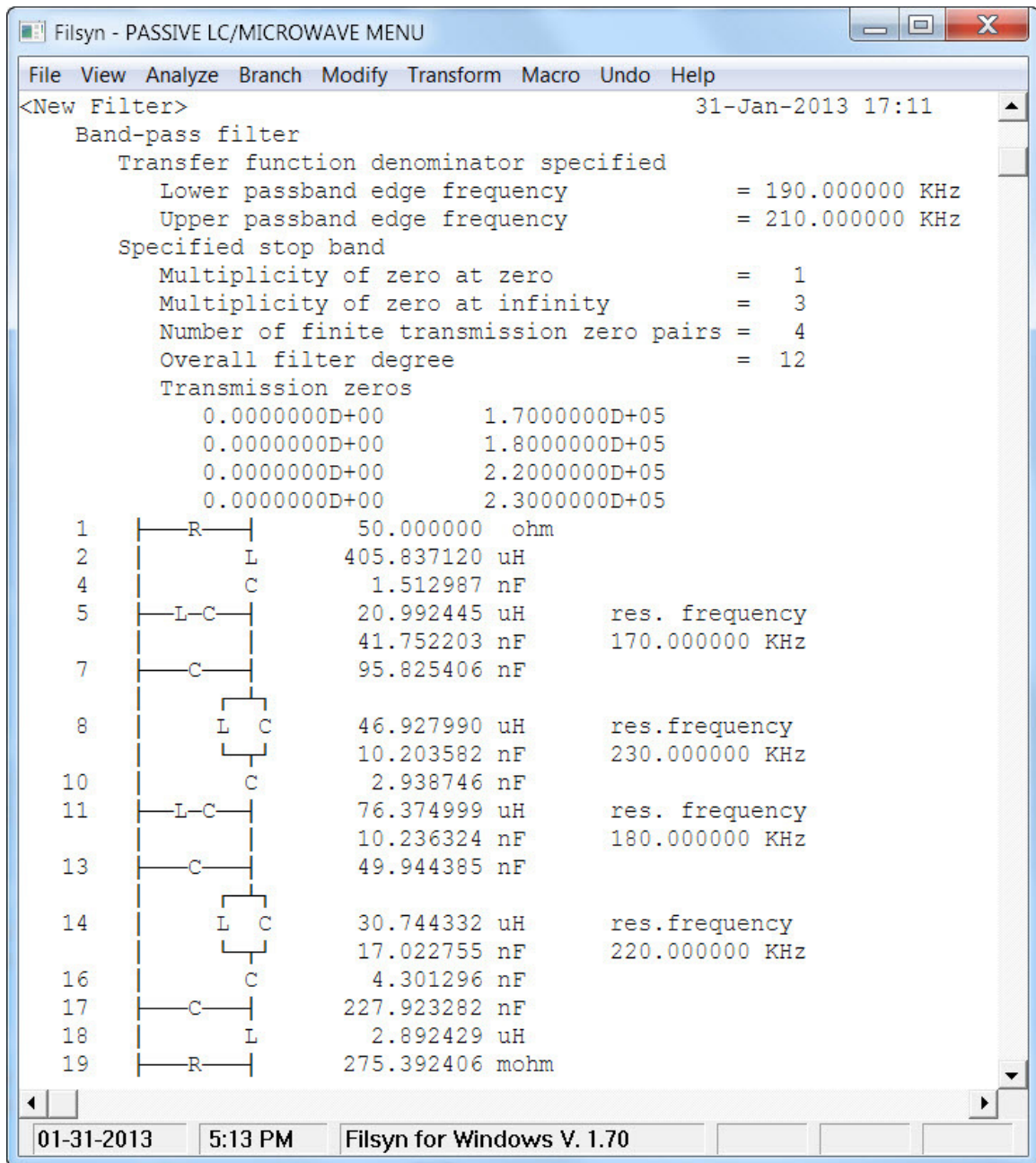
☐ Save design data

OK Cancel

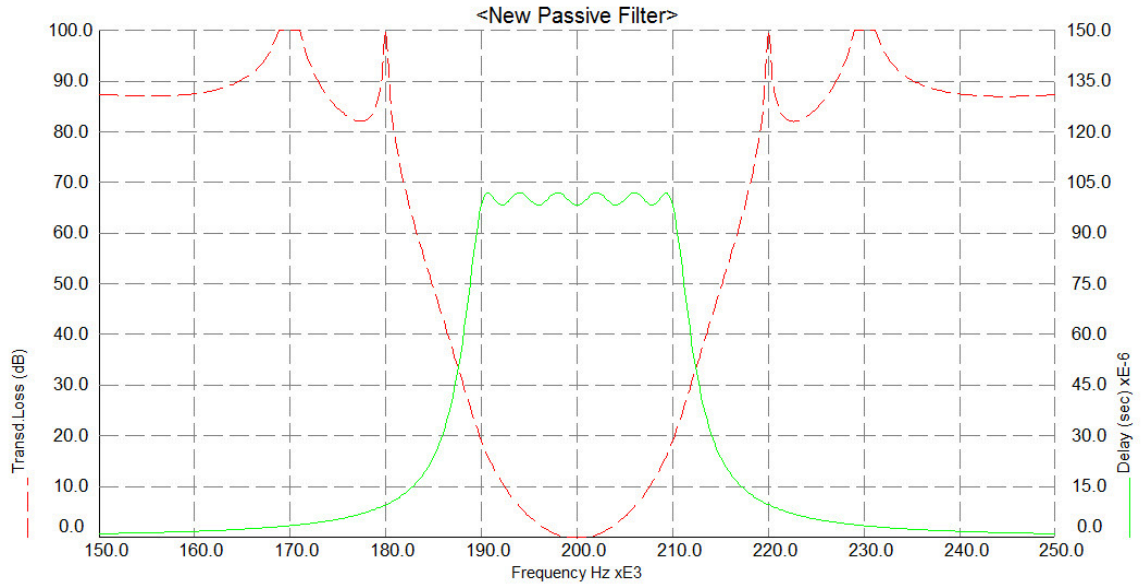
What happened here is when the **Detail parameters** button has been pushed the first time, we clicked on the **Read from file** button and specified the file containing the pole-zero data. Only the **Function zeros** (the poles) were read in at this time. In order to read in, or otherwise specify, the zeros, we have to click on the **Detail parameters** button again. This time we can either click on the **Read from file** button again, or enter the required data manually, as we did here.

The rest of the data is as usual, we specify a bandpass from 190 KHz to 210 KHz, one zero at zero and 3 at infinity and specifying the “E” function.

The resulting passive LC implementation follows uneventfully:



These element values may be modified as necessary, which we shall ignore, we only show the computed performance:



5. 3 dB bandwidth

In the linear-phase design, the normalization frequency is either related to the delay (the maximally flat delay option), or it is the constant delay bandwidth (equal-ripple delay option), as it is in this application note.

Quite often, users are most interested in the 3 dB bandwidth and the actual passband delay is of secondary interest. Since the 3 dB point may *not* be entered as a specification, the following procedure can be used to obtain a required 3 dB bandwidth, but only for filters with no finite transmission zeros.

One starts with the design just as we have done at the beginning of this application note, and use an arbitrary delay value. If we use the 100 μsec delay, of course, our results will be just as shown above. Next we perform a frequency domain analysis to locate the 3 dB (or any other loss) point, and in this specific case we find the 3 dB bandwidth to be 9.446 KHz. A simple rescaling of the natural modes of the lowpass may be used to change this to the required 10 KHz, if we need a 3 dB bandwidth of 20 KHz for the bandpass. The ratio we need is:

$$20 \text{ KHz} / 9.446 \text{ KHz} = 2.1173$$

and this means that the delay of the lowpass as well as the bandpass will *not* be 100 μsec , but instead it will be:

$$100 \mu\text{sec} / 2.1173 = 47.23 \mu\text{sec}$$

Similarly, the constant delay bandwidth will be increased to

$$10 \text{ KHz} * 2.1173 = 21.173 \text{ KHz}$$

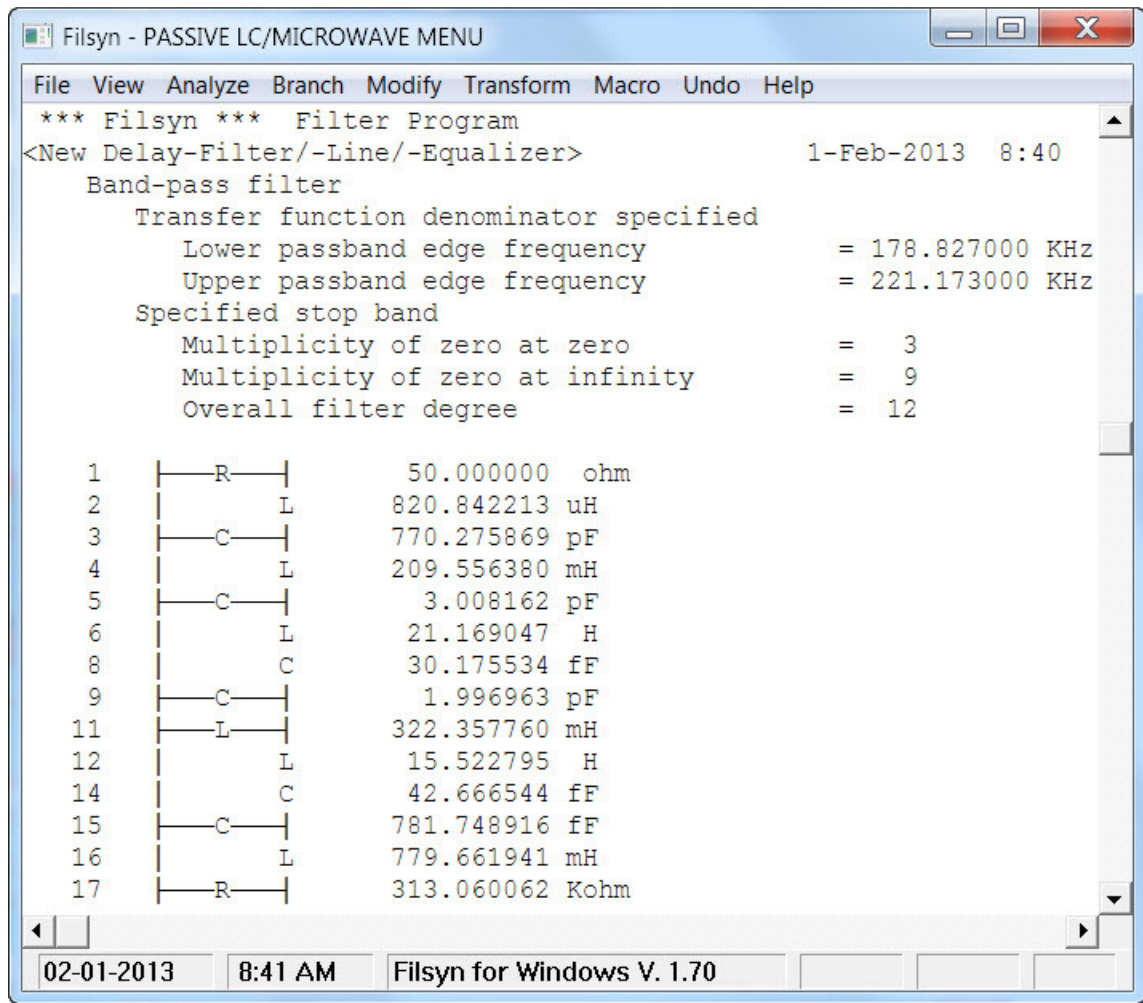
in the lowpass case, and exactly twice this for the bandpass filter.

The design will be as follows:

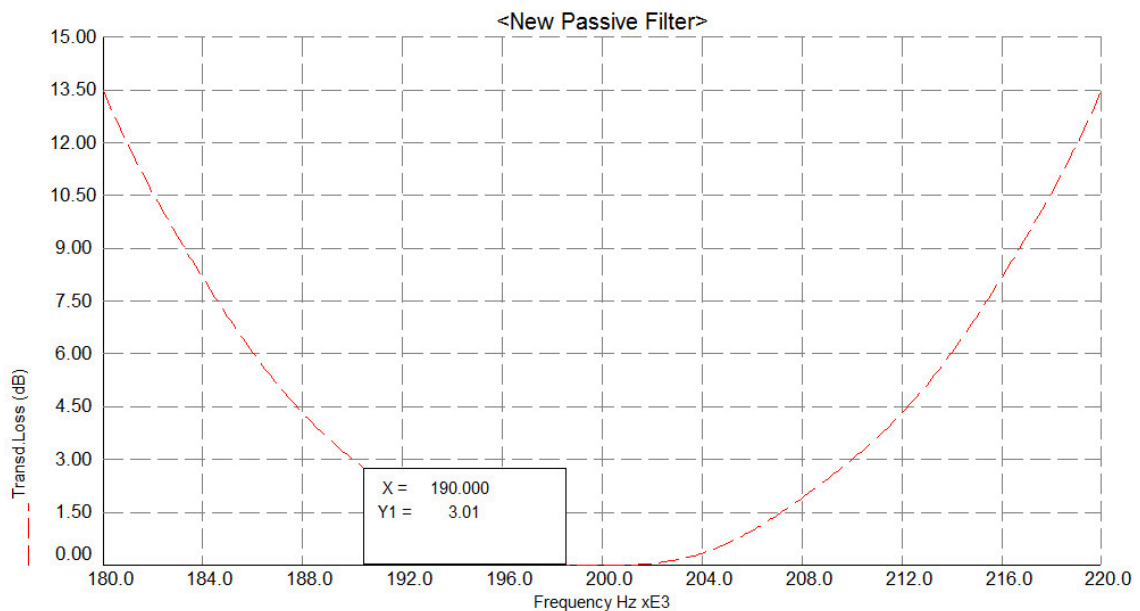
<New Delay-Filter/-Line/-Equalizer> X

<p>Delay type</p> <p><input type="radio"/> Max. flat</p> <p><input checked="" type="radio"/> Equal ripple</p> <p><input type="radio"/> Equalizer</p>	<p>Delay value (sec)</p> <p>47.230000E-06</p>	<p>Q-value</p> <p>0.00000000</p>
<p>Implementation</p> <p><input checked="" type="radio"/> LC elements</p> <p><input type="radio"/> Active RC</p> <p><input type="radio"/> Microwave</p> <p><input type="radio"/> IIR digital</p>	<p>Stopband</p> <p><input type="radio"/> Equal min.</p> <p><input checked="" type="radio"/> Specified</p>	<p>Hurwitz type</p> <p><input checked="" type="radio"/> Hurwitz</p> <p><input type="radio"/> Non Hurwitz</p>
<p>Data</p> <p>FQ or FS (Hz)</p> <p>0.00000</p>	<p>End of passband (Hz)</p> <p>21.173000E+03</p>	<p>Selection</p> <p><input checked="" type="radio"/> Auto</p> <p><input type="radio"/> Manual</p>
<p>Normalization freq (Hz)</p> <p>0.00000</p>	<p>Start of stopband (Hz)</p> <p>0.00000</p>	<p>Indicators</p>
<p>Network type</p> <p><input checked="" type="radio"/> Filter</p> <p><input type="radio"/> Delay line</p>	<p>Degree</p> <p>0</p>	<p>Shifted bandpass trans.</p>
	<p>Zeros at infinity</p> <p>6</p>	<p>Center frequency (Hz)</p> <p>200K</p>
	<p>Transm. zeros</p>	<p><input type="checkbox"/> Odd parametric</p>
	<p>Terminations</p> <p>50.000000E+00</p> <p>50.000000E+00</p>	<p><input type="checkbox"/> Save design data</p>
		<p>OK Cancel</p>

The resulting filter is shown below:



and the loss in the passband indicates that the 3 dB bandwidth is exactly 20 KHz as needed:



As a final comment, we wish to point out that the specification of the 3 dB point combined with equal-minima stopband, or any finite transmission zeros, is a more difficult problem and requires an iterative approach.

APPLICATION NOTE 8

ACTIVE RC FILTER USING BIQUADS

1. Introduction

The design of active RC filters using biquadratic building blocks can be simple, or it may need special discussion, especially if one of the multiple-feedback structures is to be used. This note will illustrate the problems that may occur, and how they may be solved using the **Filsyn** program.

2. Specification

We select a relatively simple bandpass with passband from 1 KHz to 2 KHz and stopband requirements below 750 Hz and above 2400 Hz. To obtain the optimum design, we specified no extreme transmission zeros.

Using the **Placer** option, the data entry screen is as follows:

Filter Parameters

Requirements

	Frequency (Hz)	Loss (dB)
1	750.000000E+00	40.0000000
2	2.400000E+03	35.0000000
3	0.00000	0.00000000
4	0.00000	0.00000000
5	0.00000	0.00000000
6	0.00000	0.00000000
7	0.00000	0.00000000
8	0.00000	0.00000000
9	0.00000	0.00000000
10	0.00000	0.00000000
11	0.00000	0.00000000

Fixed zeros

	Frequency (Hz)	Multiplicity
1	0.00000	0
2	0.00000	0
3	0.00000	0
4	0.00000	0
5	0.00000	0
6	0.00000	0
7	0.00000	0
8	0.00000	0
9	0.00000	0
10	0.00000	0

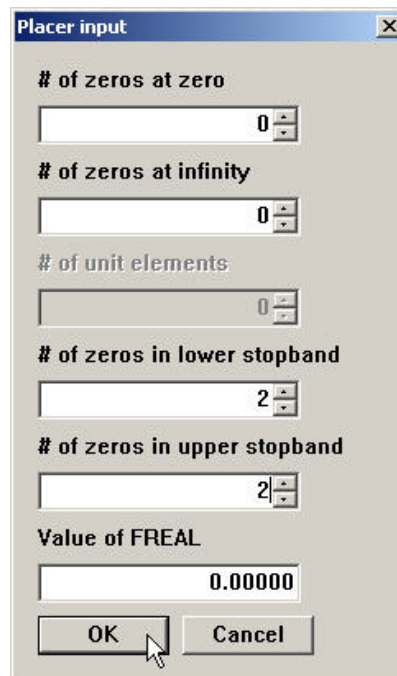
Movable zeros

	Frequency (Hz)	Multiplicity
1	0.00000	0
2	0.00000	0
3	0.00000	0
4	0.00000	0
5	0.00000	0
6	0.00000	0
7	0.00000	0
8	0.00000	0
9	0.00000	0
10	0.00000	0
11	0.00000	0

or no. of zeros below passband

and/or no. of zeros above passband

Our initial estimate of three finite transmission zeros turns out to be inadequate; hence we must increase it:



Placer input

of zeros at zero: 0

of zeros at infinity: 0

of unit elements: 0

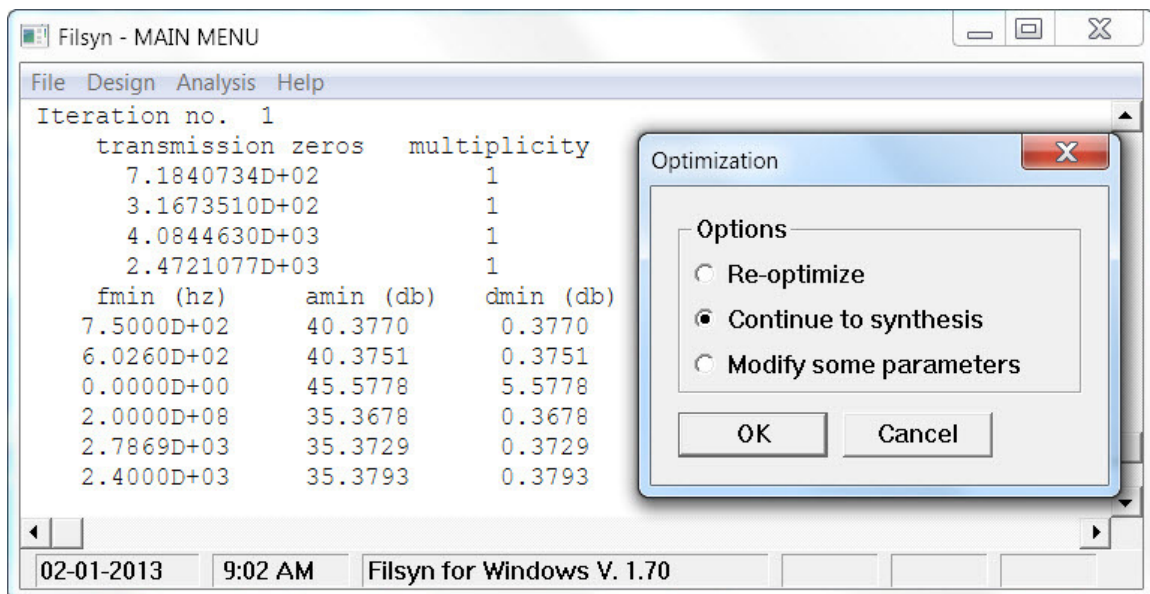
of zeros in lower stopband: 2

of zeros in upper stopband: 2

Value of FREAL: 0.00000

OK Cancel

Convergence is now obtained in three iterations, and forcing one more yields the following printout. This time everything is fine and we get a very satisfactory design:



Filsyn - MAIN MENU

File Design Analysis Help

Iteration no. 1

transmission zeros	multiplicity
7.1840734D+02	1
3.1673510D+02	1
4.0844630D+03	1
2.4721077D+03	1

fmin (hz)	amin (db)	dmin (db)
7.5000D+02	40.3770	0.3770
6.0260D+02	40.3751	0.3751
0.0000D+00	45.5778	5.5778
2.0000D+08	35.3678	0.3678
2.7869D+03	35.3729	0.3729
2.4000D+03	35.3793	0.3793

Optimization

Options

☐ Re-optimize

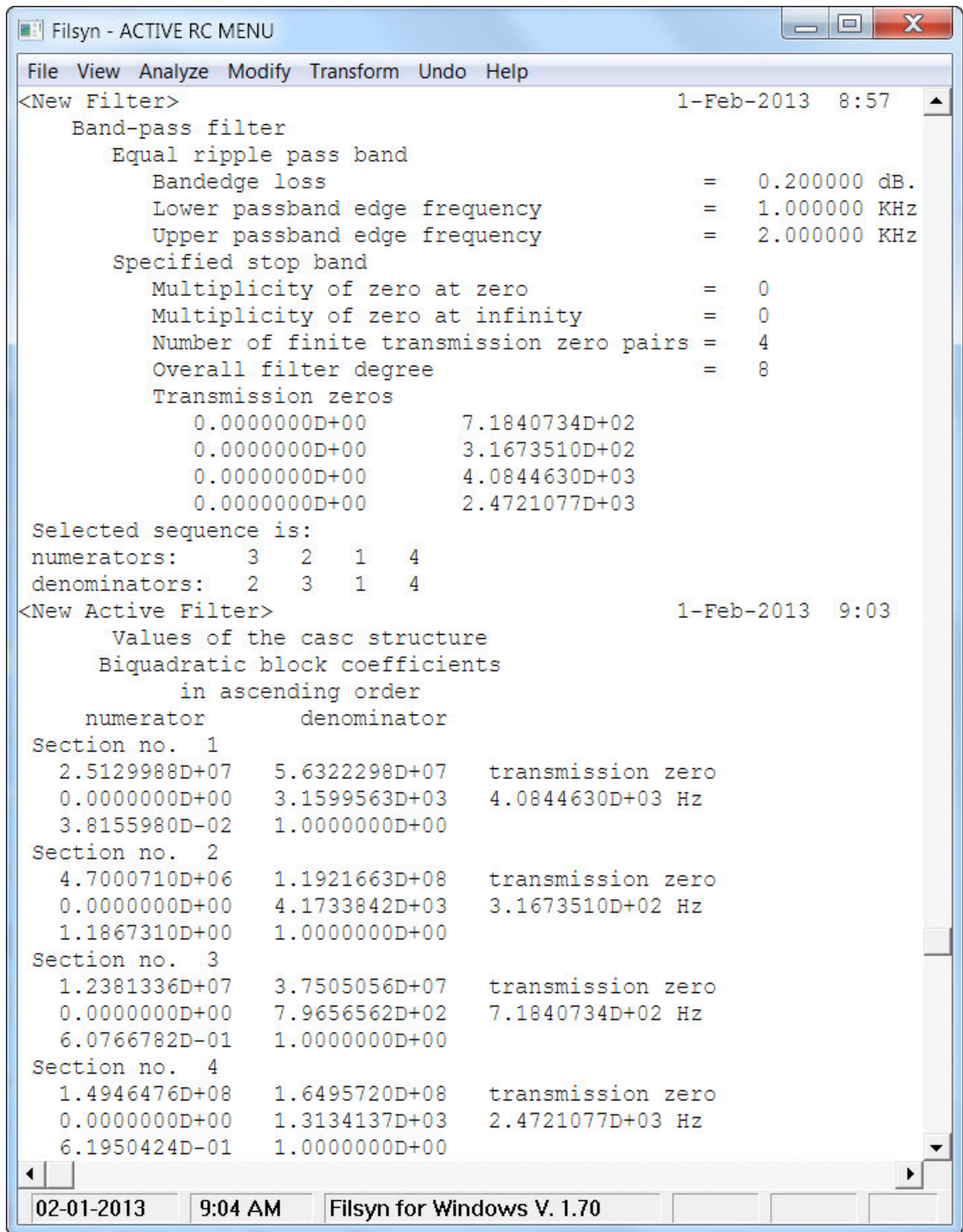
☒ Continue to synthesis

☐ Modify some parameters

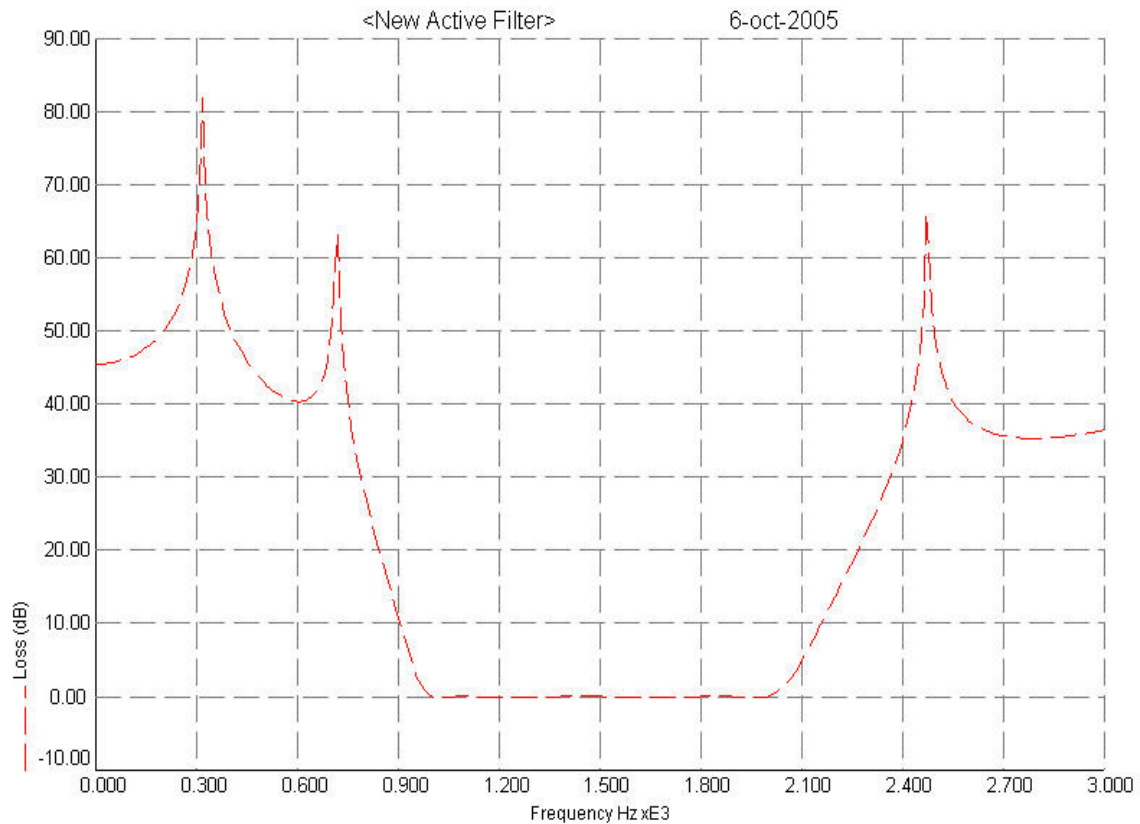
OK Cancel

02-01-2013 9:02 AM Filsyn for Windows V. 1.70

Proceeding and selecting the cascade implementation, the results are as follows:



This is satisfactory, because the wide passband yields low pole Q (below 10) values. The calculated loss vs. frequency plot is shown below and indicates a satisfactory design.

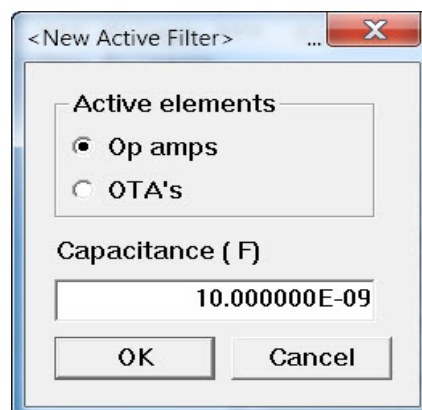


Note that the sequence of second order sections and pairing of numerator and denominator factors is computer-selected to optimize the dynamic range. However, if we so desire, this arrangement may be changed using the **Modify->Permute** menu option.

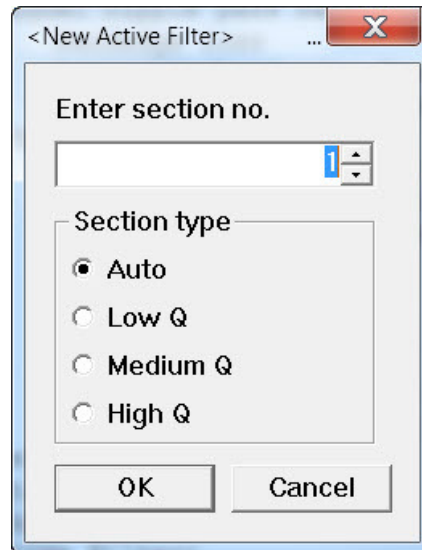
For FLF and leapfrog structures, we also have a program-selected sequence, but the program will first ask, if it is acceptable. If it is not, we may select our own sequence. This is due to the fact that in these structures the **Permute** menu option is not available.

3. Cascade design

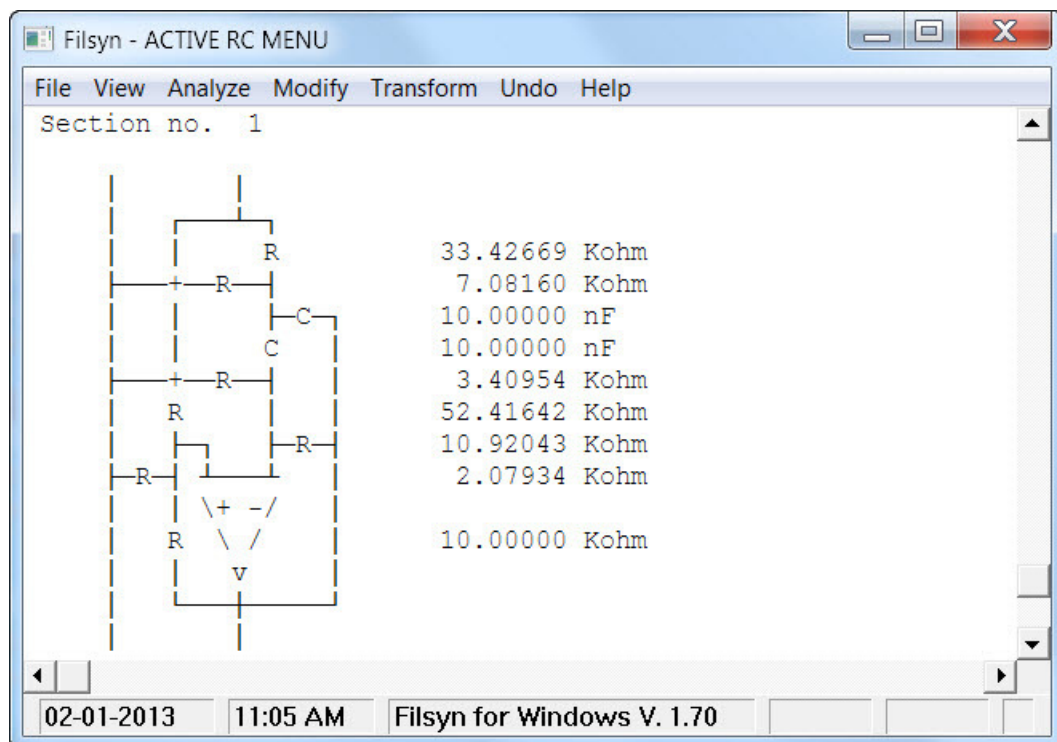
Now we proceed to the actual design using the **View->Design** menu option:



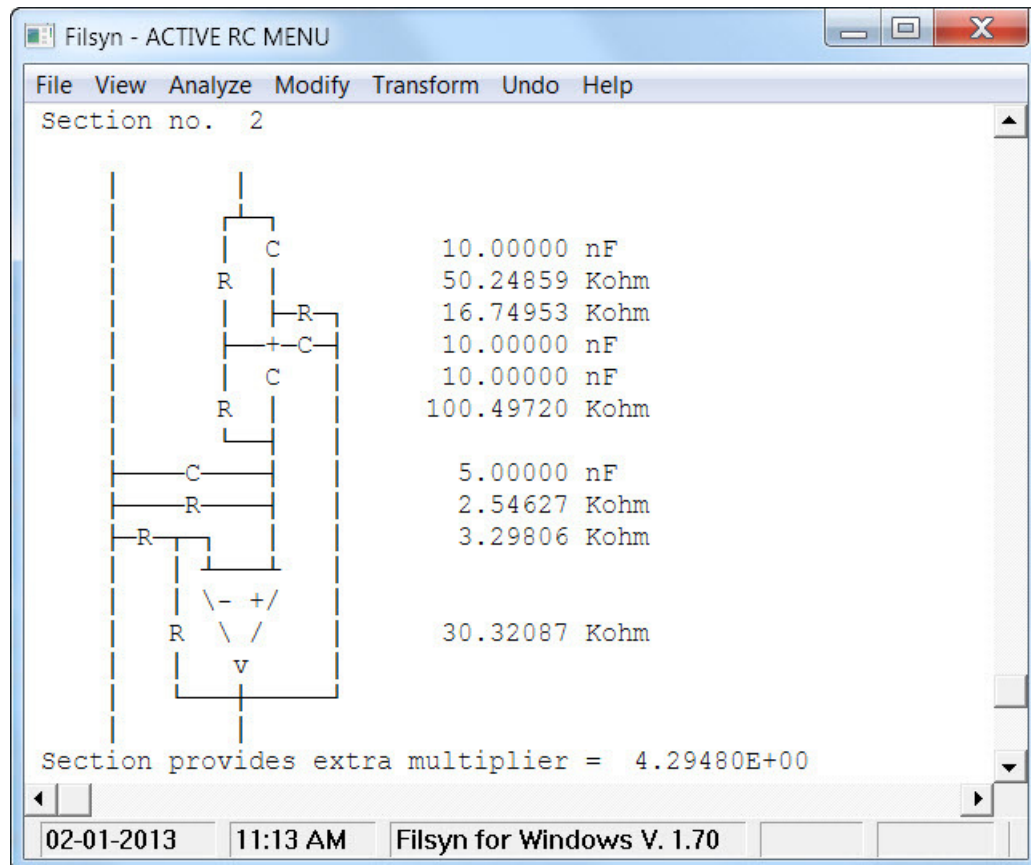
We use the operational amplifier implementation, as opposed to the operational trans-conductance amplifier one. Note that this choice is made only once, the first time the *Design* menu is selected. We also need to select a (common) capacitance value, which will be used subsequently in all sections. Next we select the section to be designed, since using the operational amplifier implementation we have several circuits we can use and hence we either must select what we think the section Q is or let the program select an appropriate circuit for us:



This time the program selected the following implementation for section 1:

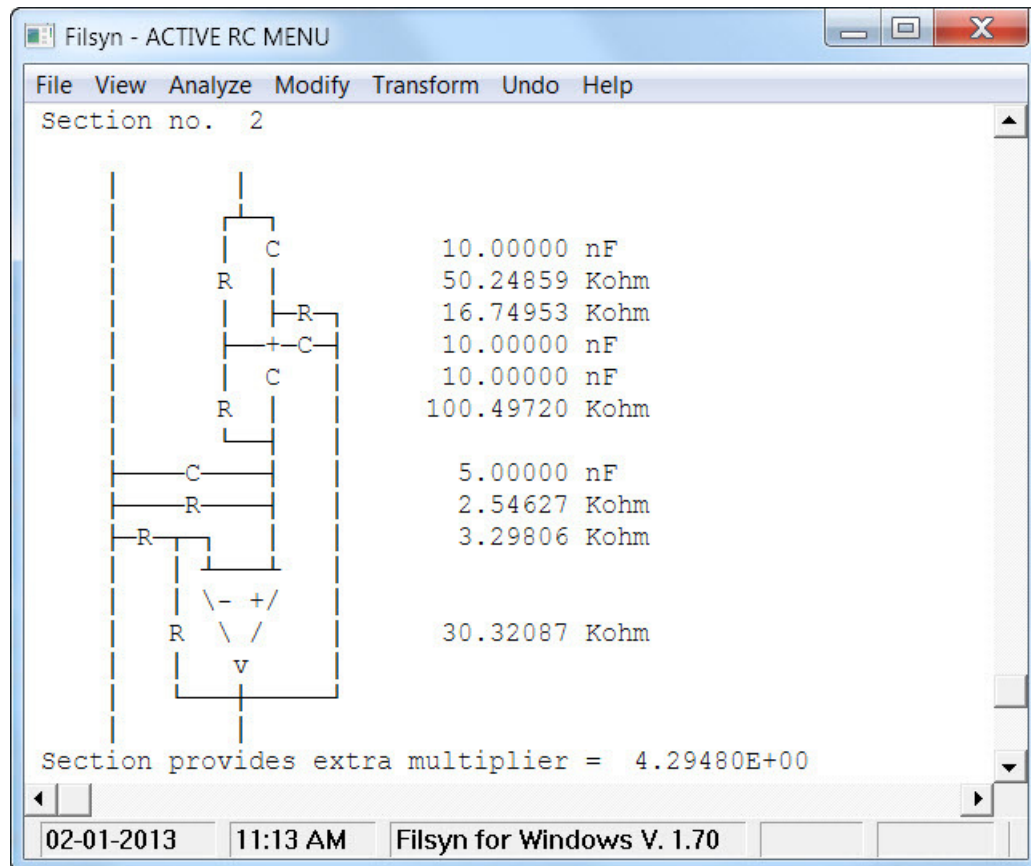


If we dislike this circuit, we can repeat the design and select what we think the Q of the section is, although in this particular case that is simply a way to select another circuit. The computer-selected section no. 2 is next:



Note the fact that this implementation realizes the third section only within a multiplier. We may disregard this as it only represents a flat loss, or compensate for it by multiplying another section by the inverse of this quantity. Whether that is possible and can be implemented without additional components is something we have to find out by trial and error. Another effect of this multiplier is that it reduces the dynamic range the filter is able to handle.

Alternatively, we have redesigned this section specifying medium Q value instead of the program selected auto option and selected the twin-T option when offered, yielding the exact same implementation:



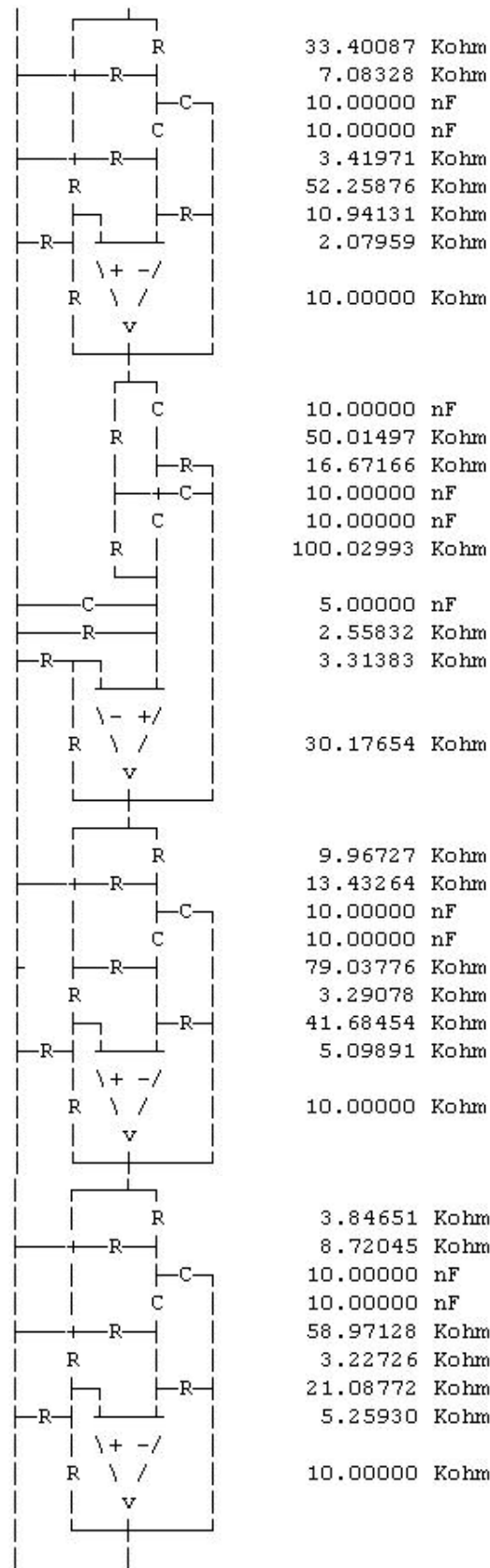
After designing the remaining sections 3 and 4 (not shown), we save the complete design in a text file using the **File->Textfile** menu option. The resulting file summarizes both the design data as well as the complete implementation. We had to break up this text file into two segments and leave out the Parts list completely, that is at the end:

```

*** S/Filsyn ***  Filter Program
<New Active Filter>                                     6-oct-2005 10:48
  Band-pass filter
    Equal-ripple pass band
      Bandedge loss                                     = 0.200000 dB.
      Lower passband edge frequency                     = 1.000000 KHz
      Upper passband edge frequency                     = 2.000000 KHz
    Specified stopband type
      Multiplicity of zero at zero                     = 0
      Multiplicity of zero at infinity                  = 0
      Number of finite transmission zeros               = 4
      Overall filter degree                             = 8
      Transmission zeros
        real part      imaginary part
        0.000000D+00   7.1848618D+02
        0.000000D+00   3.1821467D+02
        0.000000D+00   4.0787773D+03
        0.000000D+00   2.4717376D+03
<New Active Filter>                                     6-oct-2005
10:48
  Values of the casc structure
  biquadratic block coefficients
    in ascending order
      numerator      denominator
Section no.  1
  2.5135679D+07   5.6318342D+07   transmission zero
  0.0000000D+00   3.1601066D+03   4.0787773D+03 Hz
  3.8271096D-02   1.0000000D+00
Section no.  2
  4.7484946D+06   1.1922833D+08   transmission zero
  0.0000000D+00   4.1737015D+03   3.1821467D+02 Hz
  1.1878342D+00   1.0000000D+00
Section no.  3
  1.2385911D+07   3.7505211D+07   transmission zero
  0.0000000D+00   7.9637519D+02   7.1848618D+02 Hz
  6.0775896D-01   1.0000000D+00
Section no.  4
  1.4947232D+08   1.6495598D+08   transmission zero
  0.0000000D+00   1.3129192D+03   2.4717376D+03 Hz
  6.1972113D-01   1.0000000D+00

```

Active RC filter using biquads



4. Leapfrog design

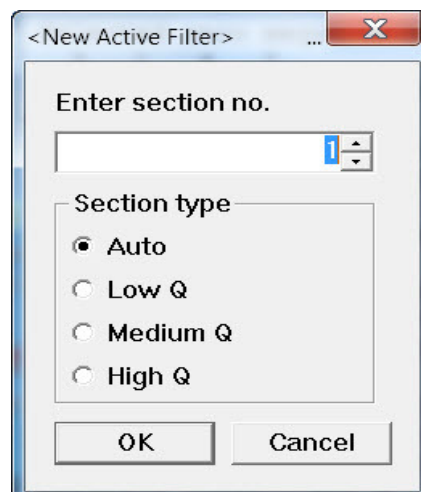
Assuming that the cascade realization is not adequate for sensitivity reasons, we repeat the design by using the leapfrog method of synthesis. The synthesis phase of this design results the following set of data:

```

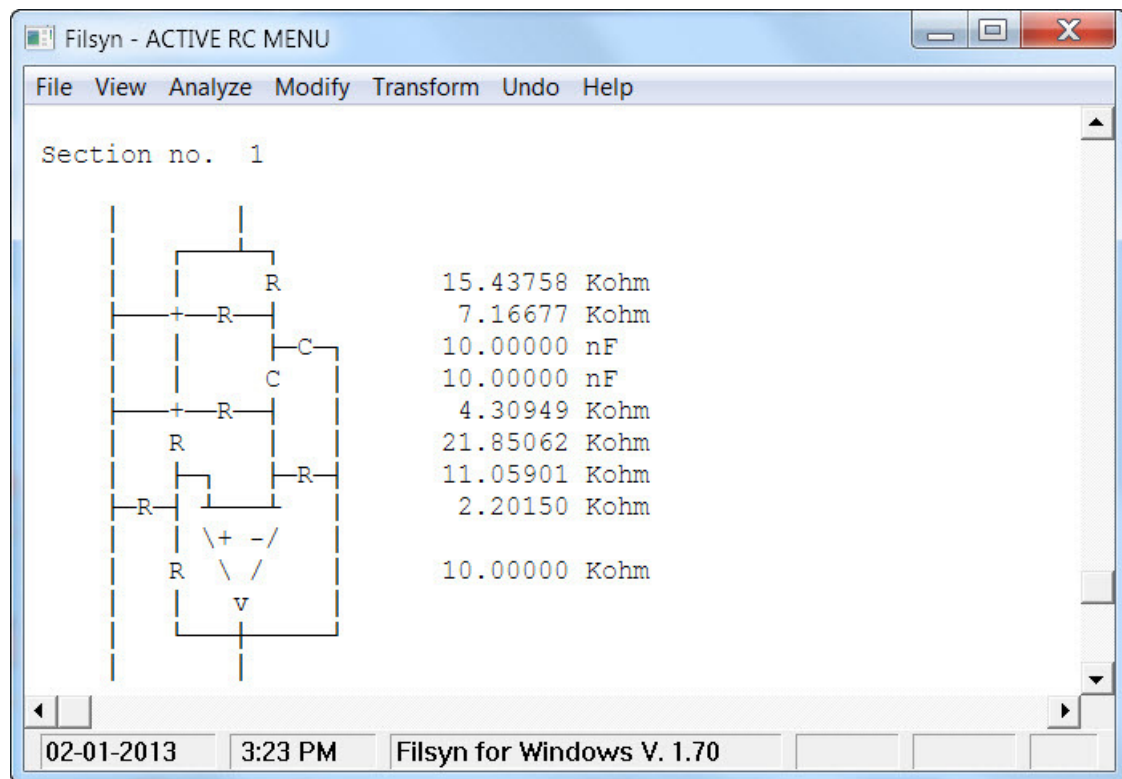
Filsyn - ACTIVE RC MENU
File View Analyze Modify Transform Undo Help
Selected zero sequence is:
  3   1   4   2
      leapfrog synthesis
<New Active Filter>                                     1-Feb-2013 12:54
  Values of the leap structure
  Biquadratic block coefficients
    in ascending order
      numerator      denominator
Section no.  1
  6.0303888D+07    8.9926482D+07    transmission zero
  0.0000000D+00    4.7167479D+03    4.0851658D+03 Hz
  9.1530579D-02    1.0000000D+00
Section no.  2
  1.5237564D+07    7.0842387D+07    transmission zero
  0.0000000D+00    0.0000000D+00    7.1840305D+02 Hz
  7.4785856D-01    1.0000000D+00
Feedback coefficient = -8.4552182D-01
Section no.  3
  6.0507313D+07    9.5472514D+07    transmission zero
  0.0000000D+00    0.0000000D+00    2.4721498D+03 Hz
  2.5078327D-01    1.0000000D+00
Feedback coefficient = -1.1019705D+00
Section no.  4
  5.9936588D+06    7.9754306D+07    transmission zero
  0.0000000D+00    5.9737342D+03    3.1660303D+02 Hz
  1.5146146D+00    1.0000000D+00
Feedback coefficient = -6.5107702D-01
  
```

02-01-2013 3:09 PM Filsyn for Windows V. 1.70

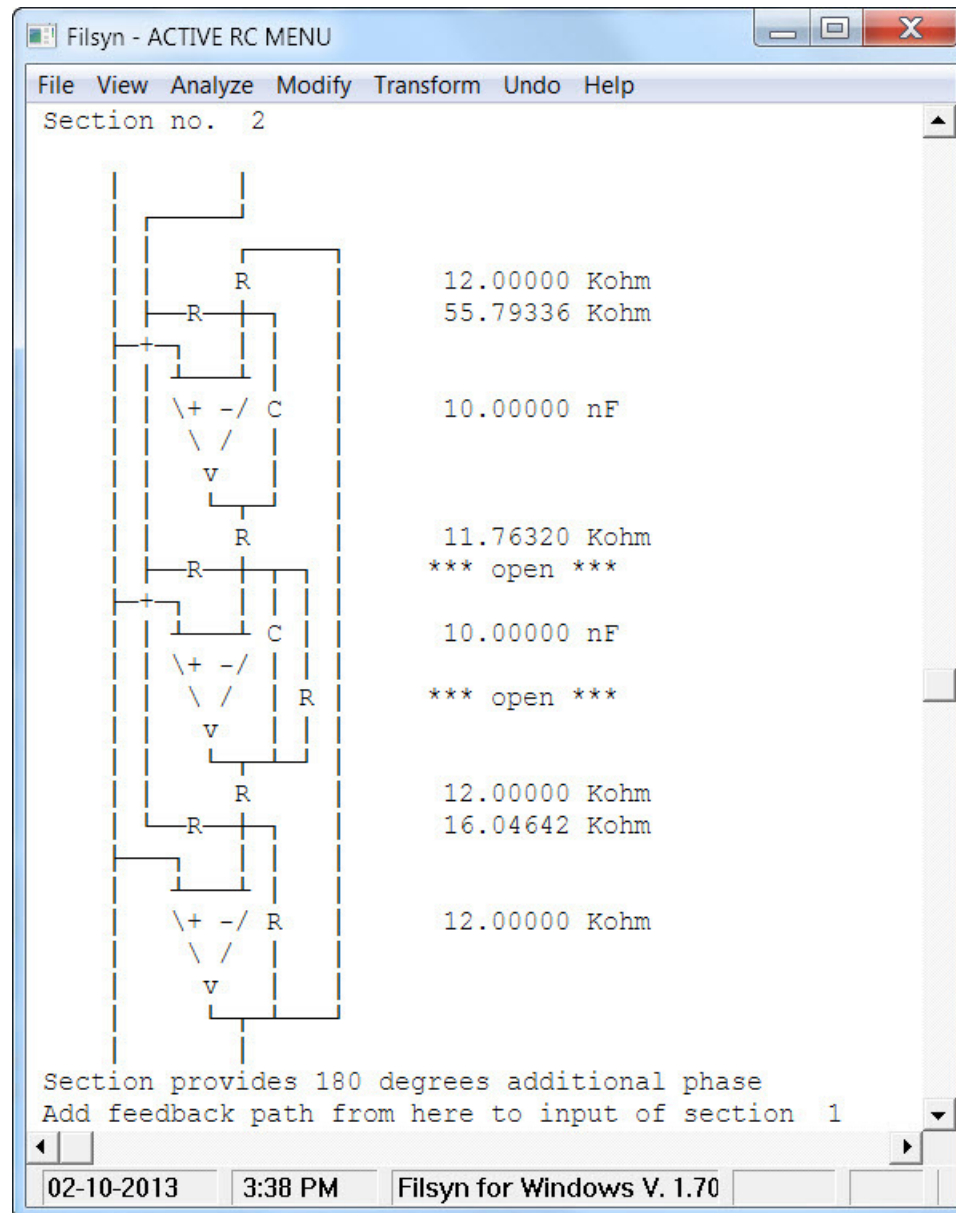
During the implementation of the first section, we use the operational amplifier form again and a preferred capacitance value. The design itself was uneventful:



A feedback path will lead from the output of section 2 to the input of this section, and the only way the input and the feedback signal can be combined is by using an amplifier summing stage, which will also introduce a phase reversal (sign change).

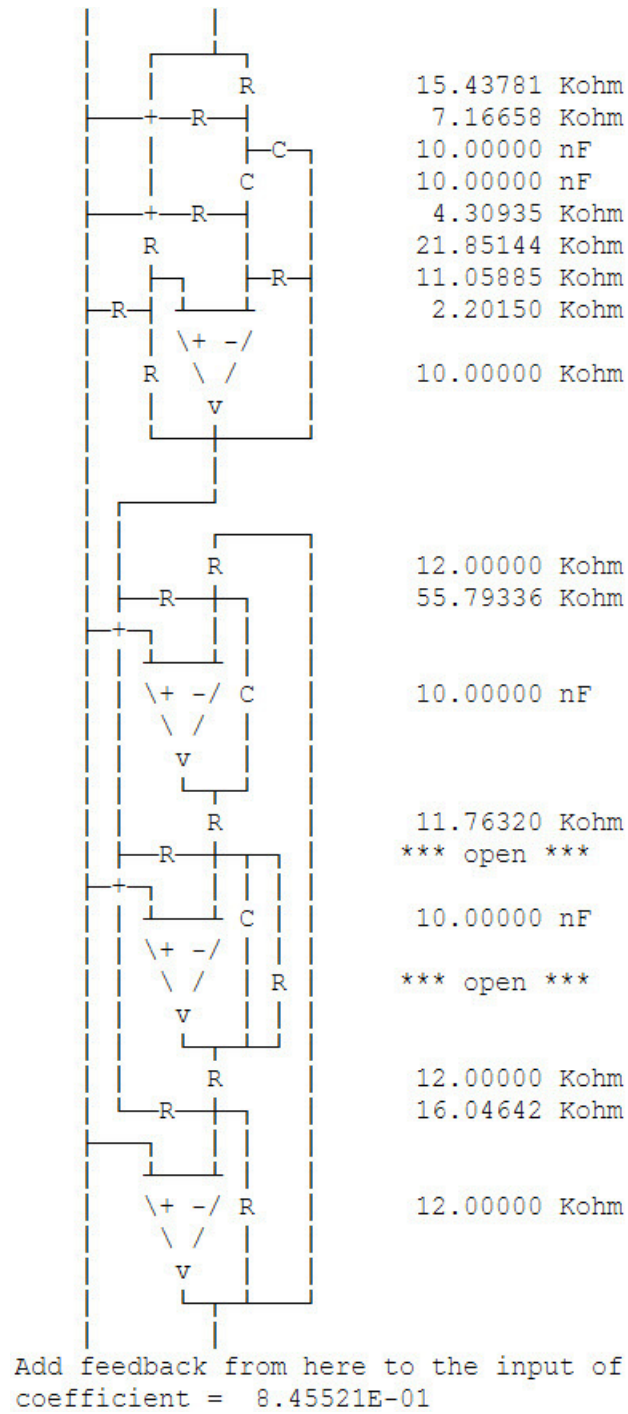


The next, second, stage is of an infinite Q section and can be realized only by a three op-amp biquad section. The three op-amp biquad can accept feedback signal without an extra summing stage, so the realization is acceptable. The three op-amp implementation is obtained for the infinite Q section cases by either the **auto** or the **high Q** selection.

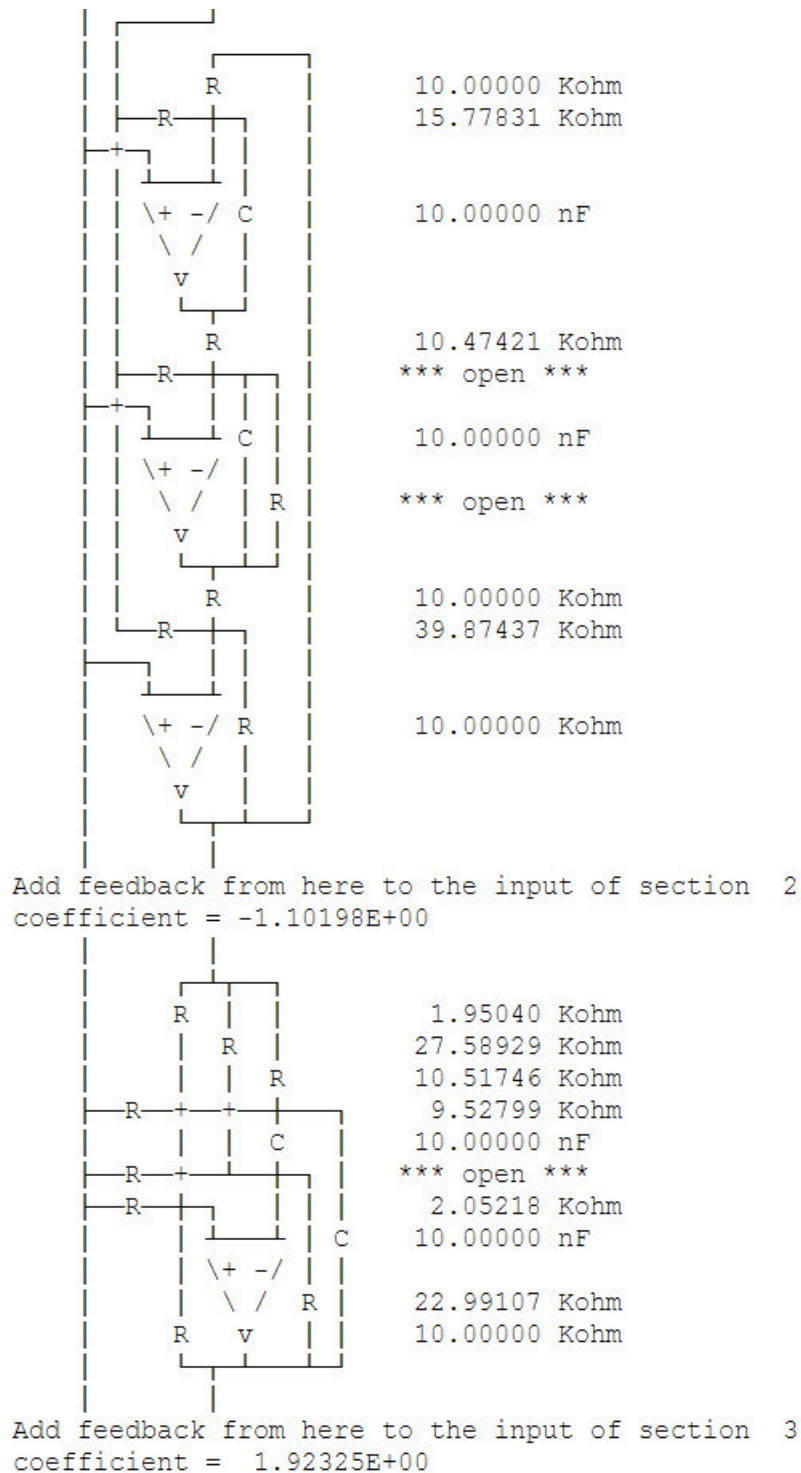


The *** open *** resistors represent open-circuits, i.e. the resistors are missing. The reasoning used above applies to the third section, so we used the three op-amp version again. One could make the program use the two op-amp circuit by reducing the infinite Q values of the internal sections to a high, but finite Q (see the **Transform->Finite Q** menu option). Completing the design of section 3 and 4 (not shown), we save the data in a text file again and show just the circuit that implements the filter in two segments:

Active RC filter using biquads



and



5. Overall structure

To complete the design, we must introduce the feedback resistors and possibly additional amplifiers to serve as summing stages. The feedback coefficients printed in the summary gives the correct value to be used, taking into account of section multipliers, if any, and also the signs.

The first feedback path goes from the output of the second stage to the input of the first, and because it is positive, and because we need a phase inverting summing stage up front, we must put in a second phase inverter, which we put between stages one and two.

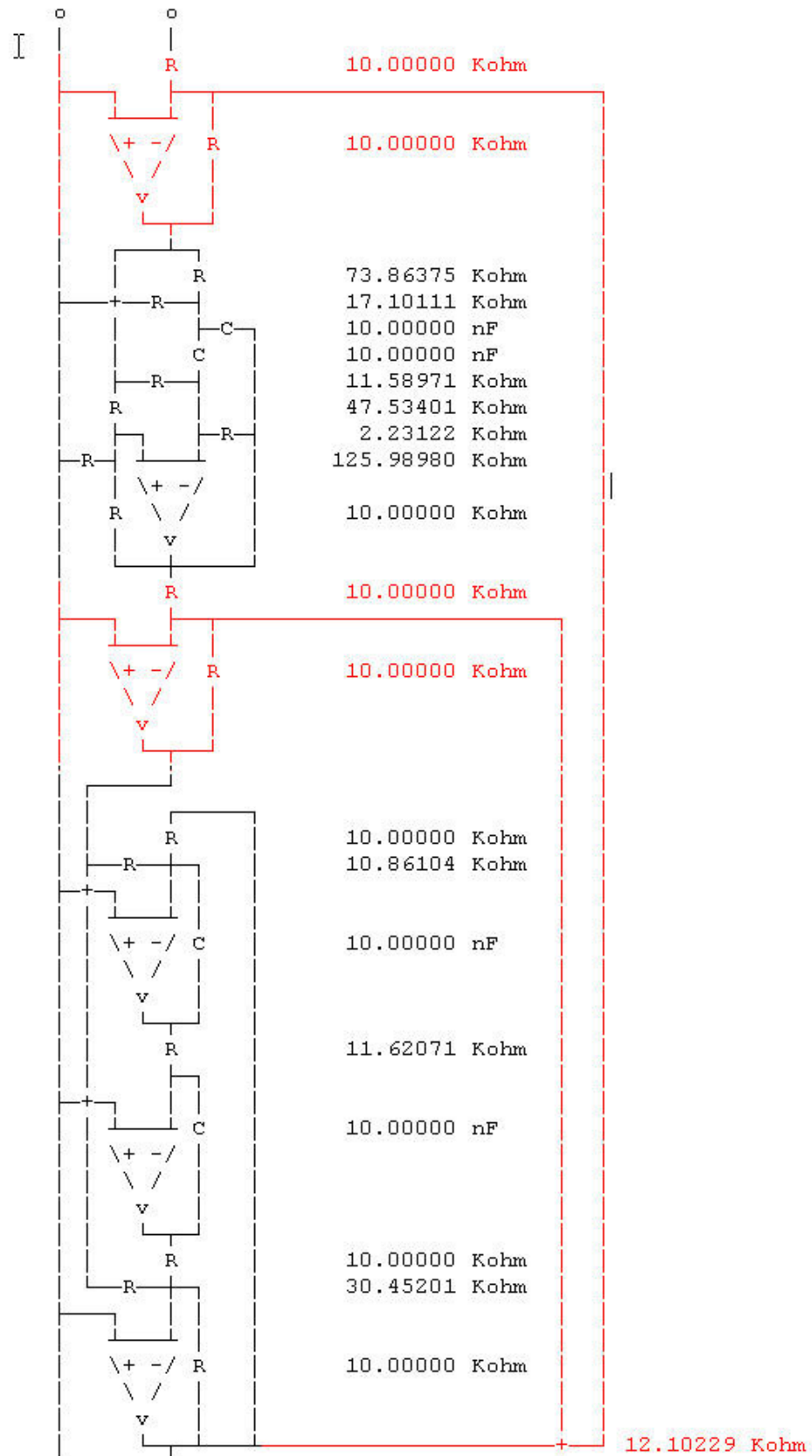
The second feedback coefficient is negative, hence we need a phase inversion which is supplied by the second phase inverter we already have there.

Finally the third feedback coefficient is also positive, hence needs no additional phase inverter stage either.

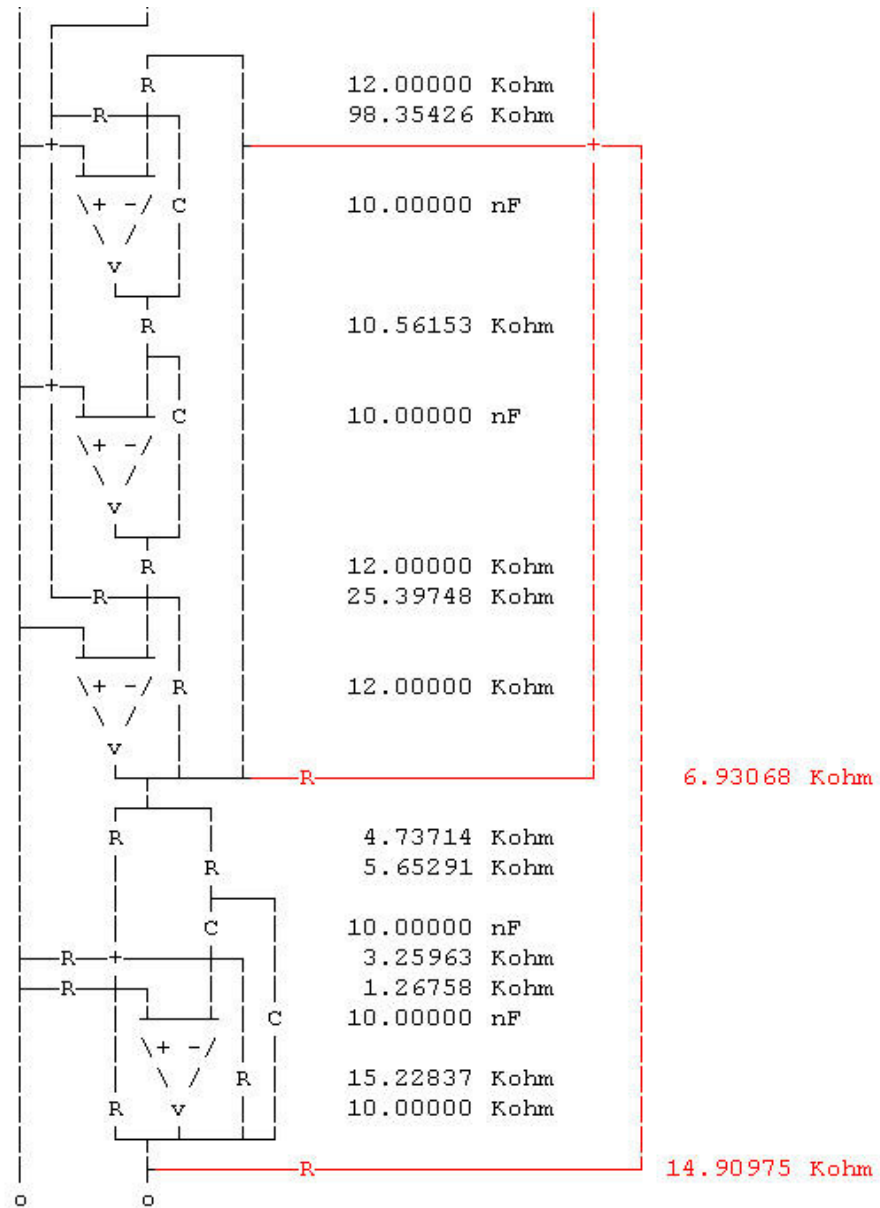
The resistor values representing the feedback coefficients are calculated by taking the other input resistor, and dividing its value by the (magnitude of the) feedback coefficient in question. To simplify the circuit we deleted all the open resistors.

The complete circuit is shown below, where the additional circuitry needed to implement the feedback paths is indicated in red. The number of amplifiers needed for this realization is two and a half times that needed for the cascade one, but the number of capacitors is less.

This figure can *not* be obtained from the program, it was prepared by editing the stored data of the complete session shown above.



Active RC filter using biquads



APPLICATION NOTE 9

DESIGN OF SWITCHED-CAPACITOR FILTERS

1. Introduction.

As described in *Chapter 6* of this manual, the switched-capacitor filter design capability is built into the active-RC segment of the program. The main reason for this was our desire to be able to offer not only the cascaded second order block implementation of these filters, but also the **Leapfrog** and the **Follow-the-leader** type feedback structures as well. These implementations yield substantial sensitivity improvements over the cascaded case.

This arrangement however, created another problem, that of having to perform the *prewarping* of the analog design manually, so that after applying the bilinear Z-transformation, the transformed transfer function should have the proper behavior. Naturally, it would be highly desirable to have the program automatically perform this prewarping step instead.

In this *Application Note* we shall describe two methods for doing this, one for the case where we wish to use the leapfrog structure and another, somewhat simpler, method for the case when the cascaded second order block implementation is acceptable.

2. Leapfrog bandpass filter.

If we need a leapfrog structure, the design *must* go through as an essentially analog one and we convert it to a sampled-data format only at the very end. Therefore, we have to find the pre-warped equivalents of all critical frequencies before commencing the design, and this can be done by a preliminary run through the digital option of the **Filsyn** program.

Consider a bandpass filter with passband from 10 KHz to 15 KHz with a 0.1 dB passband ripple, a lower stopband below 9.5 KHz and 45 dB minimum loss and an upper stopband from 16 KHz with a 35 dB loss. The sampling frequency is assumed to be 64 KHz, a low enough value, which makes it impossible to neglect the warping effect.

This data is entered into **Filsyn** in the usual manner using the **Design->Filter->New** menu option of the starting data entry screen.

The *Sampling frequency* entry needs an explanation. The **Placer** segment will display the frequencies during the iterations in normal (not prewarped) forms. In order to request the display of these values in prewarped form, we enter the sampling frequency as a negative value. This only changes this display, otherwise has no effect.

Switched capacitor filters

<New Filter>

Filter kind

- ☐ LC
- ☐ Microwave
- ☐ Active RC
- ☒ IIR Digital

Quarter-wave/Sampling freq (Hz)

-64.000000E+03

Filter type

- ☐ Low pass
- ☐ High pass
- ☒ Band pass

Lower passband freq (Hz)

10.000000E+03

Upper passband freq (Hz)

15.000000E+03

Passband type

- ☐ Max. flat
- ☒ Equal ripple
- ☐ Functional
- ☐ Sloping

Band-edge loss/return loss (dB)

0.100000000

Loss Slope (dB/oct)

6.00000000

Flat loss (dB)

0.00000000

Function Type

- ☒ E
- ☐ F

Multiplier

0.00000000

Stopband

- ☐ Monotonic
- ☐ Equal min
- ☒ Placer
- ☐ Specified

Lower stopband freq (Hz)

0.000000

Loss (dB)

50.0000000

Upper stopband freq (Hz)

0.000000

Loss (dB)

50.0000000

Detail Parameters

of zeros at zero

1

of zeros at FQ or FS/2

1

of unit elements

0

Parametric

- ☒ Conventional
- ☐ Parametric
- ☐ Matching

R1

50.000000E+00

R2

50.000000E+00

ZS parameter

-1

Q for predistortion

0.00000000

Predistortion

- ☒ Passband only
- ☐ Pass/stop bands

Shifted bandpass trans.

Center frequency (Hz)

0.000000

☐ Odd parametric

☐ Save design data

OK Cancel

where the **Detail Parameters** contain the stopband requirements:

Filter Parameters

Requirements			Fixed zeros		Movable zeros	
	Frequency (Hz)	Loss (dB)	Frequency (Hz)	Multiplicity	Frequency (Hz)	Multiplicity
1	9.500000E+03	45.00000000	1	0.000000	1	0.000000
2	16.000000E+03	35.00000000	2	0.000000	2	0.000000
3	0.000000	0.00000000	3	0.000000	3	0.000000
4	0.000000	0.00000000	4	0.000000	4	0.000000
5	0.000000	0.00000000	5	0.000000	5	0.000000
6	0.000000	0.00000000	6	0.000000	6	0.000000
7	0.000000	0.00000000	7	0.000000	7	0.000000
8	0.000000	0.00000000	8	0.000000	8	0.000000
9	0.000000	0.00000000	9	0.000000	9	0.000000
10	0.000000	0.00000000	10	0.000000	10	0.000000
11	0.000000	0.00000000			11	0.000000

or no. of zeros below passband

3

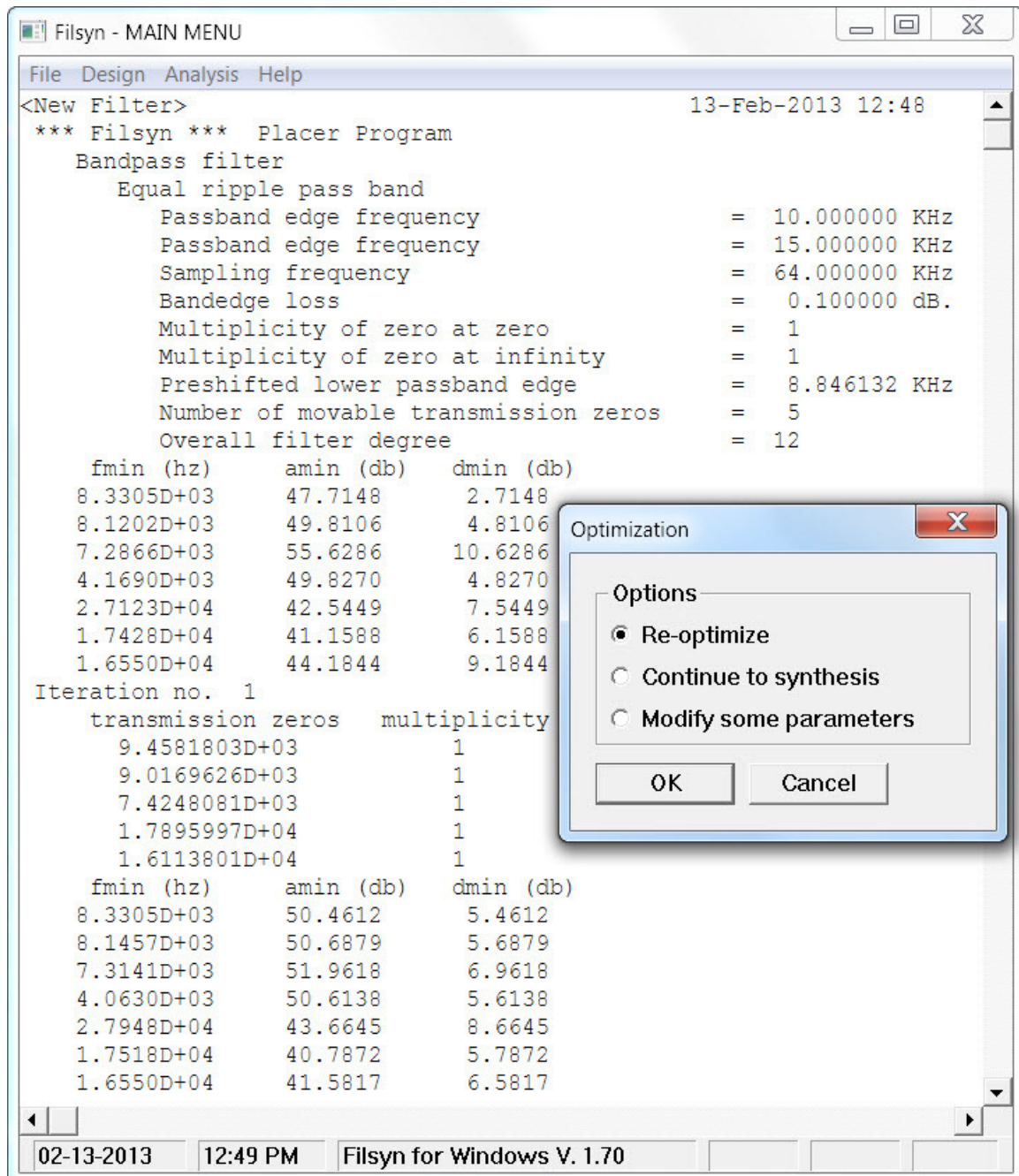
and/or no. of zeros above passband

2

OK Cancel

We estimated the need for three zeros in the lower stopband and two zeros in the upper.

Next we show the usual **Placer** header and the first iteration:



It is simple to read off the prewarped frequencies from this output. The lower passband edge is specifically indicated (8.84613 KHz), while the upper edge remains unchanged, due to our method of internal normalization, namely, the (upper) passband edge is always selected as our normalization frequency.

The prewarped equivalents of the specified breakpoints are those of the tabulated FMIN values above, which do *not* change from iteration to iteration. These are obviously 8.3305 KHz (corresponding to the 9.5 KHz breakpoint) and 16.55 KHz (corresponding to the 16 KHz value we entered).

Switched capacitor filters

We note these values and restart the whole procedure as follows, this time as an active RC design:

<New Filter>

Filter kind

- ☐ LC
- ☐ Microwave
- ☒ Active RC
- ☐ IIR Digital

Quarter-wave/Sampling freq (Hz)

Filter type

- ☐ Low pass
- ☐ High pass
- ☒ Band pass

Lower passband freq (Hz)

Upper passband freq (Hz)

Passband type

- ☐ Max. flat
- ☒ Equal ripple
- ☐ Functional
- ☐ Sloping

Band-edge loss/return loss (dB)

Loss Slope (dB/oct)

Flat loss (dB)

Function Type

- ☒ E
- ☐ F

Multiplier

Stopband

- ☐ Monotonic
- ☐ Equal min
- ☒ Placer
- ☐ Specified

Lower stopband freq (Hz)

Loss (dB)

Upper stopband freq (Hz)

Loss (dB)

Detail Parameters

of zeros at zero

of zeros at infinity

of unit elements

Parametric

- ☒ Conventional
- ☐ Parametric
- ☐ Matching

R1

R2

ZS parameter

Q for predistortion

Predistortion

- ☒ Passband only
- ☐ Pass/stop bands

Shifted bandpass trans. Center frequency (Hz)

☐ Odd parametric

☐ Save design data

and also the stopband data, all with the prewarped frequencies:

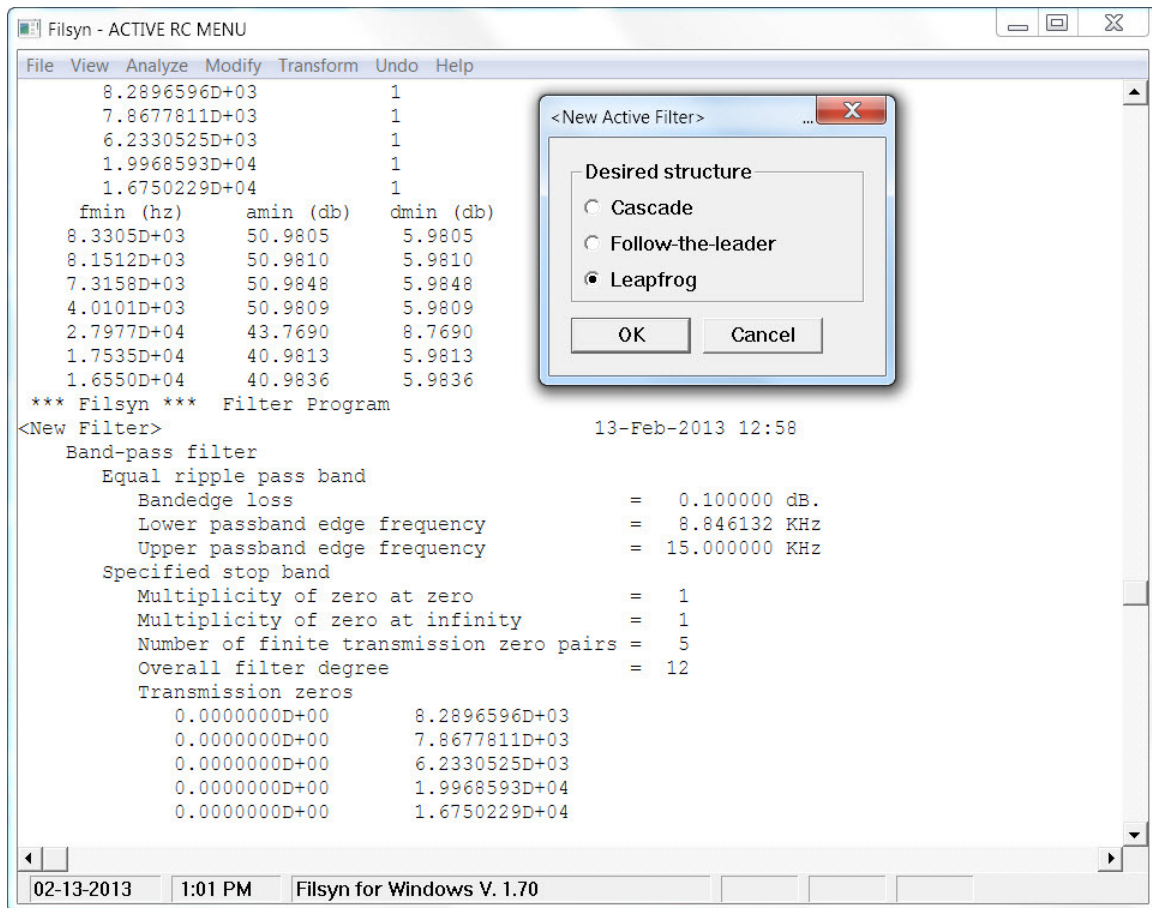
Filter Parameters

Requirements			Fixed zeros			Movable zeros		
	Frequency (Hz)	Loss (dB)		Frequency (Hz)	Multiplicity		Frequency (Hz)	Multiplicity
1	8.330500E+03	45.00000000	1	0.00000	0	1	0.00000	0
2	16.550000E+03	35.00000000	2	0.00000	0	2	0.00000	0
3	0.00000	0.00000000	3	0.00000	0	3	0.00000	0
4	0.00000	0.00000000	4	0.00000	0	4	0.00000	0
5	0.00000	0.00000000	5	0.00000	0	5	0.00000	0
6	0.00000	0.00000000	6	0.00000	0	6	0.00000	0
7	0.00000	0.00000000	7	0.00000	0	7	0.00000	0
8	0.00000	0.00000000	8	0.00000	0	8	0.00000	0
9	0.00000	0.00000000	9	0.00000	0	9	0.00000	0
10	0.00000	0.00000000	10	0.00000	0	10	0.00000	0
11	0.00000	0.00000000				11	0.00000	0

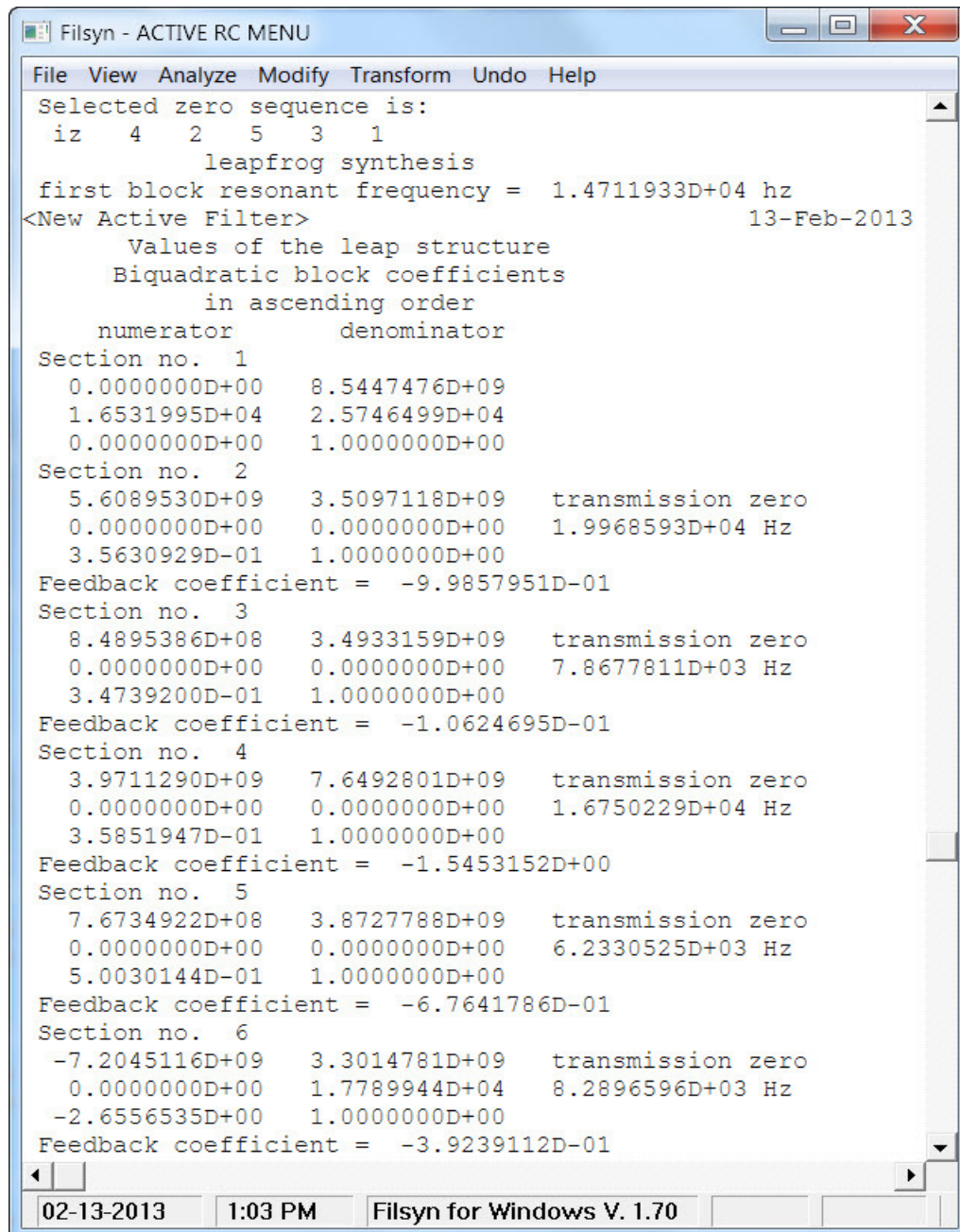
or no. of zeros below passband

and/or no. of zeros above passband

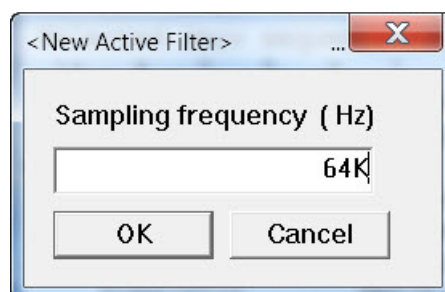
The rest is routine, we let the **Placer** option optimize the stopbands and then we proceed to the design segment and specify the **Leapfrog** structure of this filter:

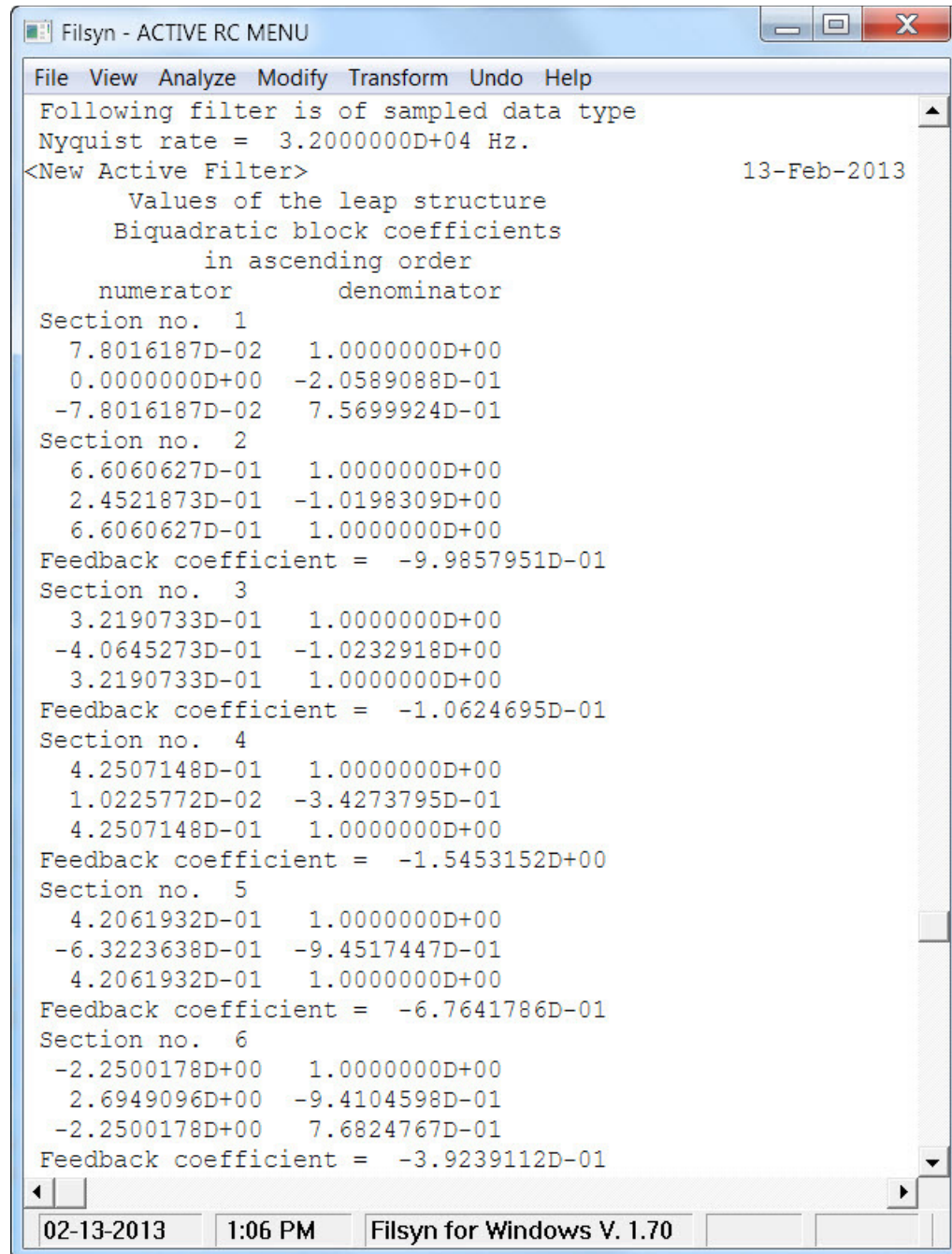


The program at one point asks for the required sequence of transmission zeros with an auto (computer selected) as an option. Selecting that we have the filter:



At this point we can select the **Transform->Z transform** menu item and get the sampled-data function that can be implemented by a switched-capacitor circuit:





Finally, the circuit implementation, using the **View->Design** menu item (see Figures 4 and 5 on pages 159-160 in Chapter 6 of this Manual for details) is given by:

Switched capacitor filters

Section no.	1	output = out2					
ca =	9.70310	cb =	9.70310	cc =	19.88188	cd =	9.70310
ce =	0.00000	cf =	3.11475	cg =	1.00000	ch =	1.00000
ci =	1.00000	cj =	0.00000	ck =	0.00000		

Section no.	2	output = out1					
ca =	1.51376	cb =	1.51376	cc =	3.18956	cd =	2.00000
ce =	1.00000	cf =	1.43965	cg =	5.93393	ch =	0.00000
ci =	4.42179	cj =	1.00000	ck =	1.04857		

feedback to previous section:	cg	g =	0.99858	ch	h =	0.99858
cii =	0.99858	cjj =	0.00000			

Section no.	3	output = out2					
ca =	3.10648	cb =	3.10648	cc =	4.11485	cd =	4.21298
ce =	0.00000	cf =	0.00000	cg =	1.00000	ch =	0.00000
ci =	1.00000	cj =	1.00000	ck =	0.00000		

feedback to previous section:	cg	g =	0.63046	ch	h =	0.00000
cii =	0.46980	cjj =	0.10625			

Section no.	4	output = out1					
ca =	2.35255	cb =	2.35255	cc =	4.71922	cd =	2.00000
ce =	1.00000	cf =	3.18862	cg =	4.27710	ch =	0.00000
ci =	4.31879	cj =	1.00000	ck =	1.52639		

feedback to previous section:	cg	g =	1.54532	ch	h =	0.00000
cii =	1.54532	cjj =	1.54532			

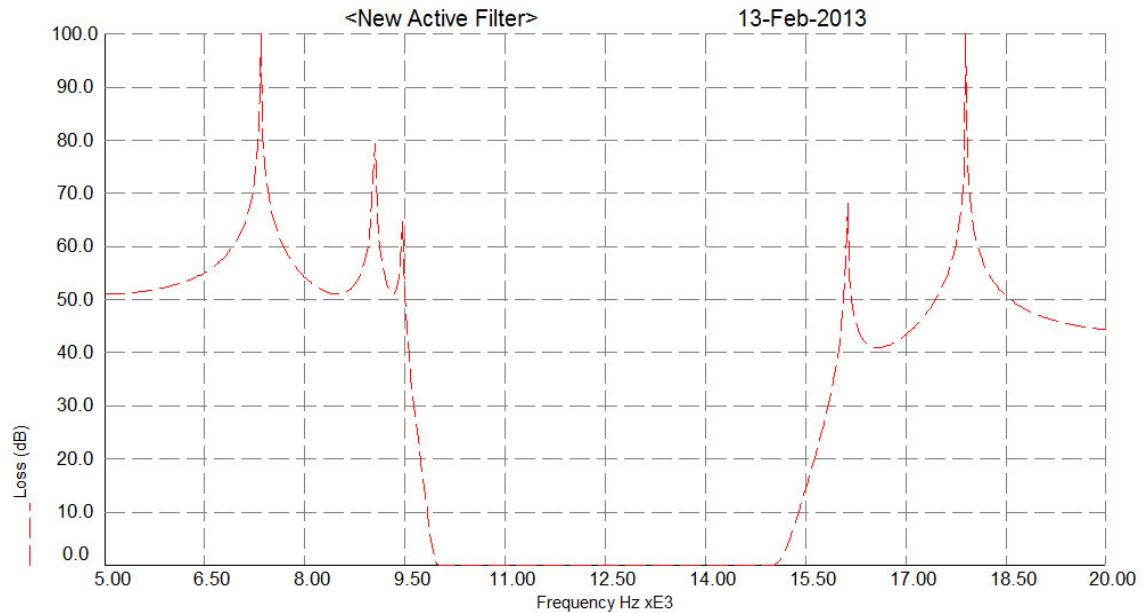
Section no.	5	output = out2					
ca =	2.37745	cb =	2.37745	cc =	5.04696	cd =	4.78464
ce =	0.00000	cf =	0.00000	cg =	1.00000	ch =	0.00000
ci =	1.00000	cj =	1.00000	ck =	0.00000		

feedback to previous section:	cg	g =	2.89310	ch	h =	0.00000
cii =	2.92131	cjj =	0.67642			

Section no.	6	output = out2					
ca =	1.00000	cb =	1.00000	cc =	3.56933	cd =	4.31495
ce =	1.00000	cf =	0.00000	cg =	7.78903	ch =	0.00000
ci =	2.25002	cj =	2.25002	ck =	0.00000		

If we wish to change the program-selected configurations we can simply repeat the menu item **View->Design** and specify different section types as desired.

An analysis in the frequency domain confirms the correctness of this design.



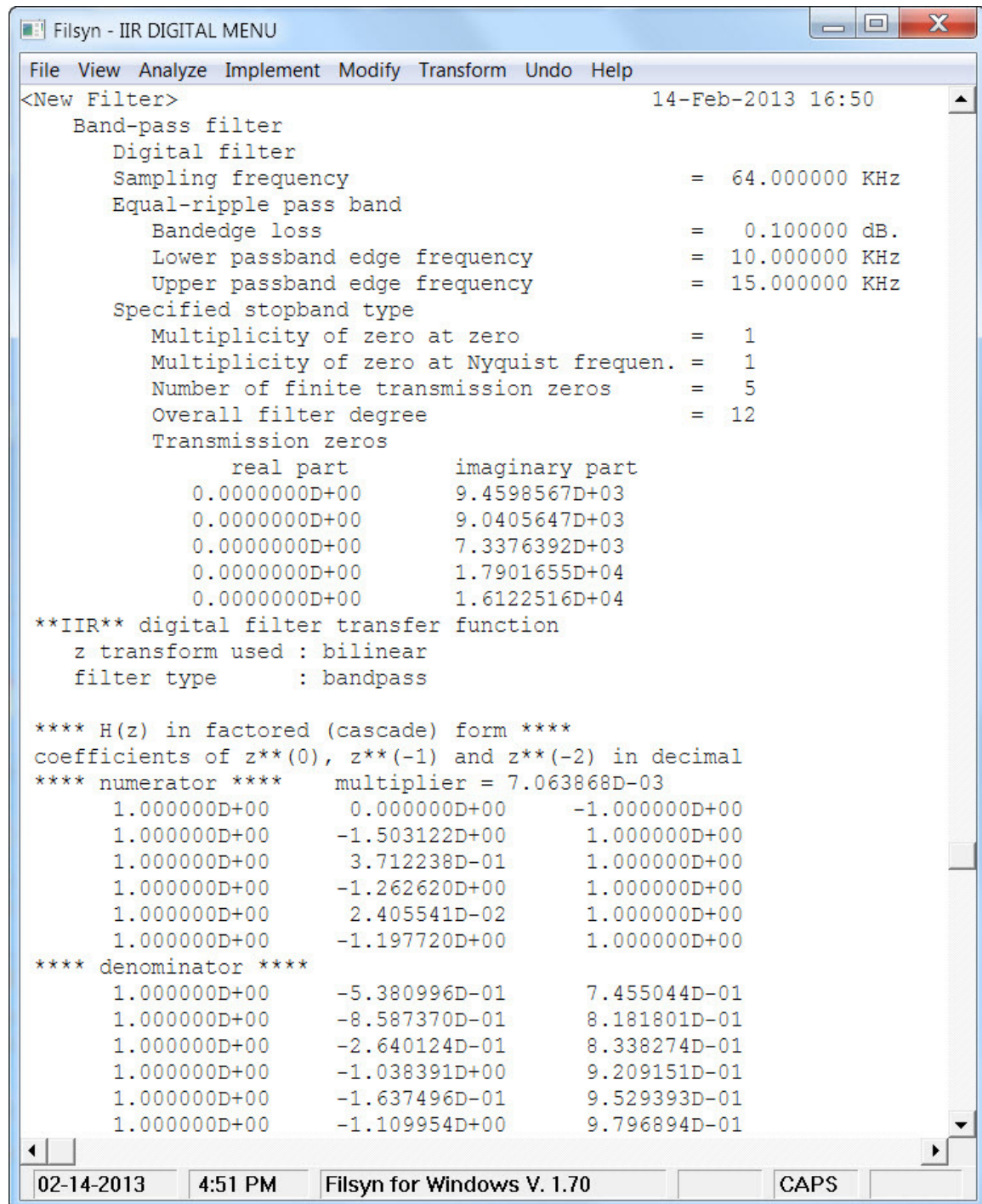
The only disadvantage of this procedure is that if we wish to delay equalize this filter, it *must* be done *before* the **Z Transform** menu item is used and therefore a warping effect will be seen on the final results; in this particular case showing about a 10% slope superimposed on the flat delay. A solution to this problem will be suggested in the summary below.

3. Cascaded design.

We shall consider the exact same filter specification, but assume that a cascaded second-order block implementation will be acceptable.

In this case we start out exactly the same way we have done above, starting the design as an IIR digital filter. Instead of doing the prewarping manually here, it is done at the beginning automatically.

After the **Placer** routine has done its job and we select the cascade form:



We do a quick frequency domain analysis in the passband (not shown) and follow that with a request for delay equalization using the **Transform->Equalizer** menu and specify two second order sections and a passband of a bit reduced in size, since it is always difficult to equalize the delay over the whole band.

<New Digital Filter>

Number of 1st order sections

Number of 2nd order sections

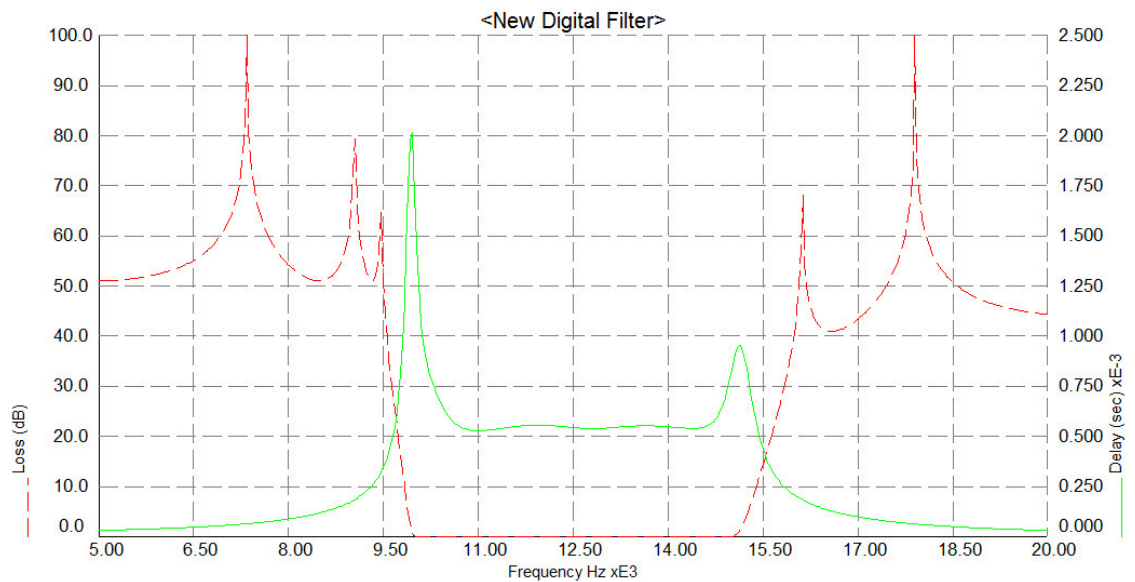
Lower approx. band edge (Hz)

Upper approx. band edge (Hz)

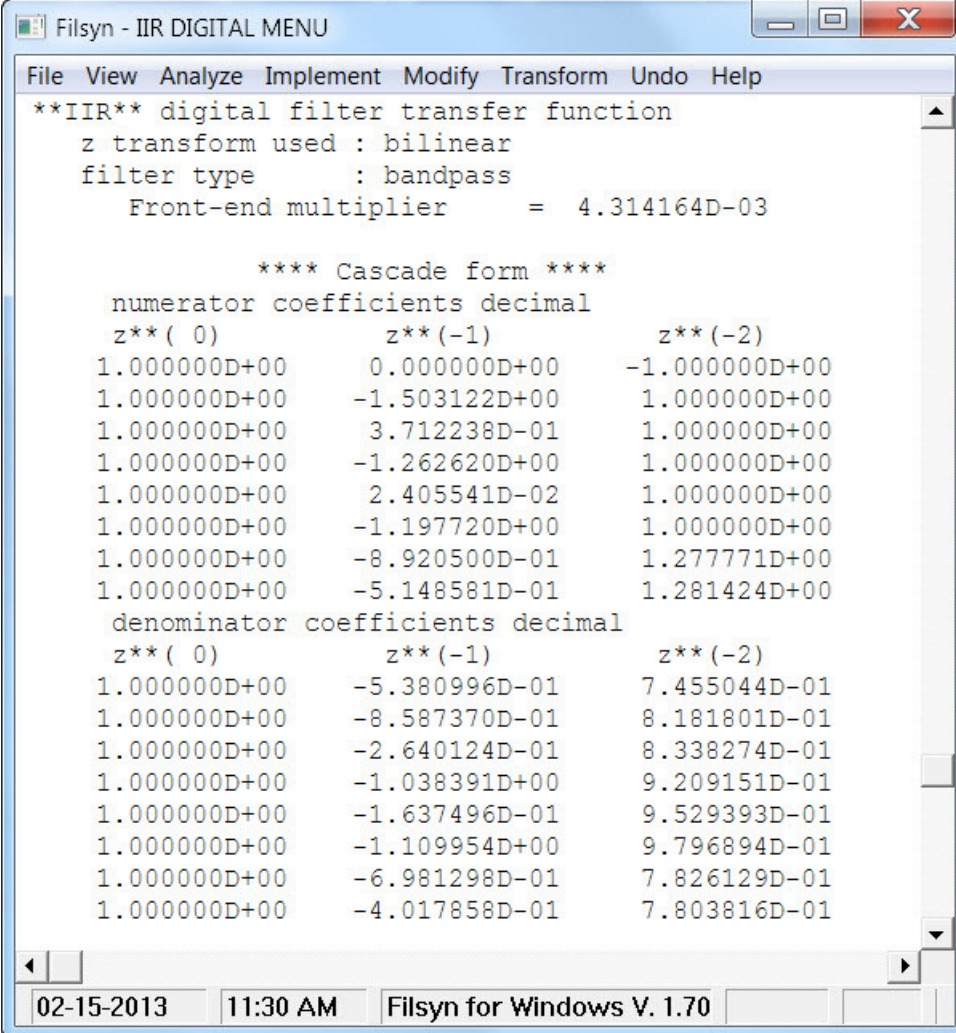
Stepsize

OK Cancel

The iteration converged in 6 steps, we accepted the results and incorporated the equalizer sections into the filter (sections 7 and 8 in the figure). First we show the analysis results:



The complete filter is next, containing the delay equalizer sections (the last two in the figure):



The screenshot shows a window titled "Filsyn - IIR DIGITAL MENU" with a menu bar (File, View, Analyze, Implement, Modify, Transform, Undo, Help). The main text area displays the following information:

```

**IIR** digital filter transfer function
z transform used : bilinear
filter type      : bandpass
Front-end multiplier = 4.314164D-03

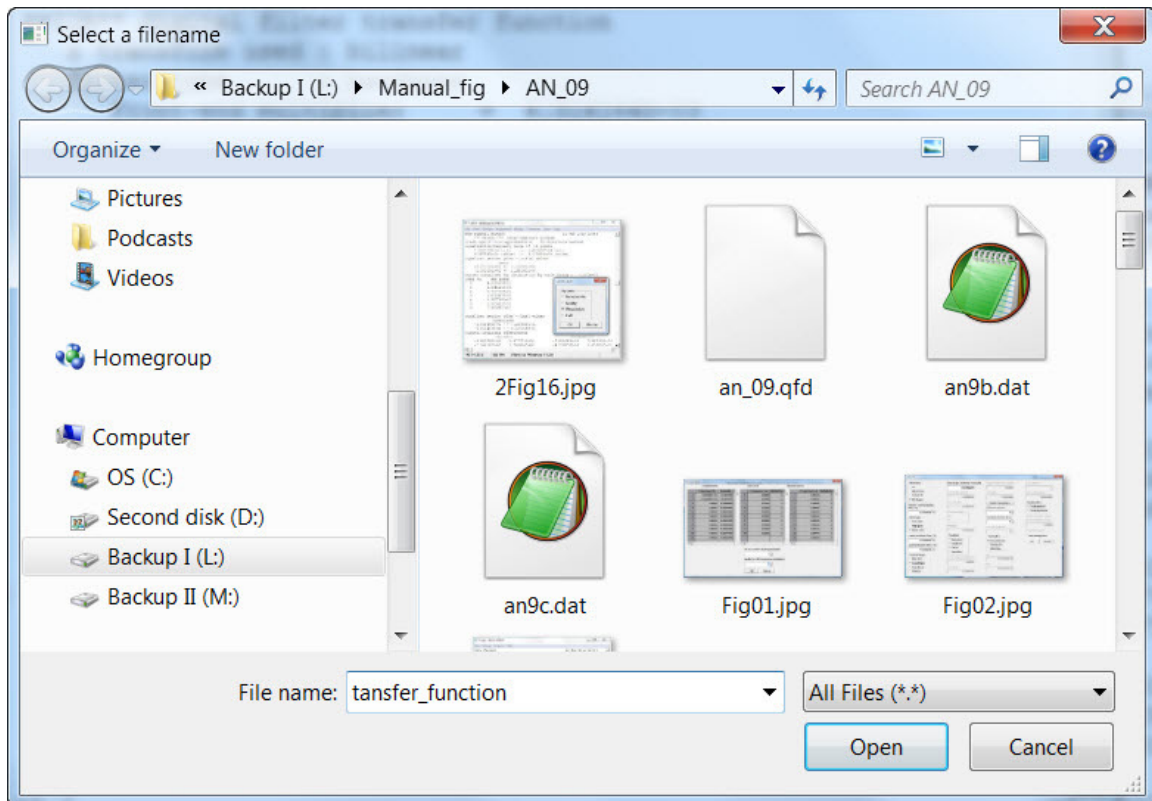
**** Cascade form ****
numerator coefficients decimal
z**( 0)      z**(-1)      z**(-2)
1.000000D+00  0.000000D+00  -1.000000D+00
1.000000D+00 -1.503122D+00  1.000000D+00
1.000000D+00  3.712238D-01  1.000000D+00
1.000000D+00 -1.262620D+00  1.000000D+00
1.000000D+00  2.405541D-02  1.000000D+00
1.000000D+00 -1.197720D+00  1.000000D+00
1.000000D+00 -8.920500D-01  1.277771D+00
1.000000D+00 -5.148581D-01  1.281424D+00
denominator coefficients decimal
z**( 0)      z**(-1)      z**(-2)
1.000000D+00 -5.380996D-01  7.455044D-01
1.000000D+00 -8.587370D-01  8.181801D-01
1.000000D+00 -2.640124D-01  8.338274D-01
1.000000D+00 -1.038391D+00  9.209151D-01
1.000000D+00 -1.637496D-01  9.529393D-01
1.000000D+00 -1.109954D+00  9.796894D-01
1.000000D+00 -6.981298D-01  7.826129D-01
1.000000D+00 -4.017858D-01  7.803816D-01

```

The status bar at the bottom shows the date "02-15-2013", time "11:30 AM", and version "Filsyn for Windows V. 1.70".

Now we have the required digital filter satisfying all the requirements. However we are in the digital segment of the program but we wish to implement it in a switched capacitor form and hence we need to transfer the data to the active RC segment.

This would be a difficult step; first we have to find and copy the transfer function data of this filter and re-enter it into the active RC segment. There is a menu item to make this process very simple. First we call the menu **View->Switched-C** that finds the data and writes it into a file:



The resulting file *transfer_function.dat* contains the data:

```
! pole-zero data for switched-capacitor filter
! <New Digital Filter> 14-Feb-2013 16:59
! numerator quadratic factors
  4.314164D-03    0.000000D+00   -4.314164D-03
  1.000000D+00   -1.503122D+00    1.000000D+00
  1.000000D+00    3.712238D-01    1.000000D+00
  1.000000D+00   -1.262620D+00    1.000000D+00
  1.000000D+00    2.405541D-02    1.000000D+00
  1.000000D+00   -1.197720D+00    1.000000D+00
  1.000000D+00   -8.920500D-01    1.277771D+00
  1.000000D+00   -5.148581D-01    1.281424D+00
! denominator quadratic factors
  1.000000D+00   -5.380996D-01    7.455044D-01
  1.000000D+00   -8.587370D-01    8.181801D-01
  1.000000D+00   -2.640124D-01    8.338274D-01
  1.000000D+00   -1.038391D+00    9.209151D-01
  1.000000D+00   -1.637496D-01    9.529393D-01
  1.000000D+00   -1.109954D+00    9.796894D-01
  1.000000D+00   -6.981298D-01    7.826129D-01
  1.000000D+00   -4.017858D-01    7.803816D-01
```

We can now enter this into the active RC analysis segment by calling **Filsyn** and select the **Analysis->Filter->New** option which opens up the entry window:

Entering filter data

Filter kind

- ☐ Passive LC
- ☐ Microwave
- ☐ Active RC
- ☒ Switched capacitor
- ☐ IIR digital

Form

- ☐ Roots
- ☒ Factors

Sampling/Q.W. freq (Hz)

64.000000E+03

Lower passband freq (Hz)

10.000000E+03

(Upper) passband/norm. (Hz)

15.000000E+03

Poles Zeros

Numerator Denominator

Feedback coeff.

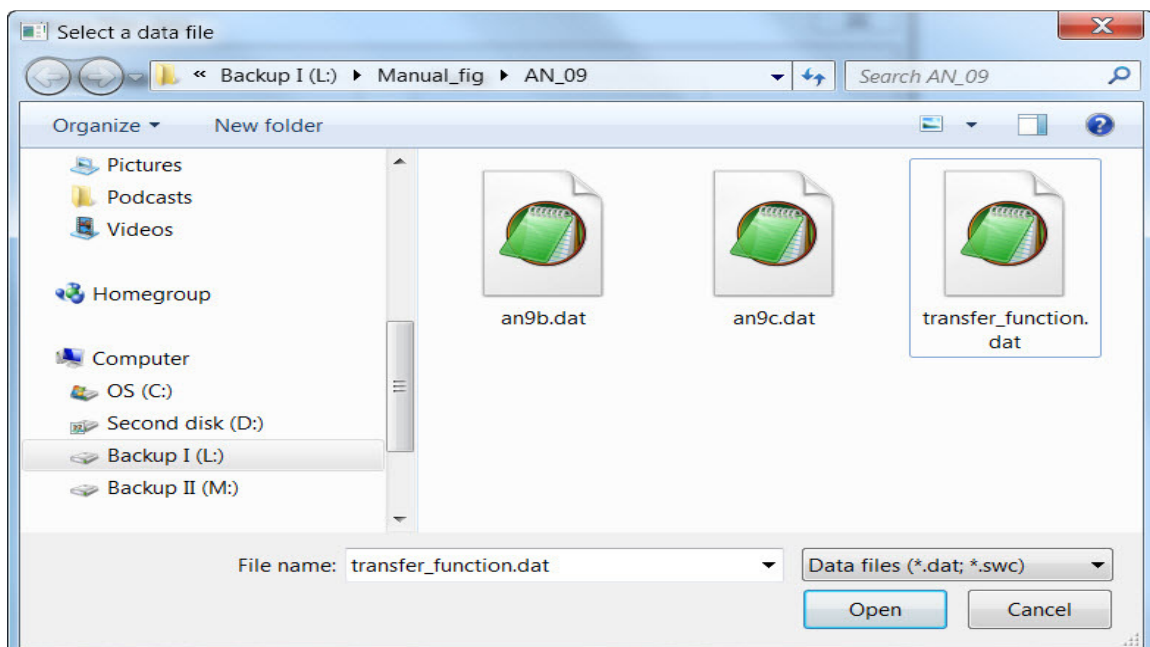
No. of zeros at infinity

0

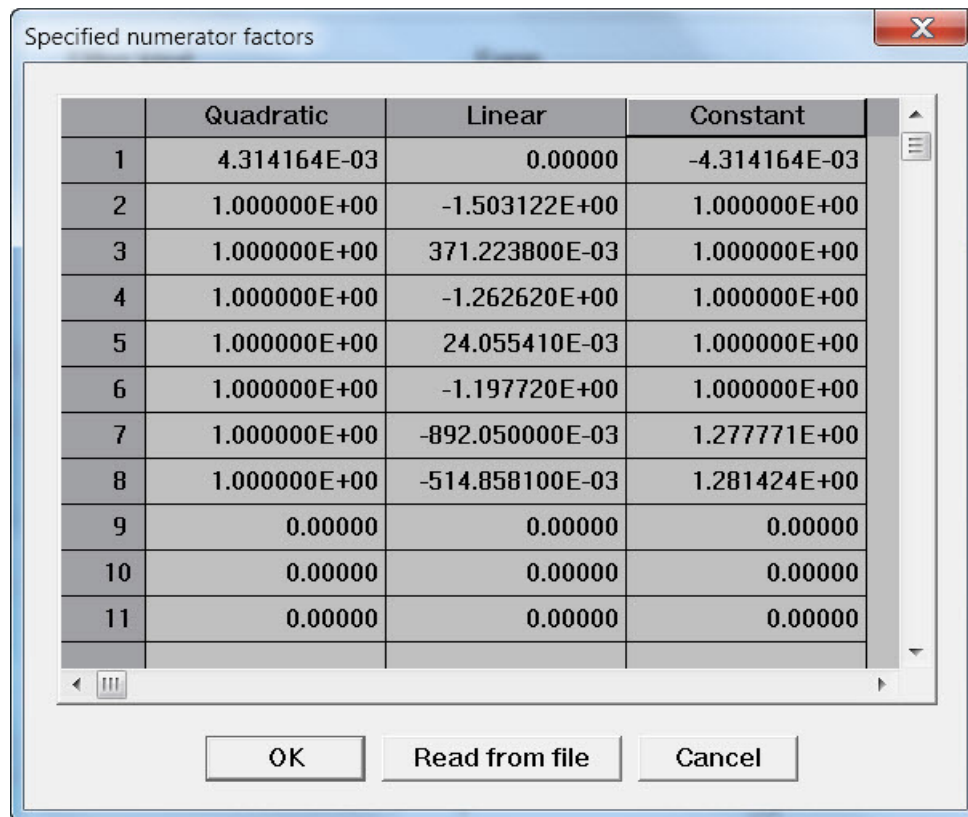
☐ Save analysis data

OK Cancel

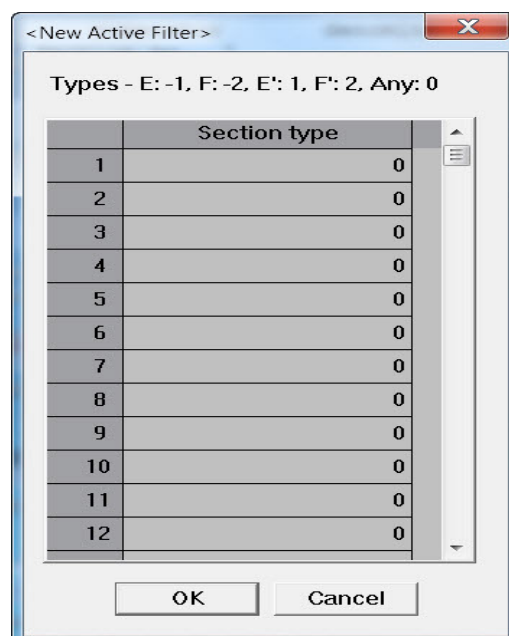
The only items we need to be careful about are the **Switched capacitor** and the **Factors** which is to be selected since the data input is in factored form. These can be read in using the **Numerator** and **Denominator** buttons in that order. Clicking on **Numerator** we can click on **Read from file** and specify the storage file we created:



Repeating that for the **Denominator** button we don't have to specify the file, the program remembers. We can check the data entry by clicking on the **Numerator** button again:



And that is all. We are already in the active RC analysis segment and the next and last decision to be made is selecting the implementations using the **View->Design** option that brings up the menu we discussed previously:



Selecting all zeros (the default) in order to let the program select the forms, we finally get the switched capacitor implementation:

```

Section no.  1  output = out2

ca = 231.79462  cb = 231.79462  cc = 279.86993  cd = 231.79462
ce =  0.00000  cf = -58.99072  cg =  1.00000  ch =  1.00000
ci =  1.00000  cj =  0.00000  ck =  0.00000

Section no.  2  output = out2

ca =  5.49995  cb =  5.49995  cc =  1.93094  cd =  2.01257
ce =  0.00000  cf = -1.00000  cg =  1.00000  ch =  0.00000
ci =  5.49995  cj =  5.49995  ck =  0.00000

Section no.  3  output = out2

ca =  6.01784  cb =  6.01784  cc =  1.56981  cd =  1.00000
ce =  0.00000  cf = -1.00000  cg =  2.37122  ch =  0.00000
ci =  6.01784  cj =  6.01784  ck =  0.00000

Section no.  4  output = out2

ca = 12.64464  cb = 12.64464  cc =  1.19684  cd =  1.35615
ce =  0.00000  cf = -1.00000  cg =  1.00000  ch =  0.00000
ci = 12.64464  cj = 12.64464  ck =  0.00000

Section no.  5  output = out2

ca = 21.24914  cb = 21.24914  cc =  1.78919  cd =  1.00000
ce =  0.00000  cf = -1.00000  cg =  2.02406  ch =  0.00000
ci = 21.24914  cj = 21.24914  ck =  0.00000

Section no.  6  output = out2

ca = 49.23542  cb = 49.23542  cc =  1.08408  cd =  1.24645
ce =  0.00000  cf = -1.00000  cg =  1.00000  ch =  0.00000
ci = 49.23542  cj = 49.23542  ck =  0.00000

Section no.  7  output = out2

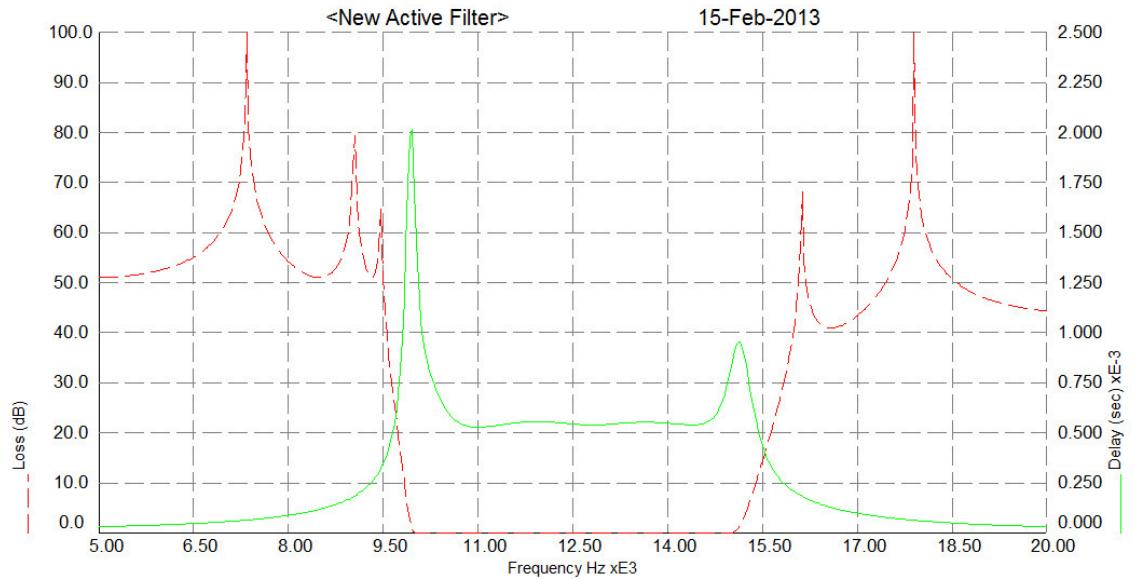
ca =  4.60009  cb =  4.60009  cc =  1.08448  cd =  1.00000
ce =  0.00000  cf = -1.00000  cg =  1.38572  ch =  0.00000
ci =  5.87786  cj =  4.60009  ck =  0.00000

Section no.  8  output = out2

ca =  4.55335  cb =  4.55335  cc =  1.37860  cd =  1.00000
ce =  0.00000  cf = -1.00000  cg =  1.76657  ch =  0.00000
ci =  5.83478  cj =  4.55335  ck =  0.00000

```

The analysis of this filter yields the exact same performance as the digital filter above, that we have started out from:



One small comment about the values shown above. The capacitance ratio of this design is substantially higher than that for the leapfrog form, although either rearranging the sequence of the factors or using other implementations for the sections might reduce it.

4. Summary

Two methods leading to the exact synthesis of (sampled-data) switched capacitor filters are described in this *Application Note*. The first is to be used when leapfrog or follow-the-leader type feedback structure is required. The second method works when the cascaded biquadratic block implementation is acceptable.

Finally note that the two filter functions (without the delay equalizer sections) are, in fact identical. Therefore, if we wish, we could use the delay sections used for version 2 of the filter (sections 7 and 8 on page 521-522 above) for the first version. These sections could simply be transferred (using the **Modify->Add** menu item) to the first design, and then repeat the **View->Design** step to get the switched-C implementation.

APPLICATION NOTE 10

LOWPASS-LIKE BANDPASS FILTERS

1. Introduction

There are instances of bandpass filter design problems, where the desired structure needs the appearance of a lowpass filter. As an example, this happens when the filter needs to pass DC current. In most of these cases the actual stopband loss requirements are either of no significance or required mainly in the upper stopband. **Filsyn** contains several approaches to this group of problems and this *Application Note* will demonstrate these with examples.

2. Impedance matching

When the stopband behavior of the filters is immaterial, but we need to match two unequal terminations over a band of frequencies, and the resulting structure must look like a lowpass, we have a **Matching** option built into the program. This option and its use are described in detail in Chapter 5 of this manual. We mention it here simply because that group of filters belongs to the more general class we describe here.

3. Equal terminations.

The class of filters we wish to describe here has the further requirement that it must be able to have equal terminations. The impedance matching structures mentioned above, clearly do not have this property. The paper by Ralph Levy (see ref. [61]) describes a method of obtaining lowpass-like filters that have an equal-ripple type loss behavior over a finite band, not including the origin and also have equal terminations. His method is restricted to odd degree filters. Here we describe a bit more generalized method, based on an iterative optimization procedure, that yields filters of degree from 5 to 19 (both even and odd degrees) and the passband of either maximally-flat or equal-ripple type. Note that the maximally flat passband case has a closed form solution and hence needs no optimization, but the degree is limited to be between 5 and 15.

To give a little background, we note, that lowpass filters have a loss function as a function of frequency of the form:

$$\text{Loss} = 10 \cdot \log_{10} (1 + |F(s)|^2)$$

Here $F(s)$ is in this case a polynomial, the characteristic function of the frequency ω and $s = j\omega$. For our purposes, we select the following form for $F(s)$:

$$F(s) = K \cdot s^m (s^2 + x_1^2)(s^2 + x_2^2) \dots (s^2 + x_n^2)$$

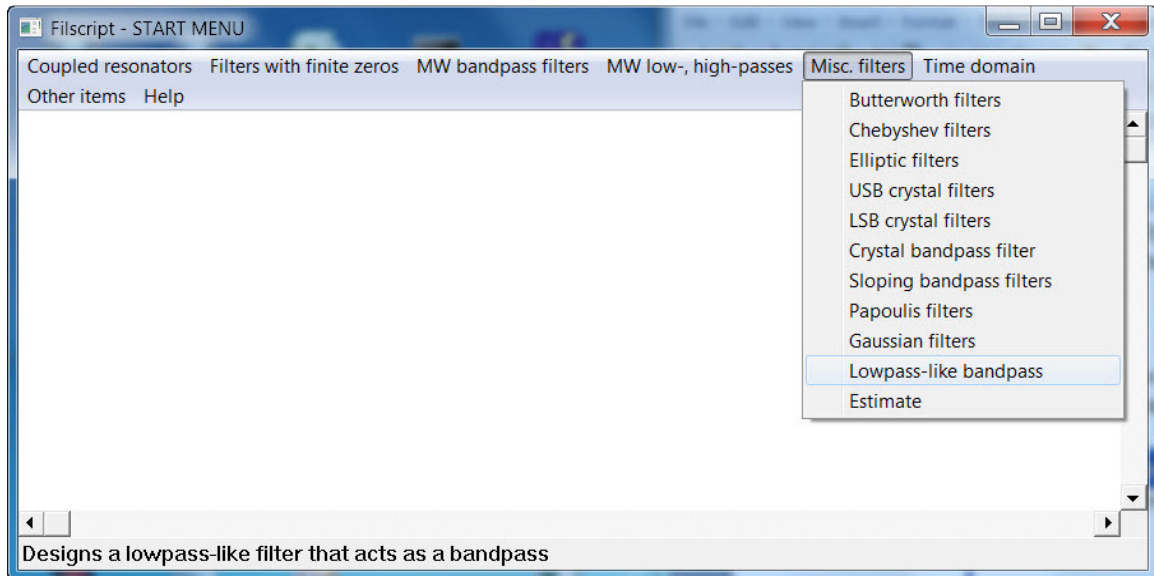
where m is 1 or 2 and n is computed as:

$$n = (\text{degree} - m)/2$$

and it must be an integer, while *degree* is the overall filter degree. The values x_1, x_2, \dots, x_n are the parameters to be determined by the optimization, but they all must be in the filter passband. In the maximally-flat case all x_i values are the same x_0 , and that can be simply computed from the fact that the magnitude of F must be the same at the passband end frequencies. Finally when all x_i values are known, the value of K can be determined from the specified passband loss ripple.

4. Example

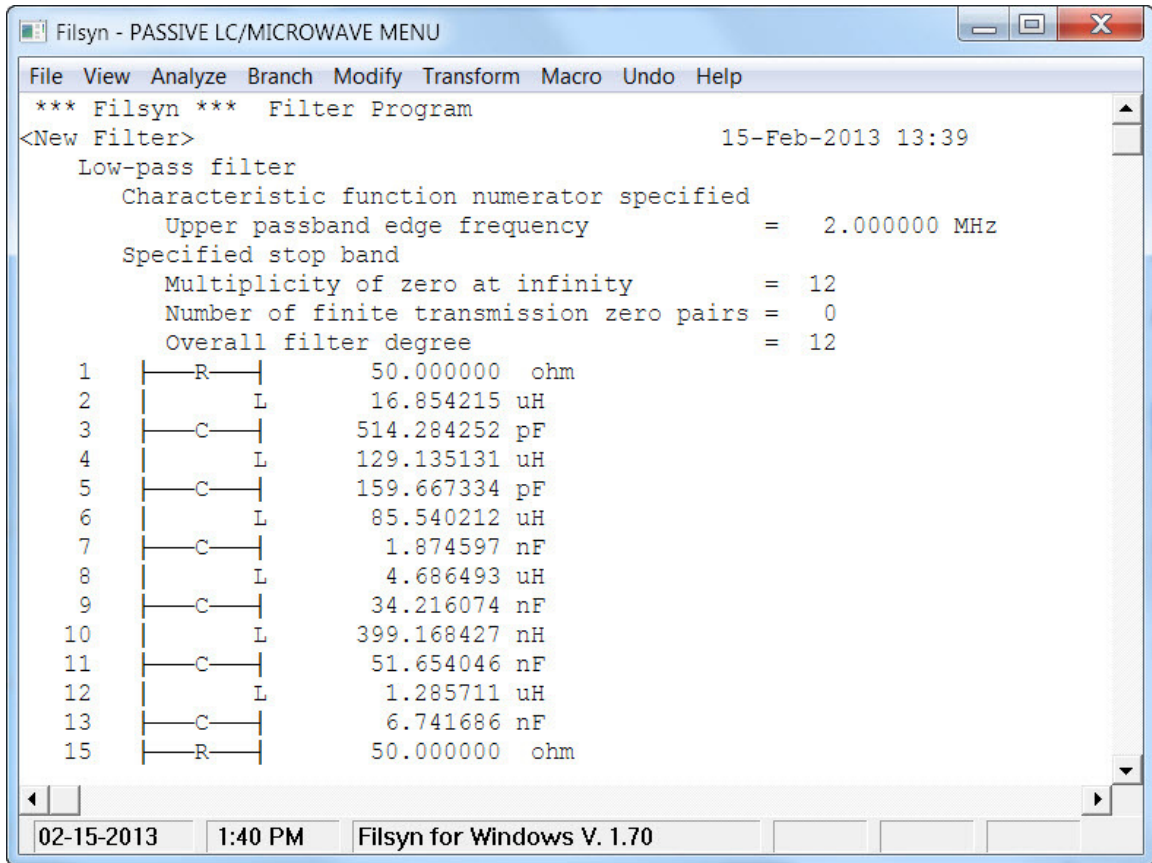
The new script file is under the name of **Misc. filters->Lowpass-like bandpass** in the **filscript.exe** executable script file.



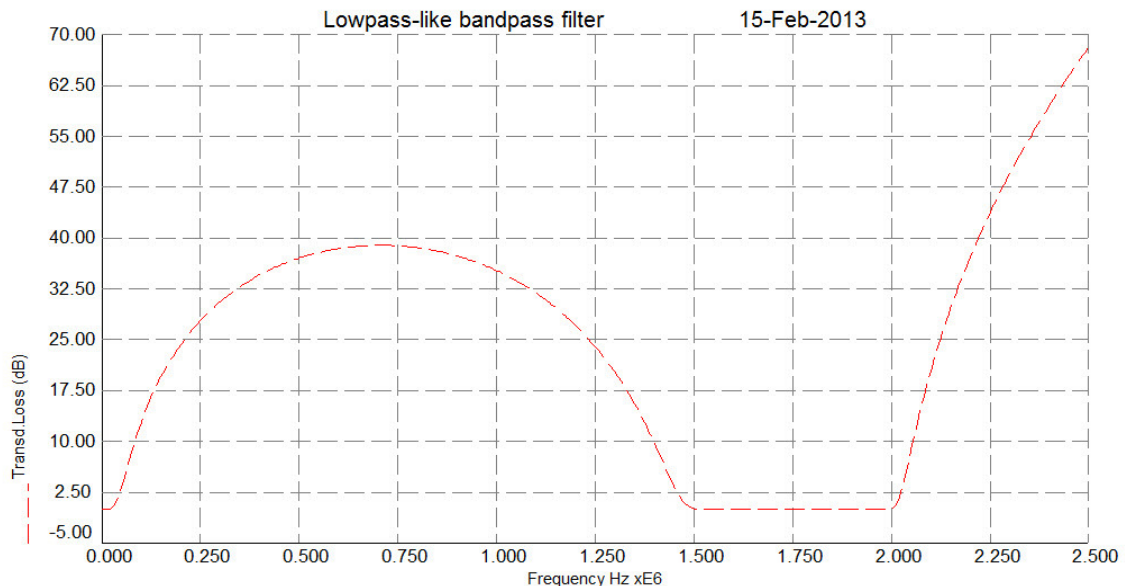
Selecting this option, the data entry window that comes up permits us to obtain extreme termination at one end:

Filter degree	12	Passband edge loss (dB)	0.10000000
Lower passband edge (Hz)	1.500000E+06	Input termination	50.000000
Upper passband edge (Hz)	2.000000E+06	Output termination	50.000000
Filter type	<input type="radio"/> Maximally flat <input checked="" type="radio"/> Equal ripple		
		OK	Cancel

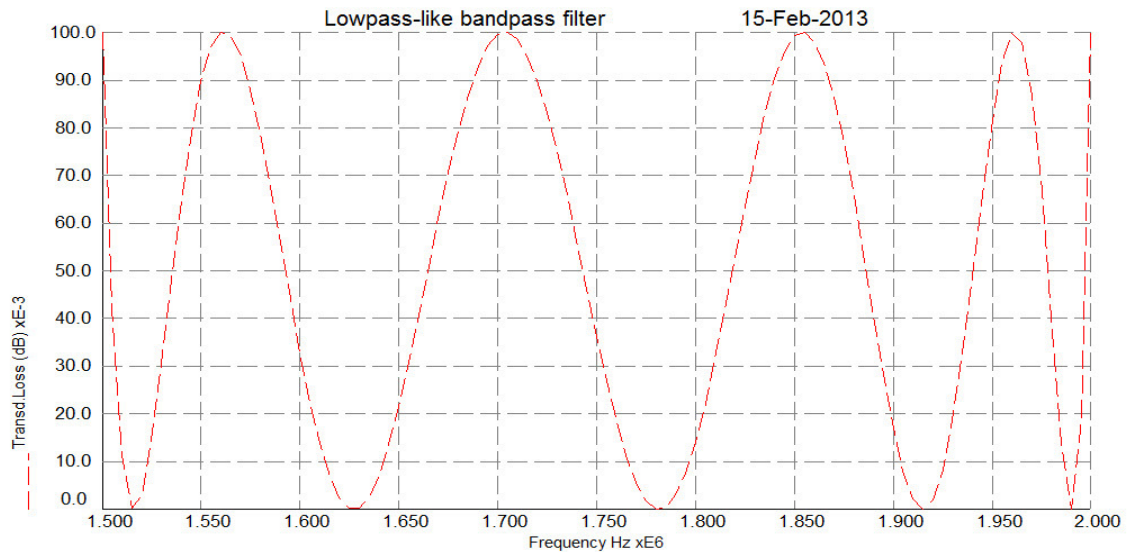
where we selected a 12th degree design with a 0.1 dB equal ripple loss in the passband from 1.5 MHz to 2 MHz. (Note that the global frequency unit is set to be Hz). Clicking on the **OK** button, we see a set of windows going by, specifying functional input and stopping at the end showing the filter circuit:



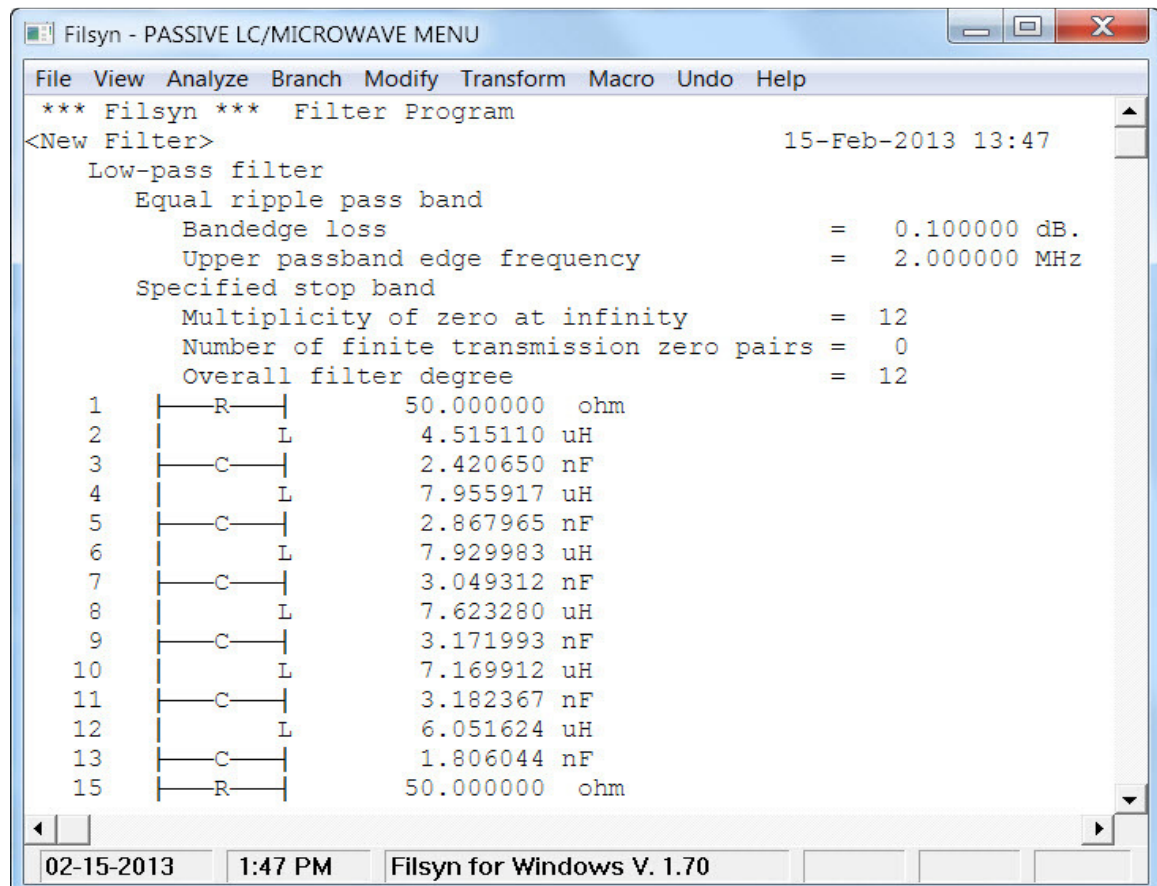
Analyzing this filter, we obtain the following loss plot:



Here we can read off the loss at 2.5 MHz as 68.01 dB. The passband details are shown next:



which indicates the accuracy of the procedure. The loss in the lower stopband may or may not be required. To compare this with a standard Chebyshev 12th order lowpass, we designed such a filter yielding the circuit:



As you can see, the element values are more desirable here, the element value spread being only 1.761, as opposed to the horrendous 324.46 for the design above. On the other hand, the loss of the standard Chebyshev design at 2.5 MHz is only 49.36 dB.

Comments:

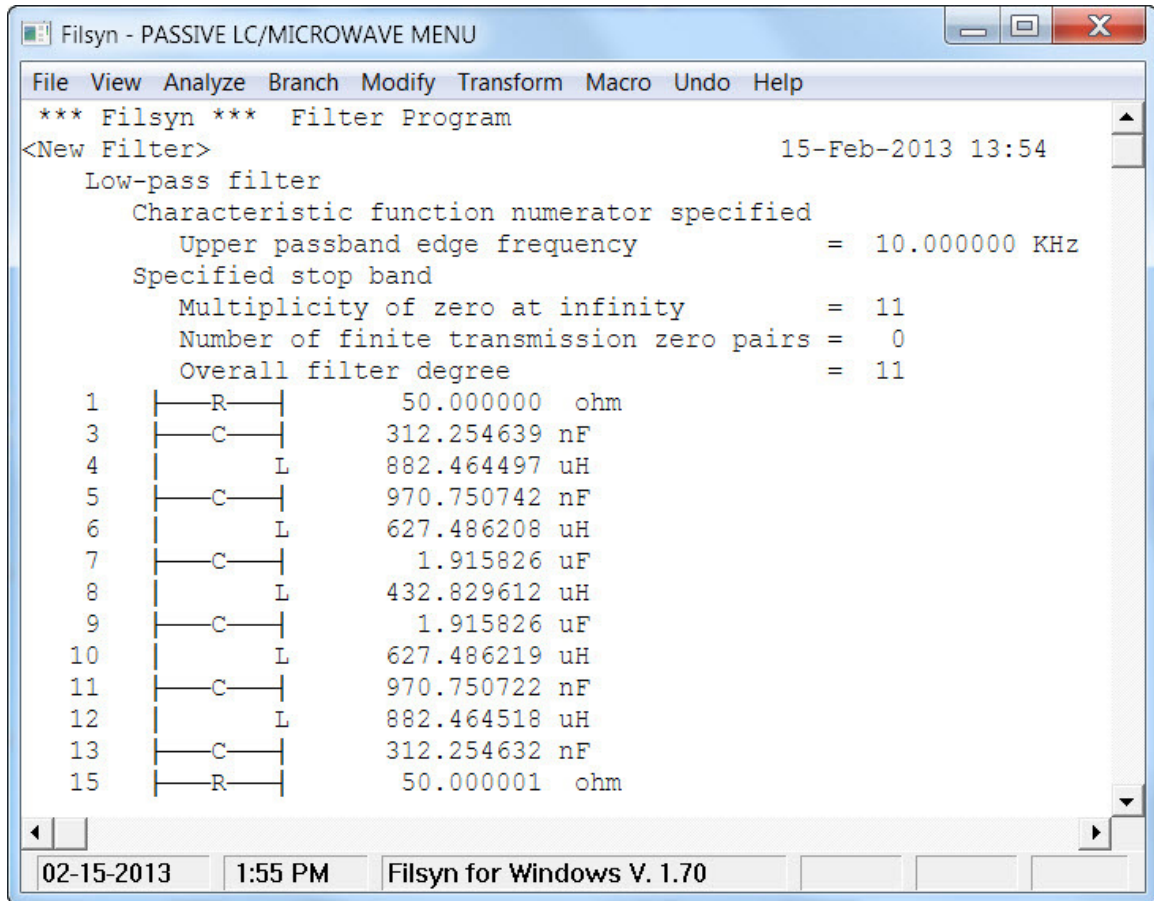
The analysis results on page 532 show a feature of these filters, the loss below the passband. If we widen the passband (reduce the value of $\alpha = f_a/f_b$) this loss is also reduced. In the limit this loss is reduced to the value of the passband ripple, in which case the filter degenerates to a simple Chebyshev type lowpass. Just before that happens, the program will print an advisory message to the user and stop the design.

5. Maximally-flat passband example.

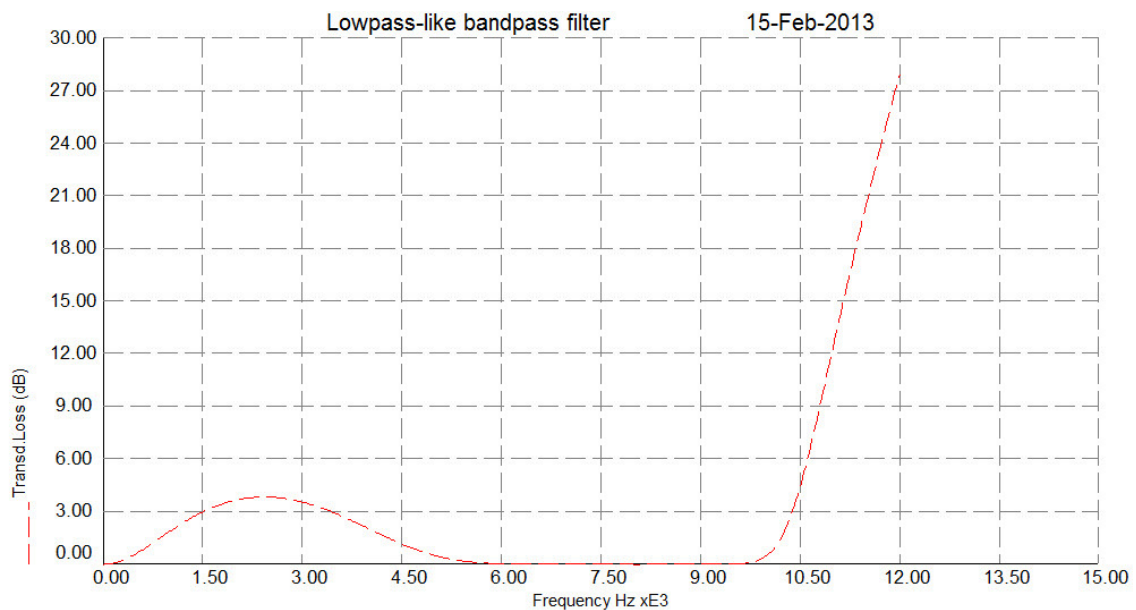
Consider a 11th order filter with passband from 5 KHz to 10 KHz and 0.5 dB loss at the band edges:

Filter degree	Passband edge loss (dB)
11	0.500000000
Lower passband edge (Hz)	Input termination
5.000000E+03	50.0000000
Upper passband edge (Hz)	Output termination
10.000000E+03	50.0000000
Filter type	
<input checked="" type="radio"/> Maximally flat <input type="radio"/> Equal ripple	
<input type="button" value="OK"/> <input type="button" value="Cancel"/>	

When the script file stops, we have the following filter – note the interesting fact that the circuit is completely symmetrical:



The analysis of the structure shows perfect agreement with expectations:



Comments:

As we increase the bandwidth in the case of maximally-flat design, the effect will be different from the equal-ripple case. The loss peak below the band will decrease initially, but there will be a point where the peak will show up *inside* the specified passband. Since that cannot be tolerated, the program detects this and after issuing an error message, stops the design.

While unequal terminations are seemingly acceptable, the program only accepts $R_2 = 0.0$ as an option. Otherwise, if R_2 is not equal to R_1 , it will be made equal. This way we can design filters that can be paralleled

APPLICATION NOTE 11

SINGLE SIDEBAND CRYSTAL FILTERS

1. Introduction

Very narrow band ladder filters may be constructed to contain only piezoelectric crystals and capacitors. In order to obtain the proper structures, we have created a pair of scripts inside the **filscript.exe** executable file. These are the **USB crystal filter** and **LSB crystal filter** under the **Misc(ellaneous) filters** heading.

2. USB filter

Consider a 600 Hz wide passband from 1 MHz to 1.0006 MHz and a stopband from 2 KHz below the passband to 300 Hz below and hoping to get about 70 dB loss using four crystals. The specification is entered as:

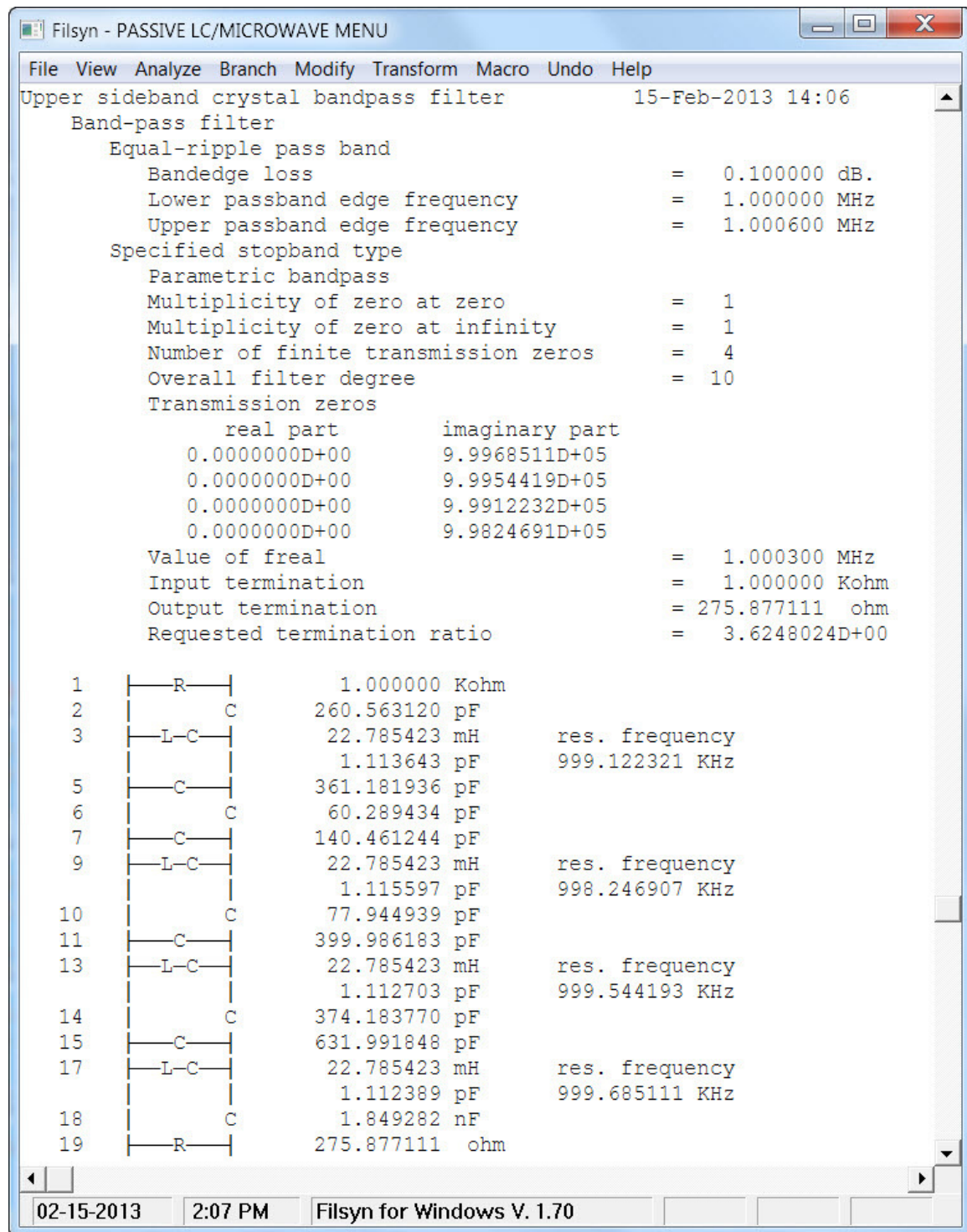
Upper sideband crystal filter

Lower passband edge (MHz)	Lower end of lower stopband (MHz)
1.000000E+00	998.000000E-03
Upper passband edge (MHz)	Upper end of lower stopband (MHz)
1.000600E+00	999.700000E-03
Passband loss ripple (dB)	
0.100000000	
Number of crystals	
4	
Stopband type	
<input checked="" type="radio"/> Equal minima	
<input type="radio"/> Specified	

	Transm. zeros (MHz)
1	0.00000
2	0.00000
3	0.00000
4	0.00000
5	0.00000
6	0.00000
7	0.00000
8	0.00000

OK Cancel

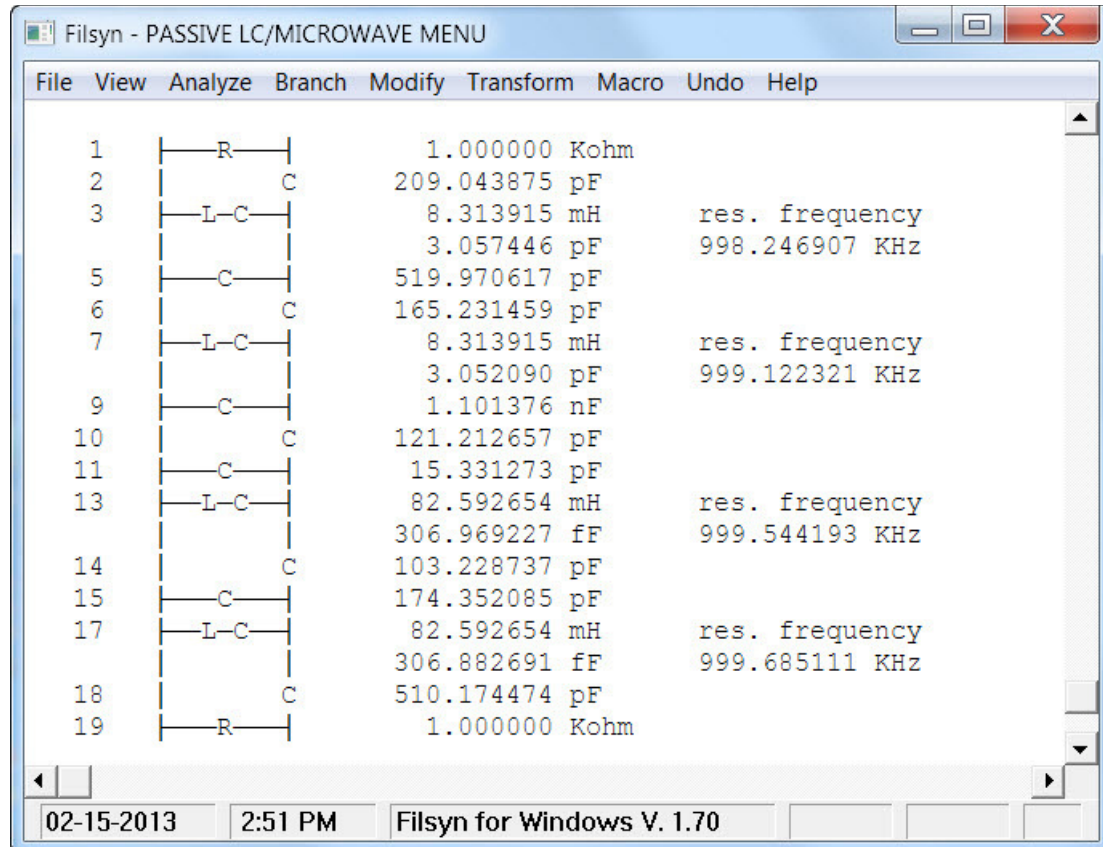
This is entered into the **filscript.exe** program and while the process does not stop here, we briefly explain what the program has done so far.



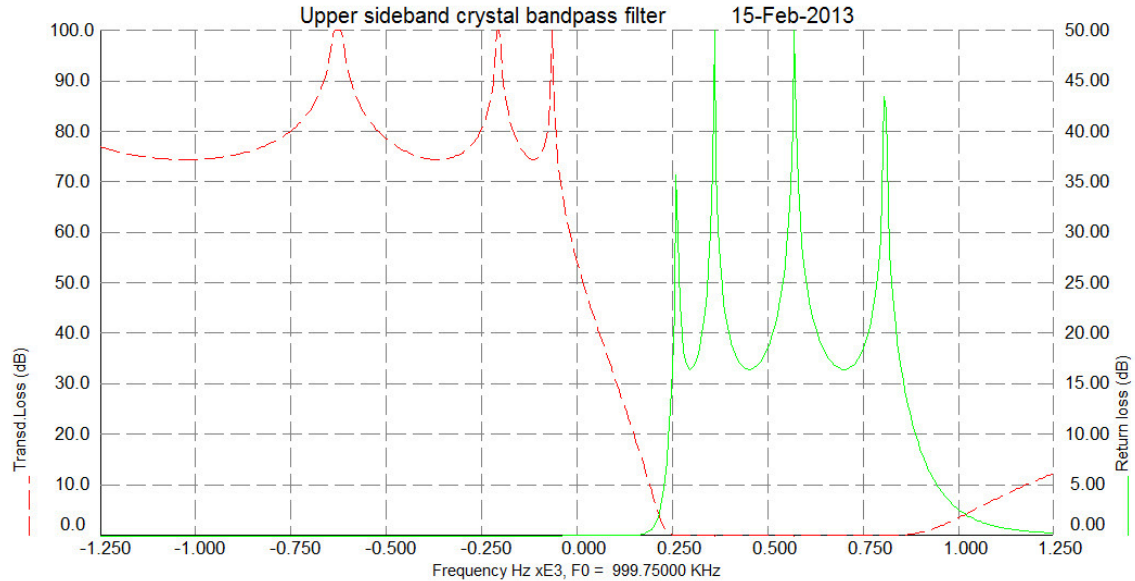
It first called the **Placer** segment of the program to optimize the location of the transmission zeros using the parametric bandpass case. Then it has generated a standard ladder circuit containing shunt resonant LC circuits to implement the four transmission zeros and a few capacitors. Finally it used a number of common ladder conversion steps,

including various Norton transformation steps to make all the inductors to have the same value yielding the circuit above.

All crystals are now essentially identical, and all have shunt capacitors across, essential to crystal implementation. The so-called capacitance ratio -- the ratio of the shunt C to the resonating C -- is quite acceptable. The lowest value is 126 of the second crystal. This can be modified by rearranging the sequence of the resonant branches. For instance by interchanging the first two resonant circuits and doing some other minor modifications, we have reached the circuit here:



This is a much more convenient set of values even if the inductors are not all equal. The terminations are equal and the smallest capacitance ratio is 170. Further trial and error steps are also possible. The analysis in the frequency domain also shows a very satisfactory result:



Comment: As the data input screen above shows, we can alternatively specify the locations of the transmission zeros if we so desire. This is useful if we also wish to make all crystals resonate at the same frequency. Since this may cause some minor numerical instability, it is advisable to use frequencies that are not exactly equal but differ by a very small amount.

3. LSB filter

This filter type is also described in detail in *Application Note 3* above, except that this filter has no inductors at the two ends, only crystals and capacitors. The parametric design is very similar though and the advantage of this script is that the procedure is completely automated. The example we shall use to illustrate this design is essentially a flipped around version of the USB example above, except that we shall specify the four transmission zeros to be, essentially but not strictly, identical.

Note also, that as in the filters of *Application Note 3*, the required capacitance ratio must be specified, due to the nature of the configuration.

Lower sideband crystal filter

Lower passband edge (MHz): 999.400000E-03

Upper passband edge (MHz): 1.000000E+00

Passband loss ripple (dB): 0.100000000

Number of crystals: 4

Required capacitance ratio: 150.000000

Stopband type:
☐ Equal minima
☒ Specified

Upper end of upper stopband (MHz): 0.00000

Lower end of upper stopband (MHz): 0.00000

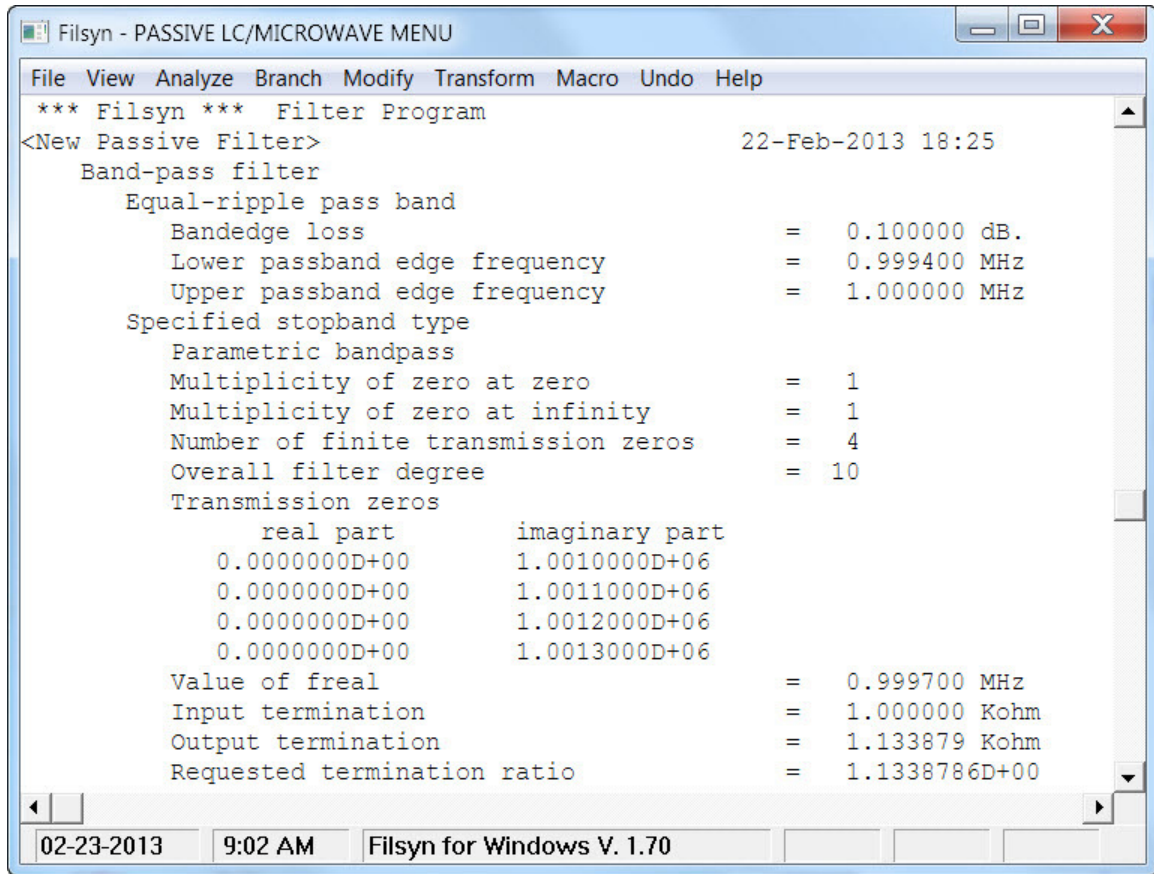
	Transm. zeros (MHz)
1	1.001000E+00
2	1.001100E+00
3	1.001200E+00
4	1.0013
5	0.00000
6	0.00000
7	0.00000
8	0.00000

OK Cancel

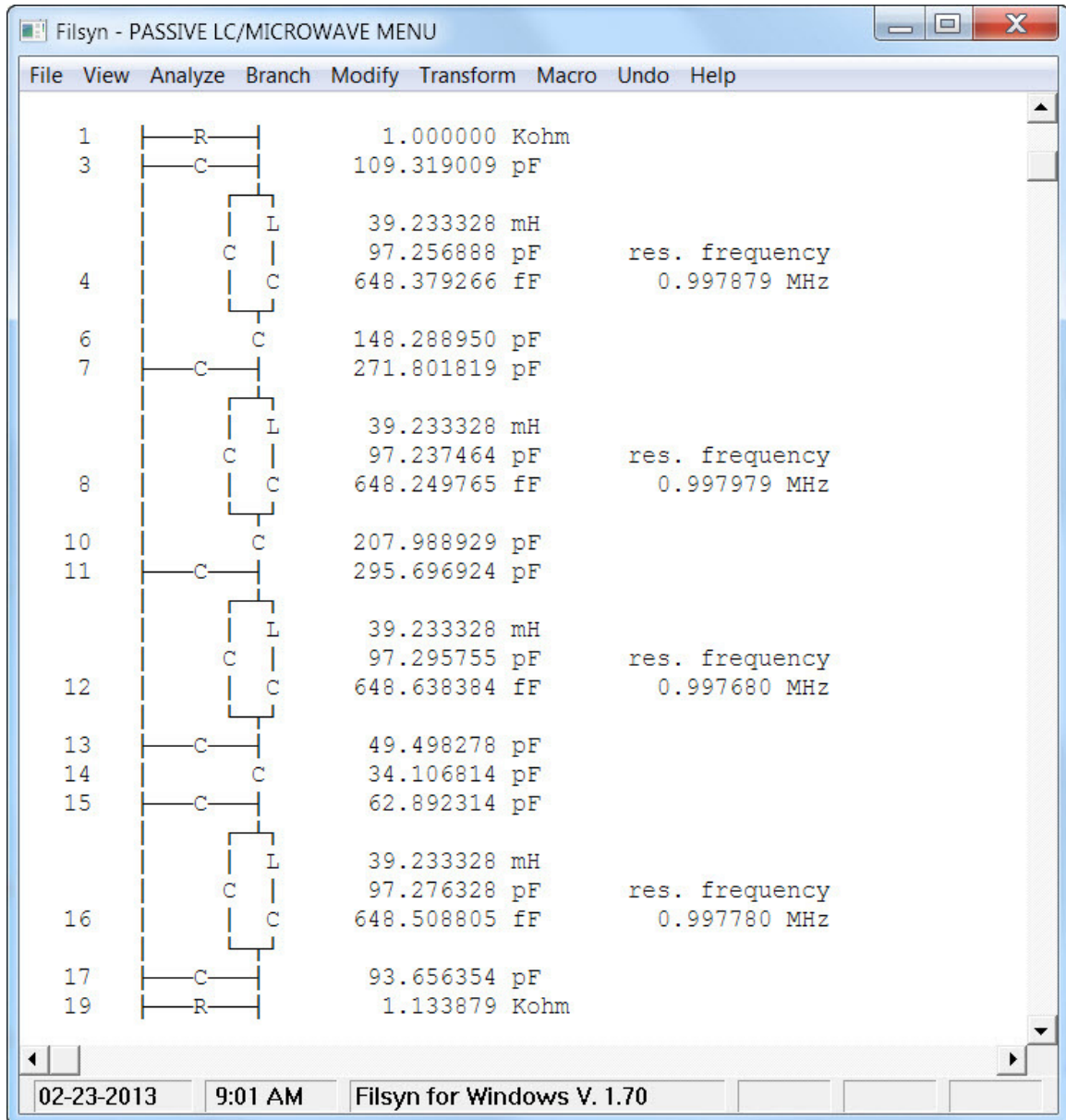
After the design process stops, we have the summary of the filter but first we again describe briefly what is happening.

The synthesis is automatic and results in a ladder containing four parallel resonators in the series branches to implement the specified transmission zeros, plus a few capacitors. One of these resonators will have a capacitor in series. This capacitor will be split and one part, together with the resonator, will be converted to the equivalent 3 element branch that is the electrical equivalent of a crystal -- series LC branch with a parallel capacitor -- with the requested capacitance ratio. This process is then repeated for the other three resonators.

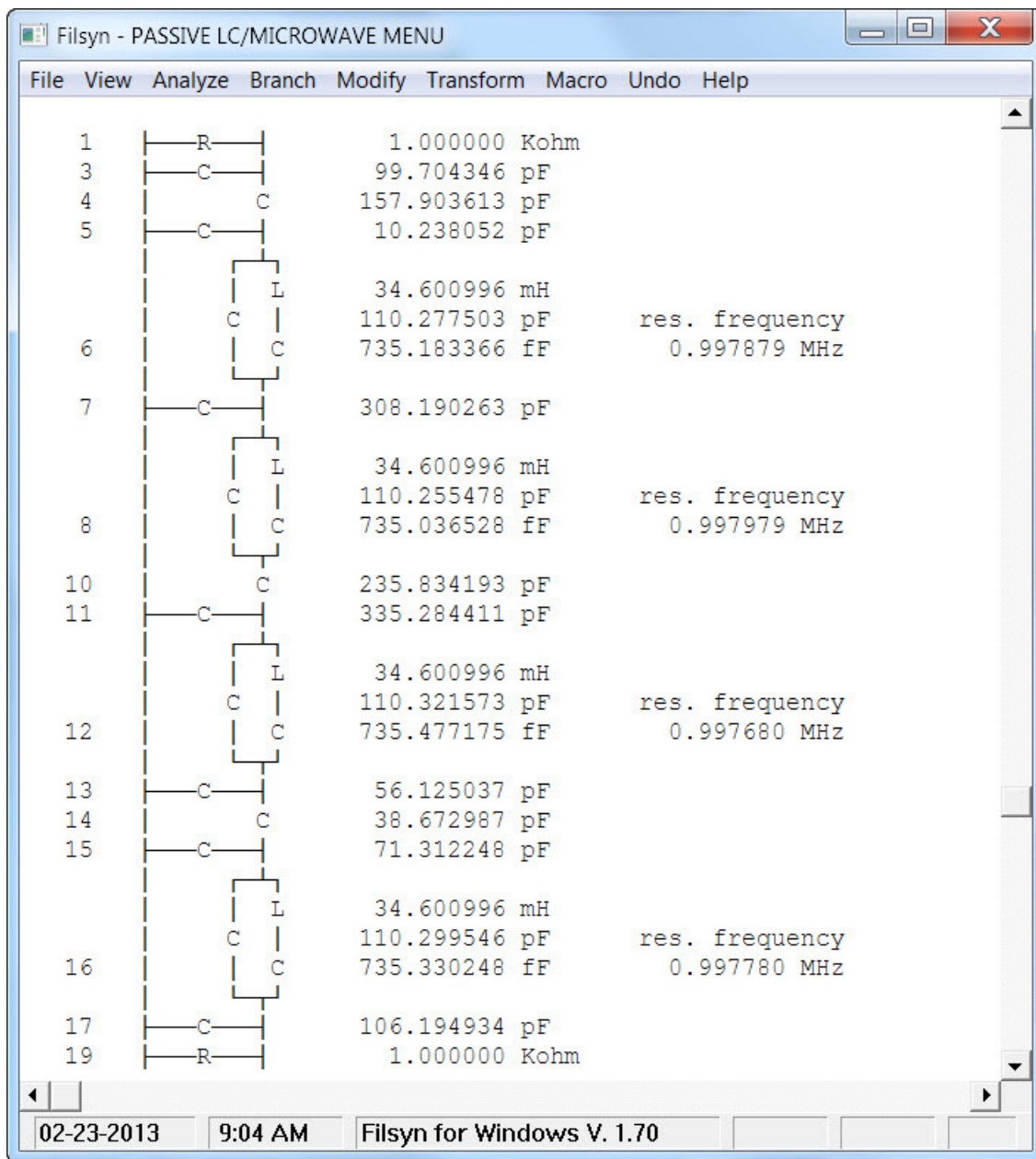
Finally we use the Norton transformation procedure again to make all four inductors identical. The resulting program output is shown in two parts:



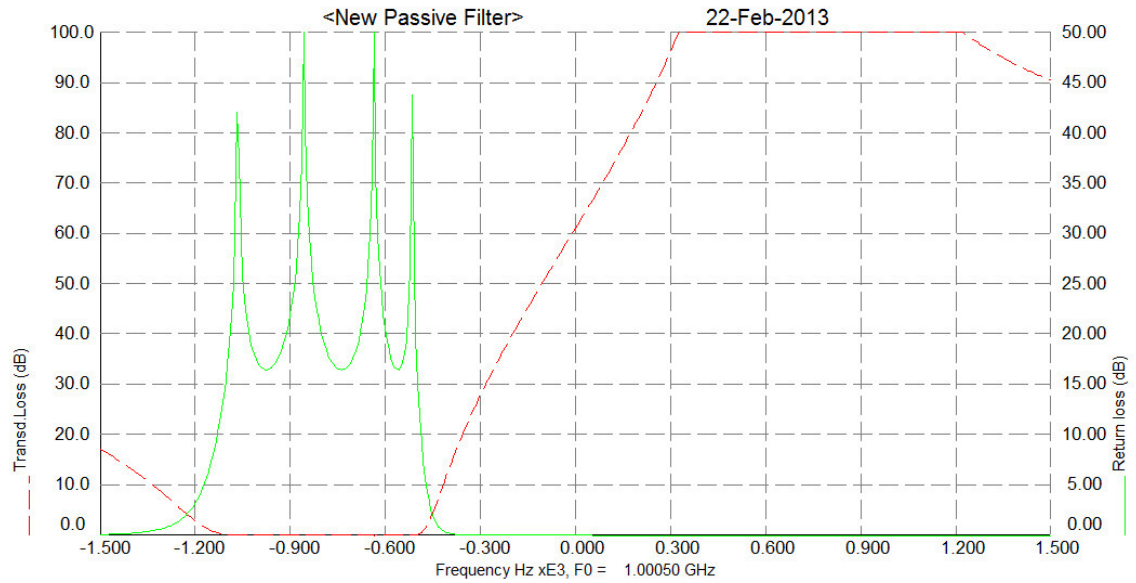
The second half contains the circuit details:



As can be seen, the crystals all have the required capacitance ratio of 150 and they all have identical motional inductances and very close, although not identical, resonant frequencies. In this particular case, the unequal terminations cause no problem, one step of simple Norton transformation, using C_3 and C_6 , solves it:



Finally the frequency domain analysis indicates excellent performance:



Please note that for plotting purposes loss values over 100 dB are replaced by 100 dB, but the tabulated results are not changed.

APPLICATION NOTE 12

MINIMUM-PHASE FIR FILTERS

1. Introduction

Most FIR filters we design are of the linear-phase variety, that have most of their zeros on the unit circle, while the rest are divided; half of them are outside the unit circle, the other half inside. Occasionally, we need to have a filter with *all* zeros not on the unit circle to be *inside*. These filters have much less phase shift, hence the name, but the loss is completely unaffected.

Several ways are described in the literature for converting any FIR filter into a minimum-phase form, but the simplest and most straightforward is to

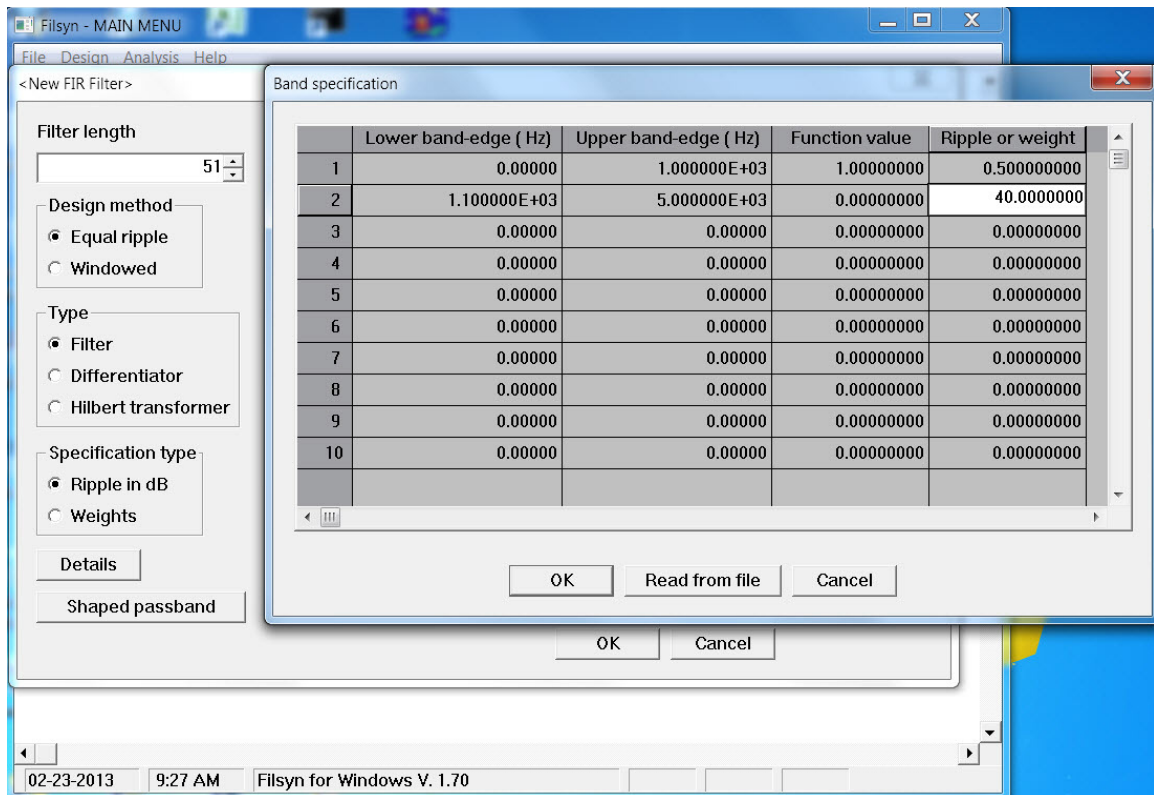
- compute all its zeros
- shift the zeros outside the unit circle inside, by inverting them
- multiplying all the factors together again
- scaling the function to have zero loss in the passband.

We have separate menu items that perform these operations individually, but in order to simplify the procedure, we introduced the **Minimum phase** menu item, that performs all of these operations in sequence at once.

Furthermore, we have rewritten the part of the program that performs the factoring of the function, the first and most critical step in the procedure. Currently the program can handle FIR filters of 512 taps and the procedure has been tested up to 501 taps with excellent results. The program can handle single and double roots; higher order roots (extremely unlikely) have not been tried and probably would not work.

2. Example

Consider a 51 tap equal-ripple type lowpass filter, with 10 KHz sampling frequency, passband to 1 KHz and stopband starting at 1.1 KHz. This filter is specified in **Filsyn** as:



To enter the band specification shown above we need to click on the **Details** button. The resulting filter is shown next:

The screenshot shows a Windows application window titled "Filsyn - FIR DIGITAL MENU". The menu bar includes File, View, Analyze, Modify, Transform, Undo, and Help. The main text area displays the following information:

```

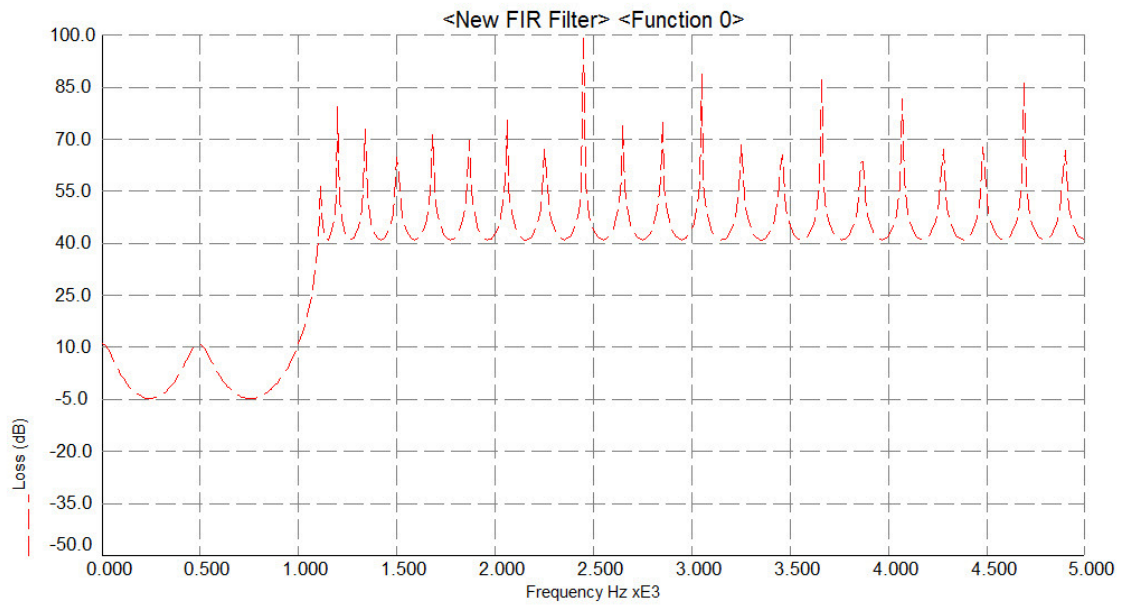
Finite impulse response (FIR)
Linear phase digital filter design
<New FIR Filter>                                     23-Feb-2013   9:28
Remez exchange algorithm
Bandpass filter
Filter length = 51
***** Impulse response *****
Function no. 0
Sampling frequency = 1.0000000D+04
decimal                                         hexadecimal
h( 1) = -1.7814889D-03 = h( 51) = -0.0074c06cc
h( 2) = -1.4988167D-02 = h( 50) = -0.03d643b88
h( 3) = -2.5013904D-02 = h( 49) = -0.06674faad
h( 4) = -4.0662162D-02 = h( 48) = -0.0a68d5e06
h( 5) = -5.6151177D-02 = h( 47) = -0.0e5fec6c9
h( 6) = -6.8349628D-02 = h( 46) = -0.117f5c786
h( 7) = -7.3450829D-02 = h( 45) = -0.12cdac6e6
h( 8) = -6.8726008D-02 = h( 44) = -0.119807134
h( 9) = -5.3499544D-02 = h( 43) = -0.0db22569b
h(10) = -2.9798920D-02 = h( 42) = -0.07a0e6eaa
h(11) = -2.2619436D-03 = h( 41) = -0.00943d1df
h(12) = 2.2802388D-02 = h( 40) = 0.05d660965
h(13) = 3.9029745D-02 = h( 39) = 0.09fdda77c
h(14) = 4.1881698D-02 = h( 38) = 0.0ab8c24b6
h(15) = 3.0260200D-02 = h( 37) = 0.07bf21eaa
h(16) = 7.2819090D-03 = h( 36) = 0.01dd3a28e
h(17) = -2.0105822D-02 = h( 35) = -0.0525a7b64
h(18) = -4.2705131D-02 = h( 34) = -0.0aeeb935a
h(19) = -5.1485606D-02 = h( 33) = -0.0d2e29213
h(20) = -4.0189434D-02 = h( 32) = -0.0a49dacfd
h(21) = -7.3866746D-03 = h( 31) = -0.01e417d60
h(22) = 4.2698353D-02 = h( 30) = 0.0aee477eb
h(23) = 1.0086761D-01 = h( 29) = 0.19d275a5e
h(24) = 1.5508007D-01 = h( 28) = 0.27b353c85
h(25) = 1.9344292D-01 = h( 27) = 0.318579a06
h(26) = 2.0728784D-01 = h( 26) = 0.3510d0efc

band 1      band 2
Lower band edge  0.000000000  0.110000000
Upper band edge  0.100000000  0.500000000
Desired value    1.000000000  0.000000000
Weighting        0.500000000  40.000000000
Deviation        0.719135284  0.008989191
Deviation in db  15.736256599 -40.925590515

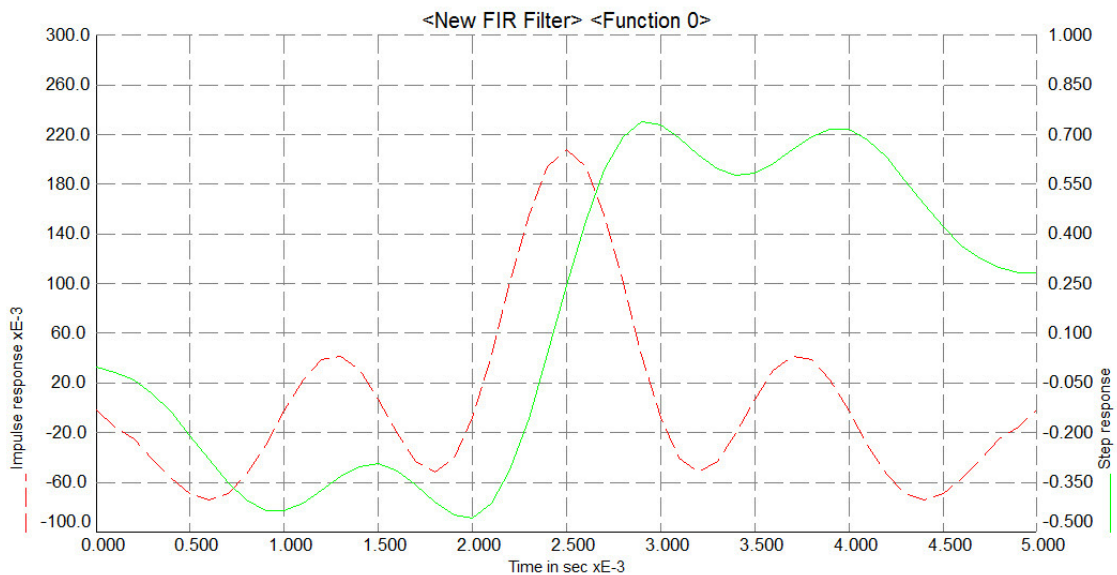
```

The status bar at the bottom shows the date 02-23-2013, time 9:29 AM, and version Filsyn for Windows V. 1.70.

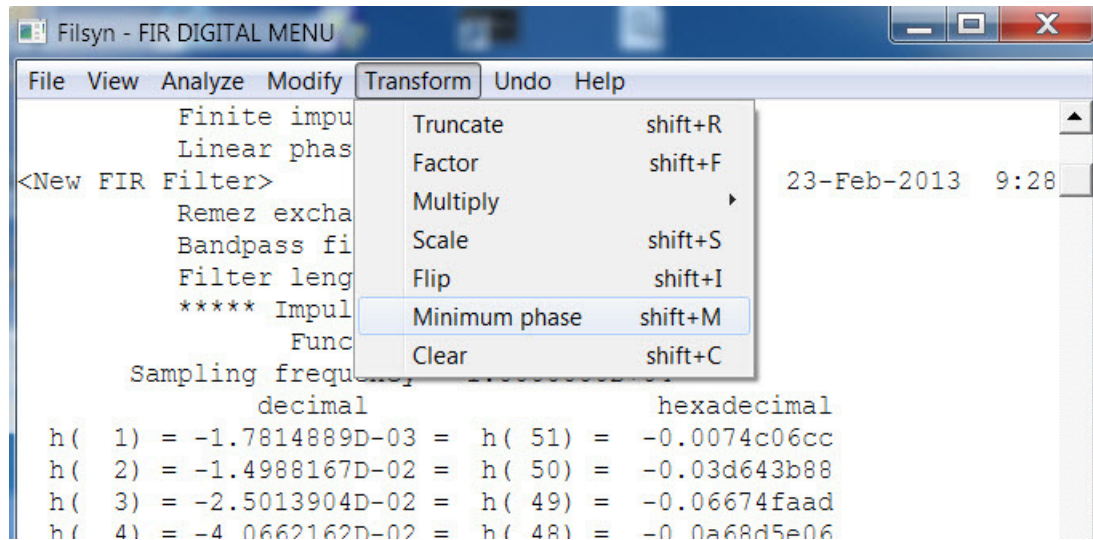
The computed loss behavior is as expected:



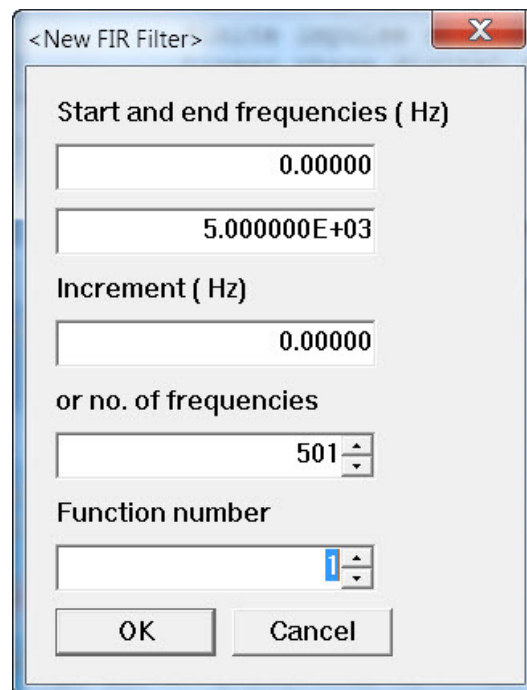
while the time domain behavior is as follows:

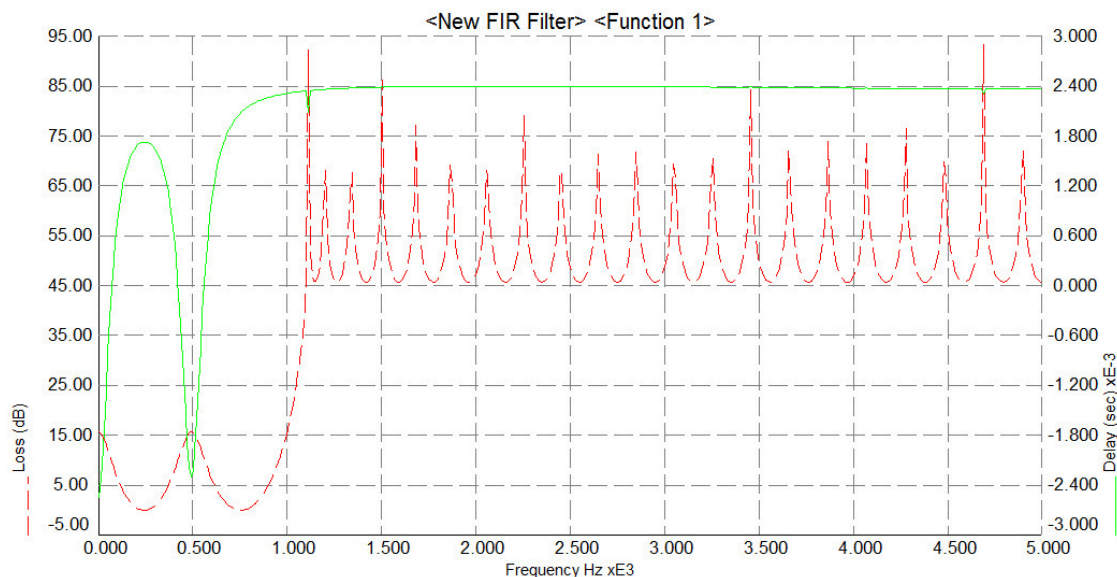


Next we show the menus under the **Transform** header which contains the **Minimum phase** option we need and call next:

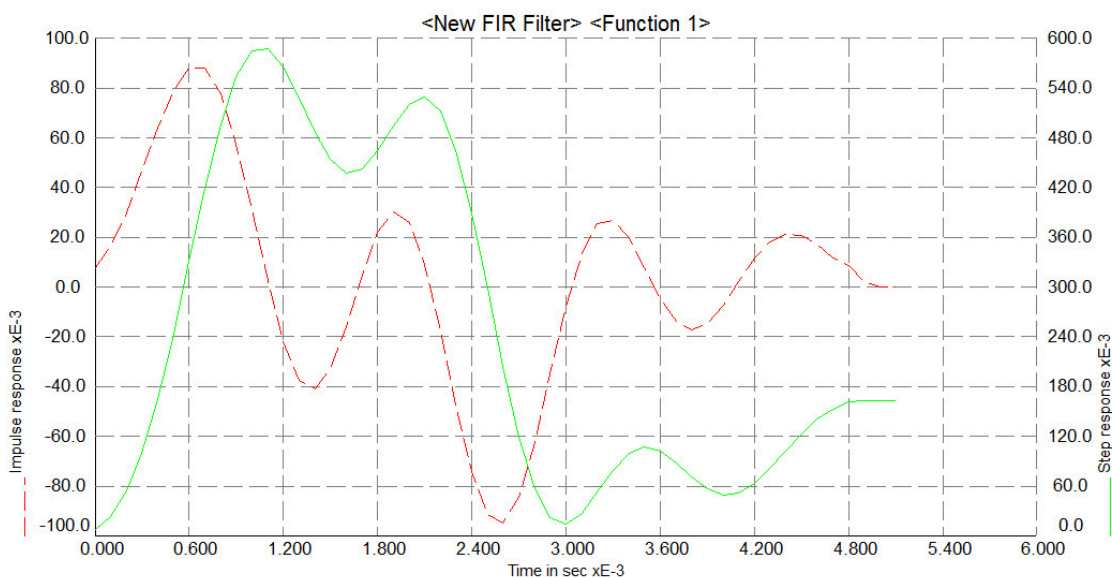


The results are nearly instantaneous and yield a filter with computed loss and delay responses as follows. Note that *Function 1* must be specified here for the analysis:





The loss is clearly identical to the one obtained before, but the delay is quite different. The delay was not displayed before, since it was exactly constant.



Please note that older versions of the program has a **Square root** menu item, that was essentially performing the same operation as this new **Minimum phase** does, but it was complicated to use, limited in scope and unreliable, hence it was deleted.

APPLICATION NOTE 13

COUPLED SHUNT RESONATORS

1. Introduction

A large number of LC bandpass filters are designed in the form of shunt parallel resonators coupled by single and/or double element branches. We also wish to have all the resonators to have identical inductors or sometimes identical capacitors.

If the filters have only transmission zeros at zero and infinite frequencies, then all coupling branches will be single element ones and the design of such structures is straightforward. Consider a bandpass with 3 zeros at zero frequency and nine zeros at infinity. This will yield a filter with closely arithmetical symmetry of its loss versus frequency characteristic.

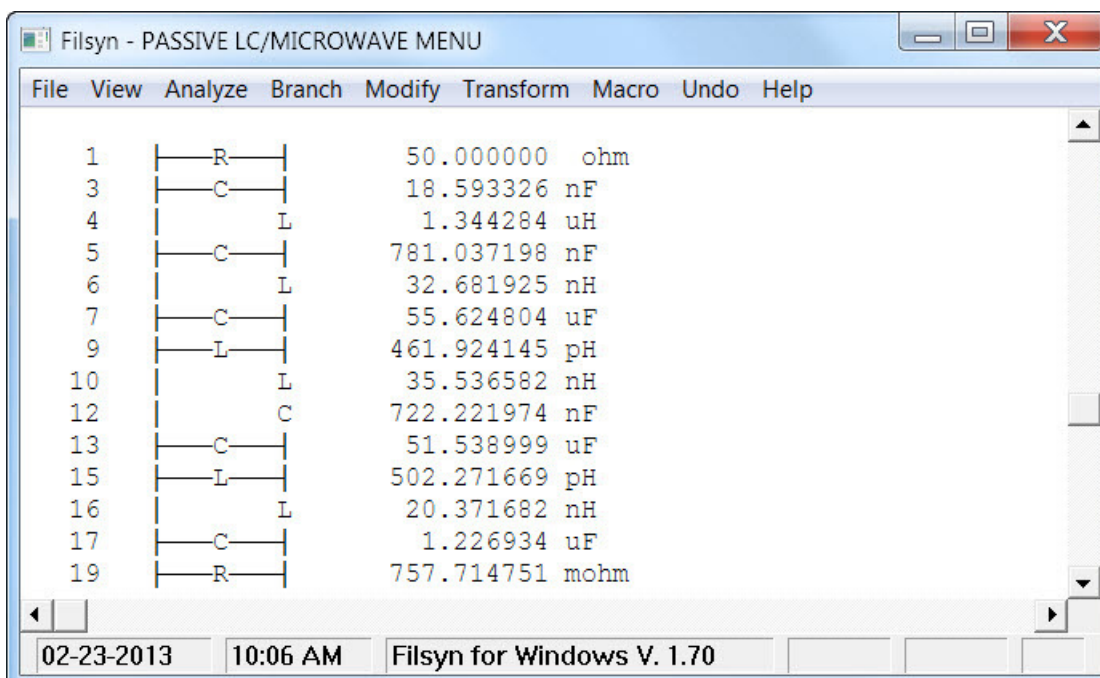
2. Example 1

The design data is entered as:

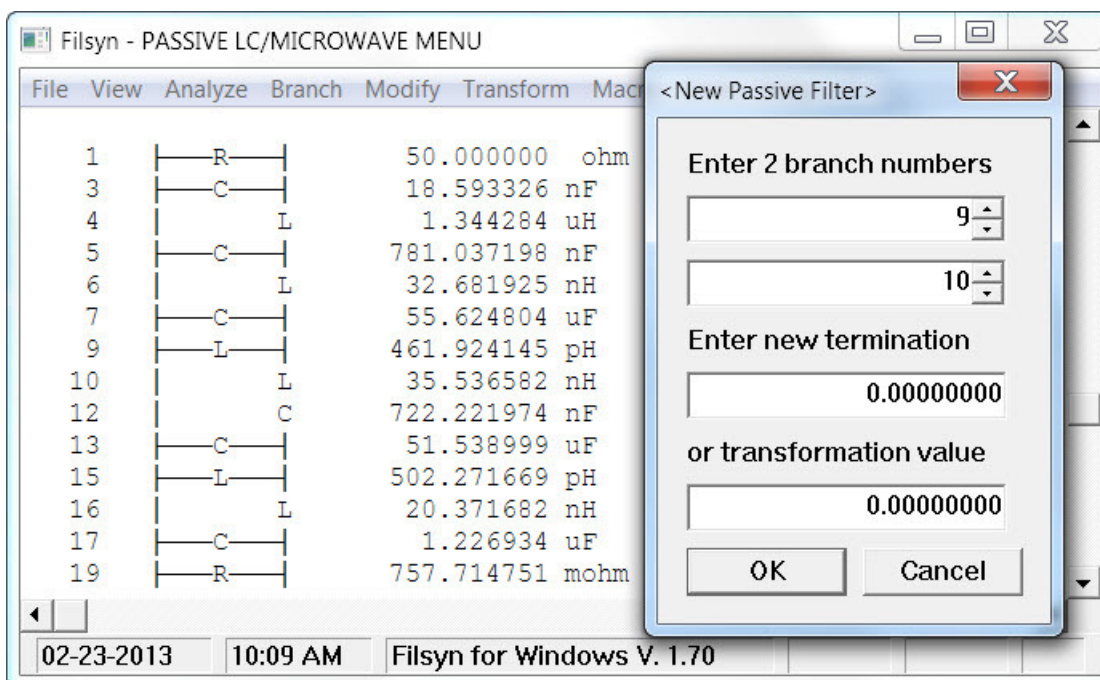
The screenshot shows the 'New Filter' dialog box with the following settings:

- Filter kind:** LC
- Filter type:** Band pass
- Lower passband freq (Hz):** 900.000000E+03
- Upper passband freq (Hz):** 1.100000E+06
- Passband type:** Equal ripple
- Band-edge loss/return loss (dB):** 0.100000000
- Loss Slope (dB/oct):** 6.00000000
- Flat loss (dB):** 0.00000000
- Function Type:** E
- Multiplier:** 0.00000000
- Stopband:** Specified
- Lower stopband freq (Hz):** 0.000000
- Loss (dB):** 50.0000000
- Upper stopband freq (Hz):** 0.000000
- Loss (dB):** 50.0000000
- # of zeros at zero:** 3
- # of zeros at infinity:** 9
- # of unit elements:** 0
- Parametric:** Conventional
- R1:** 50.000000E+00
- R2:** 50.000000E+00
- ZS parameter:** -1
- Q for predistortion:** 0.00000000
- Predistortion:** Passband only
- Shifted bandpass trans. Center frequency (Hz):** 0.000000
- Odd parametric:** ☐
- Save design data:** ☐

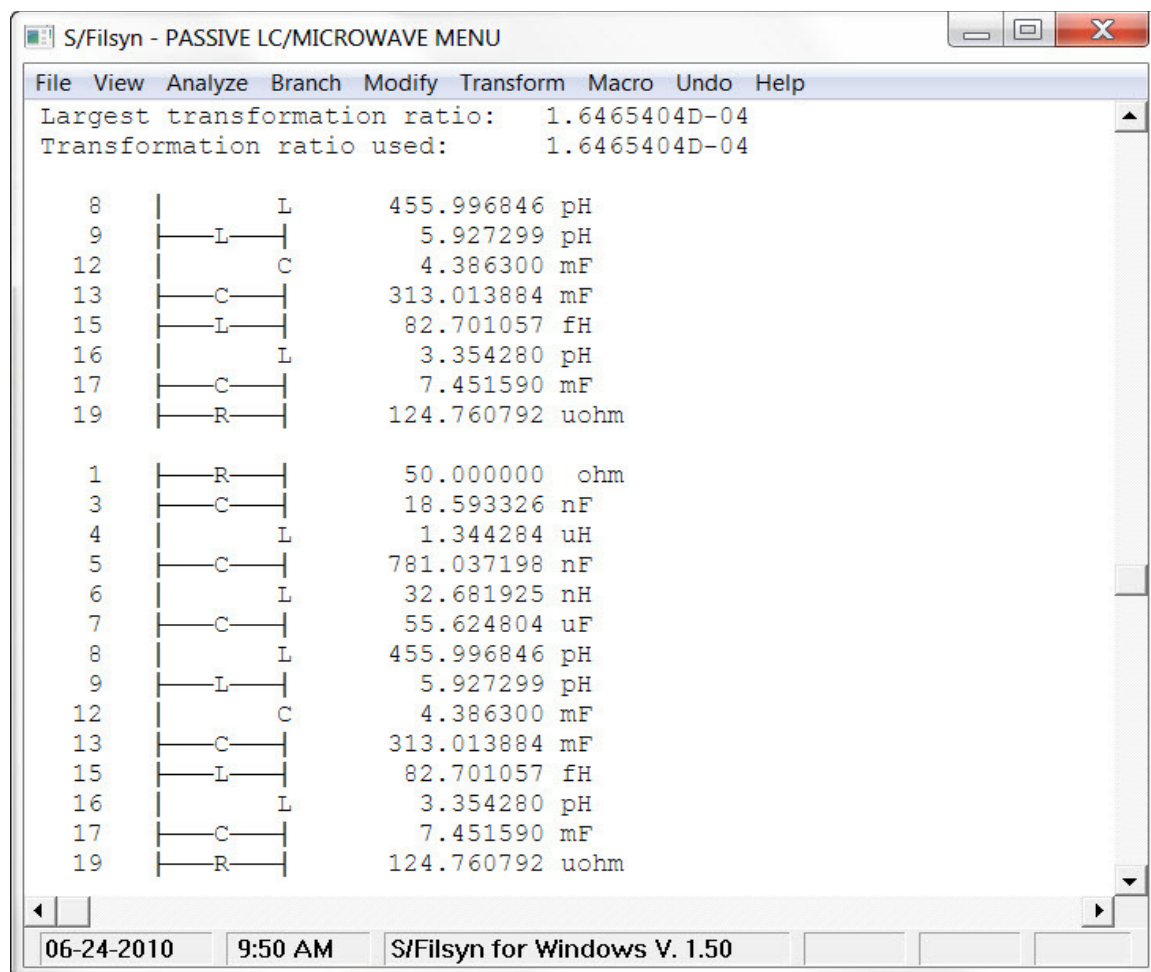
The resulting filter in the computer-generated form is not the one we need and therefore using the **Transform -> Dual** menu item we, get its dual:



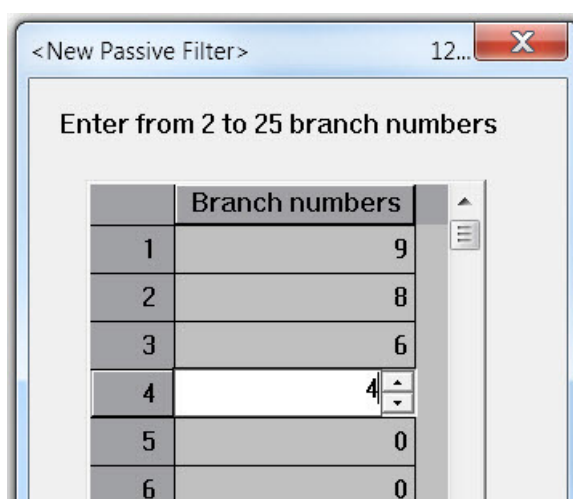
In order to get a resonator in every shunt branch, we need to first shift all shunt inductors into a single branch at one end of this structure, except the capacitor C_{12} , that splits the configuration into two separate segments. Hence we just shift L_9 to the other side of L_{10} ; This is done using the **Modify -> Norton -> Simple** menu item:



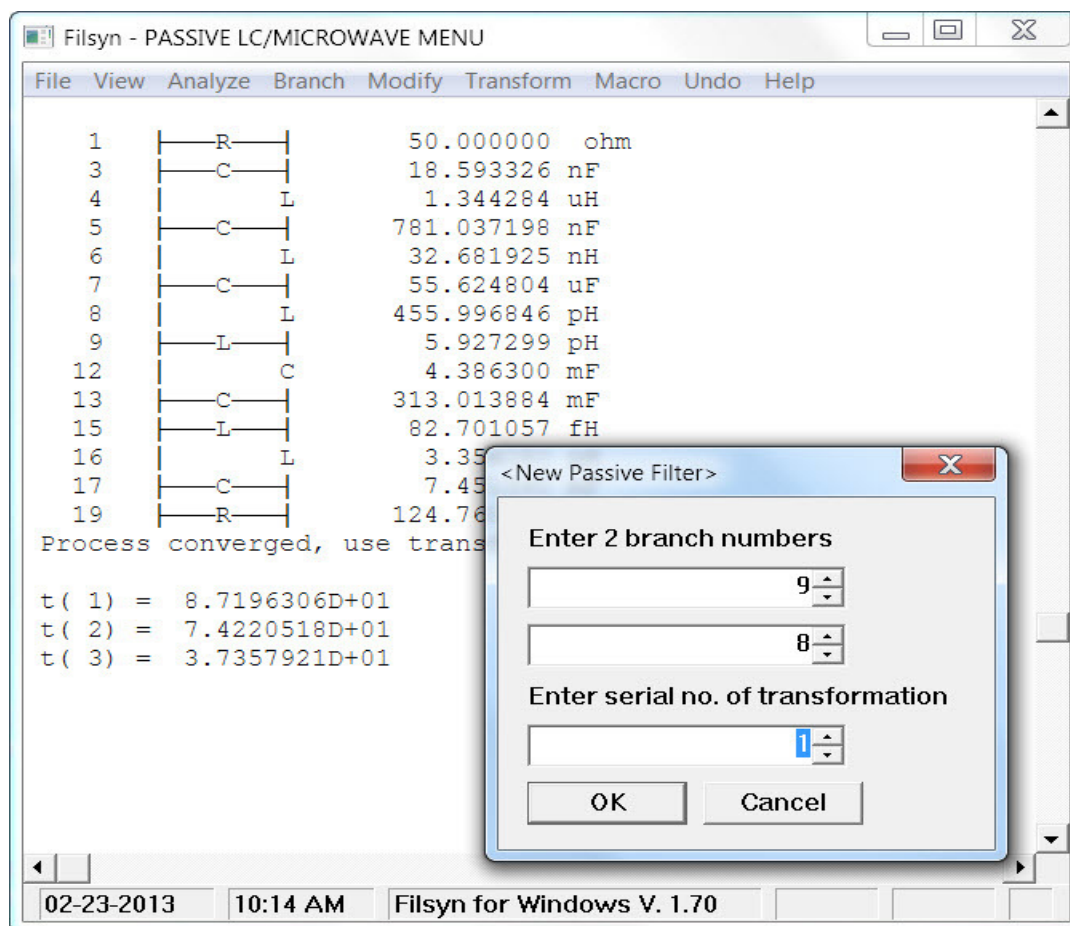
No other data is necessary. This yields:



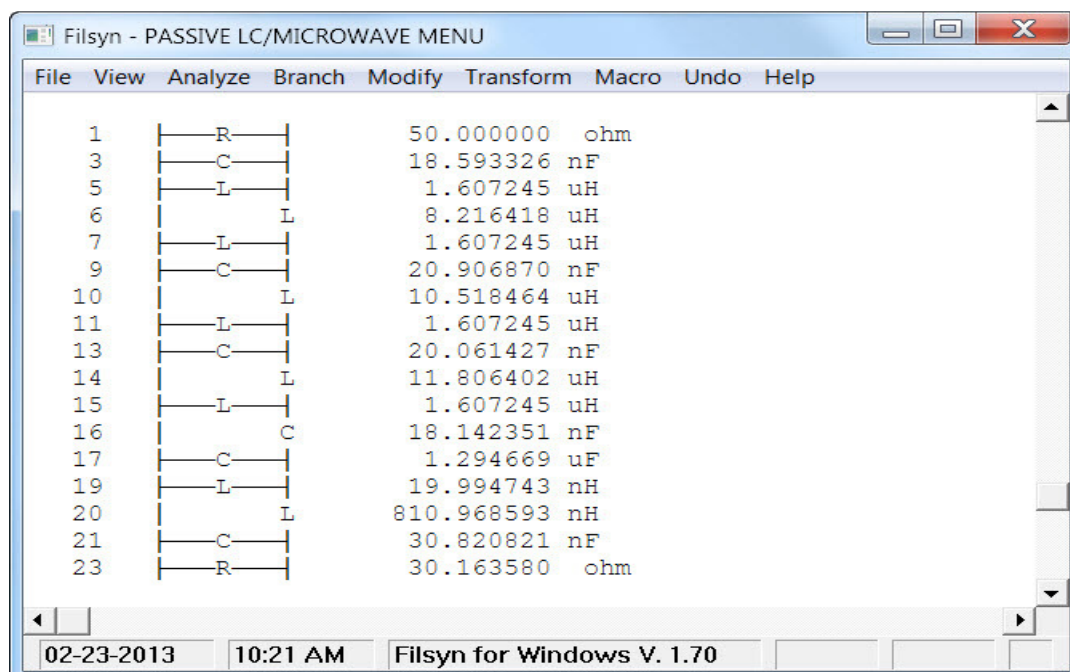
Next we distribute L_9 into L's parallel to C_7 , C_5 and C_3 with all having equal values. This needs the use of the **Modify -> Norton -> All -> All** menu item:



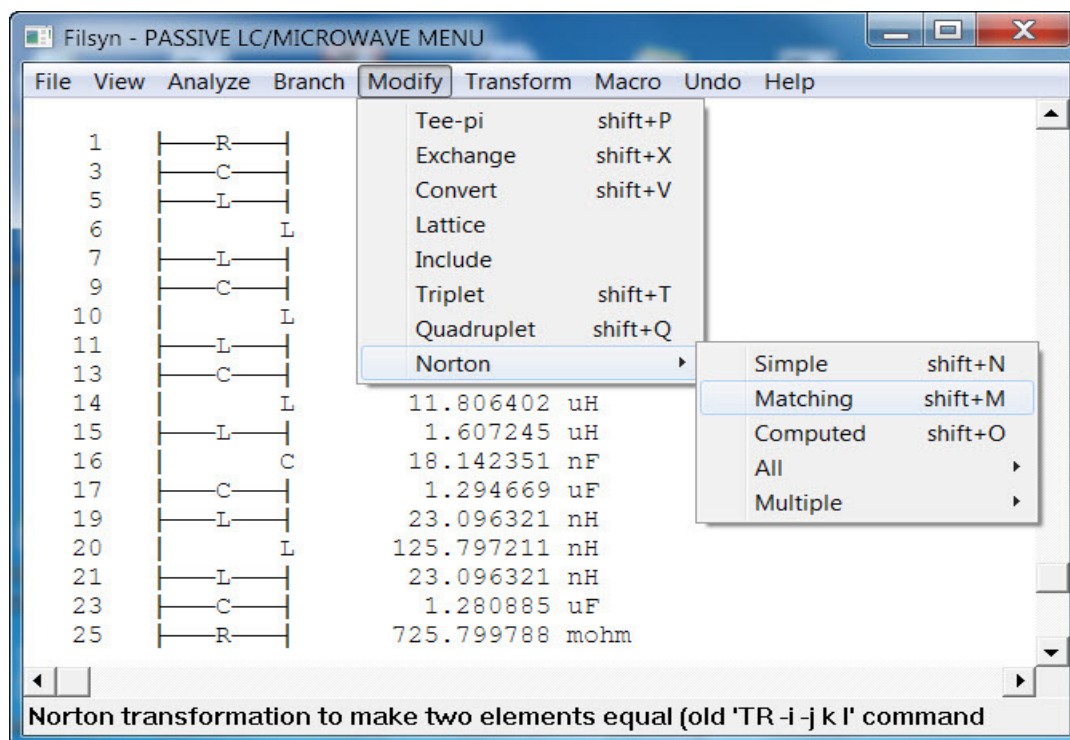
This command just computes the necessary transformation ratios, which we have to use next in the **Modify -> Norton -> Computed** menu:



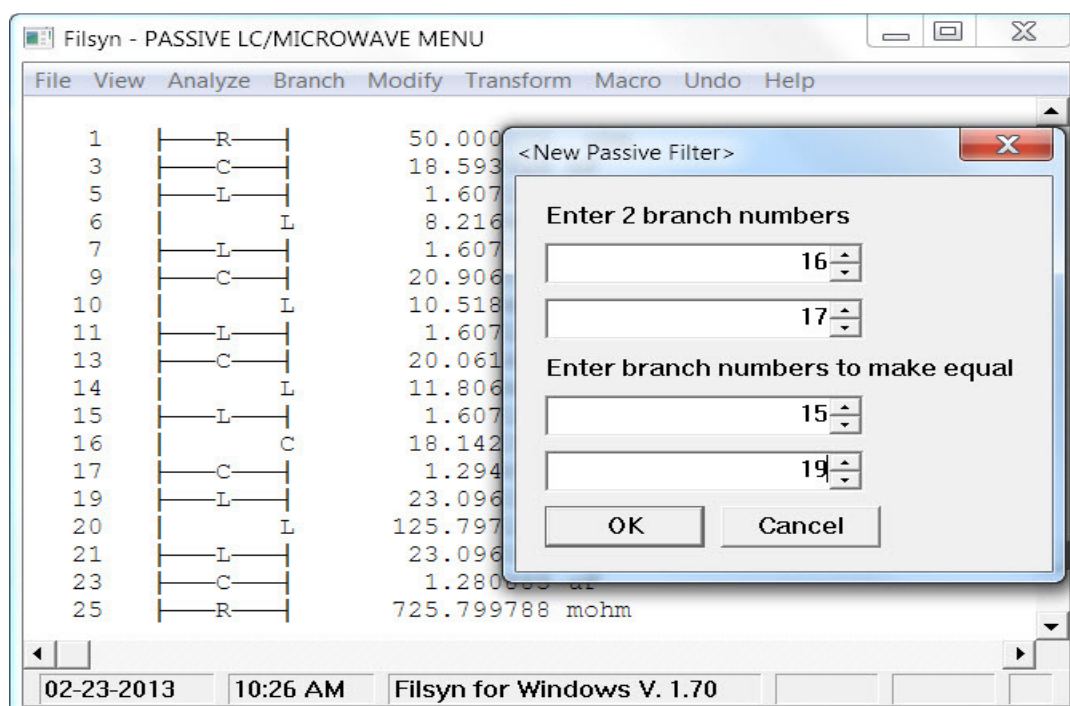
Performing this step two more times and a similar step using L_{15} and L_{16} , the circuit has nearly the right form:



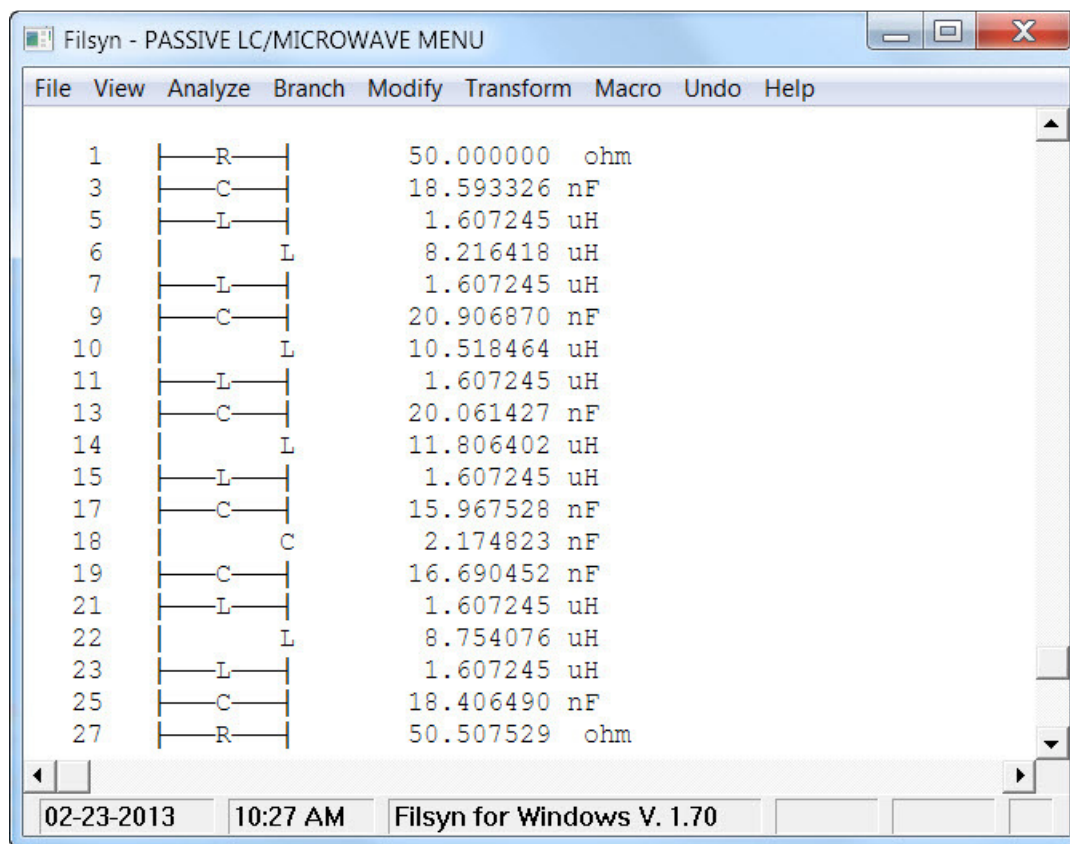
Next we need to use the Norton step again to convert the L section of branches 19 and 20 to a symmetrical pi section and the last step is to use C_{16} and C_{17} to make the values of L_{15} and L_{19} the same. Leaving most of these out, this last step is done using the **Modify -> Norton -> Matching** menu:



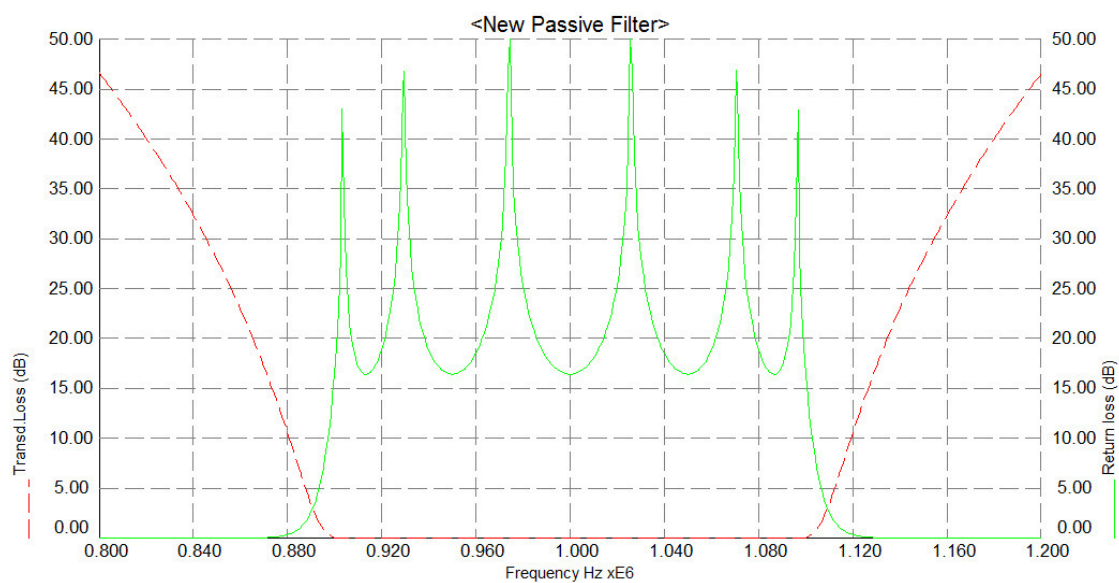
Here we enter the proper branch numbers:



The final circuit is exactly what we need, all shunt inductors are identical:

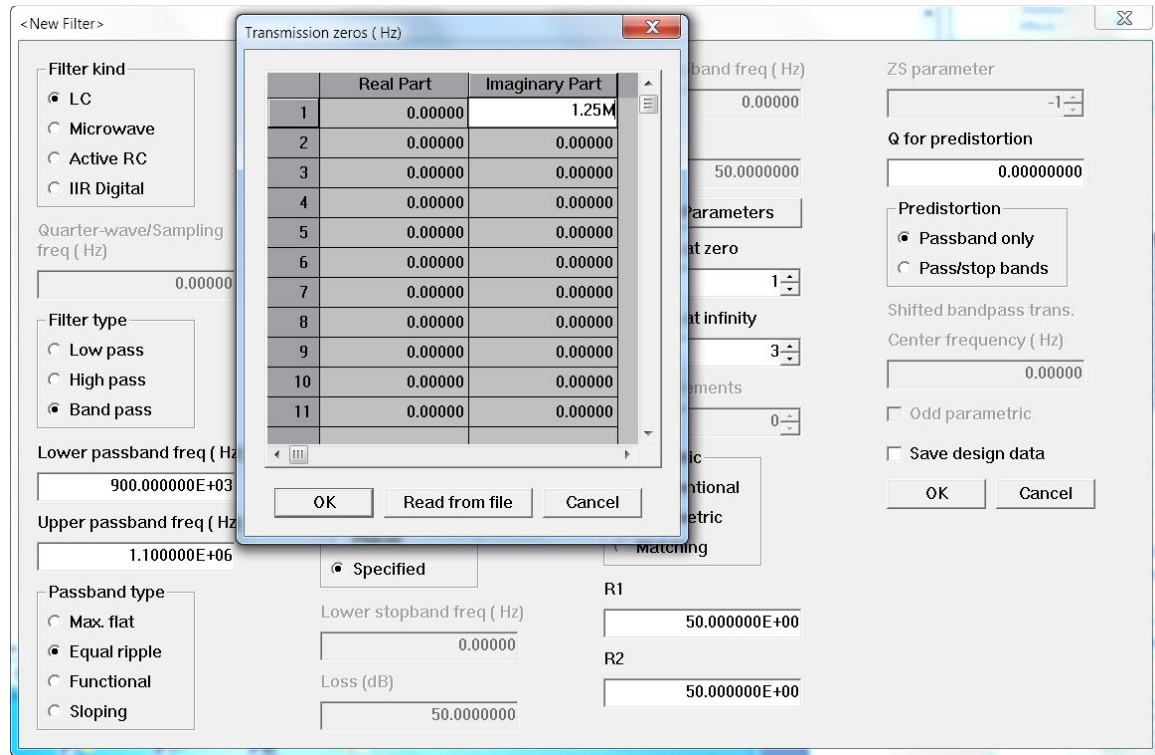


The computed performance is also perfect:

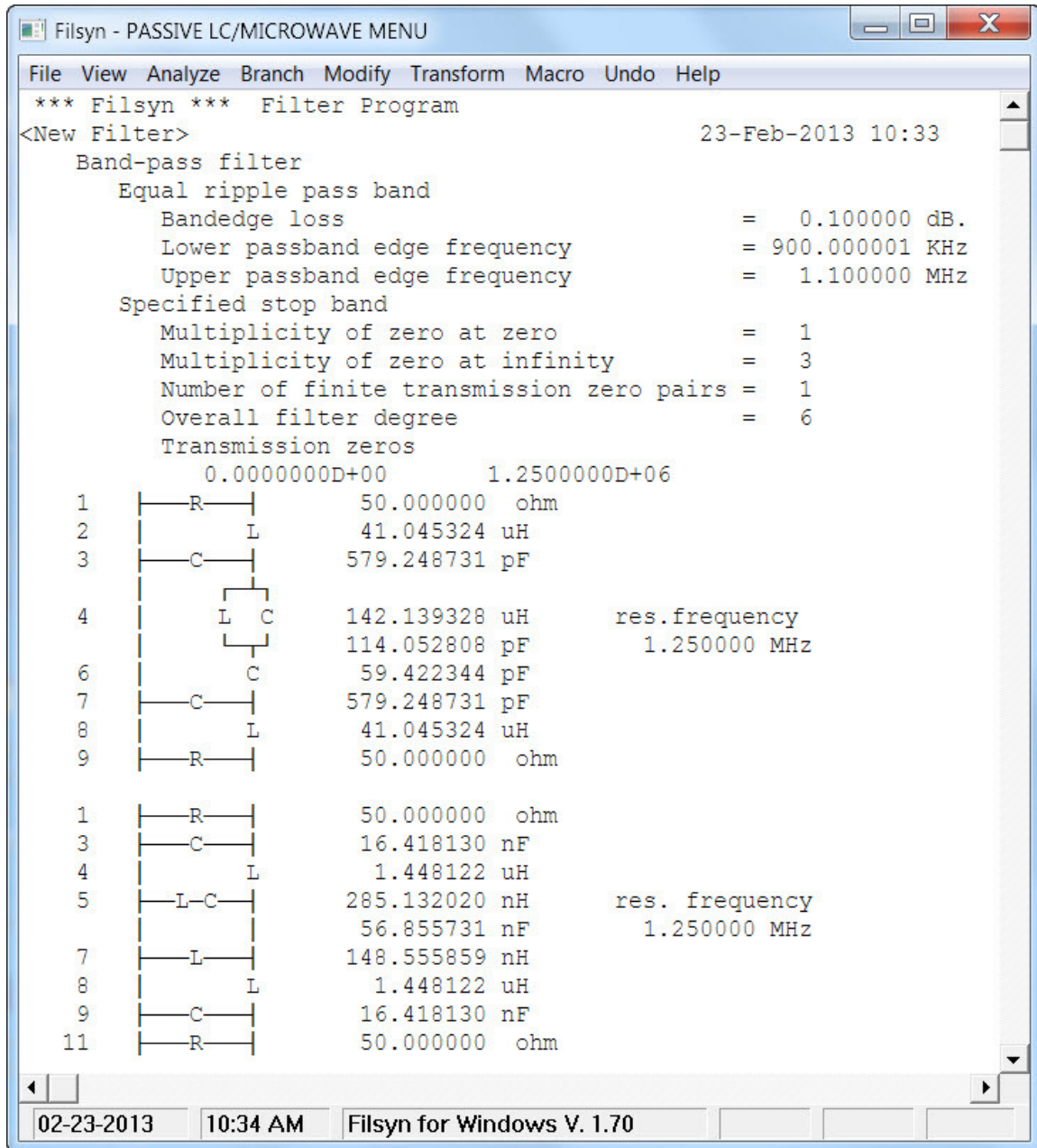


Considering the more general case of having finite transmission zeros in the filter as well, we face a somewhat more complex situation with other options. Let us again start with a simple example.

3. Example 2

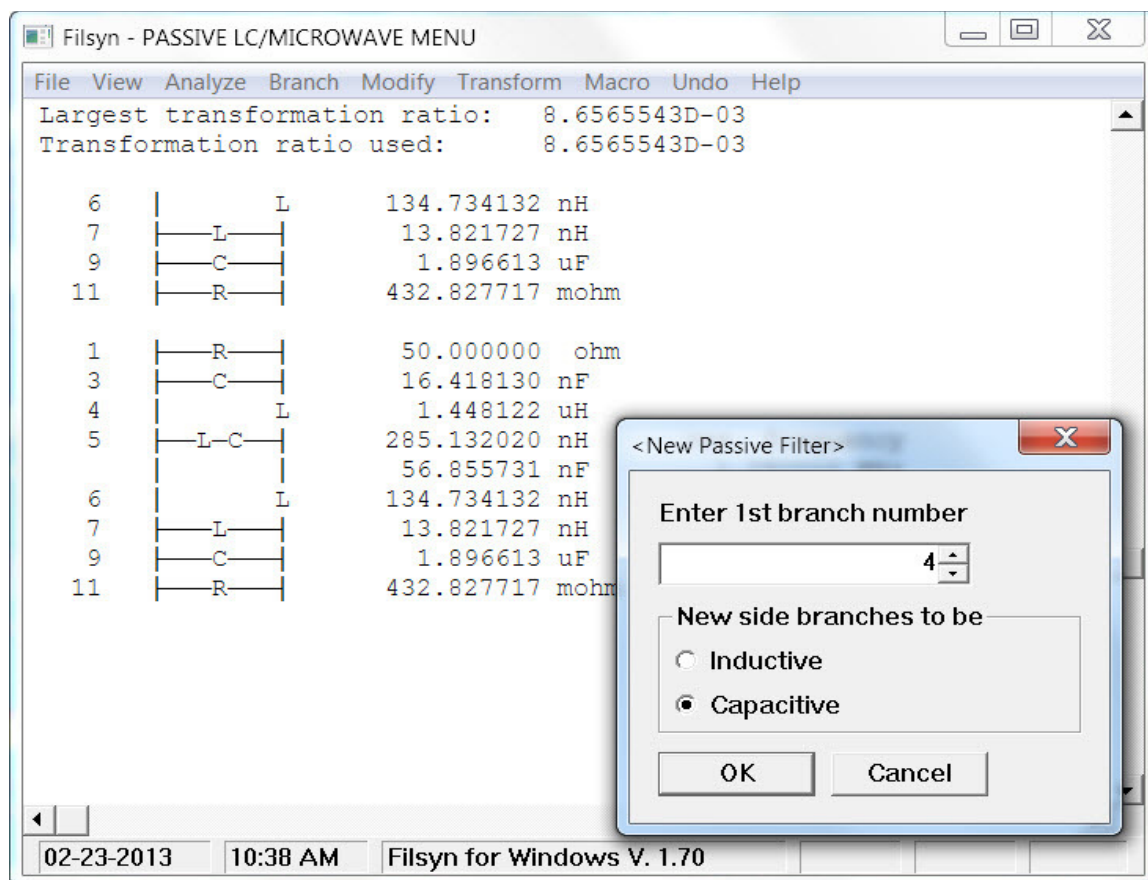


Same passband specification, but only one zero at zero and three at infinite frequencies as well as one finite zero. The computer-selected configuration is:

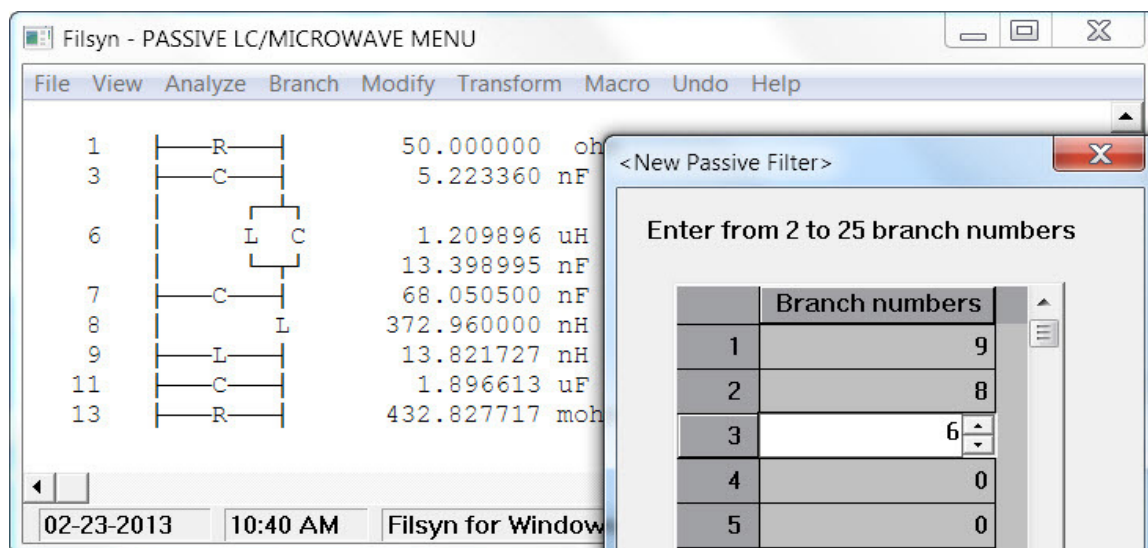


where we again opted for the dual structure instead.

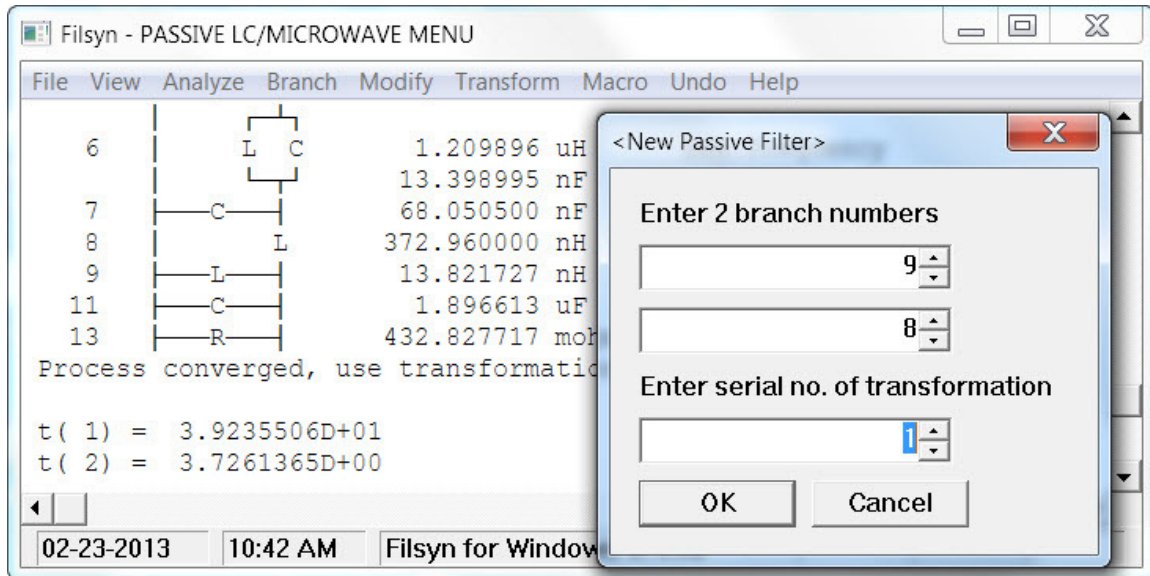
Interchanging branches L_7 and L_8 (using again the **Modify -> Norton -> Simple** menu) results in a Tee section starting with L_4 , which we can convert to a Pi section using the **Modify -> Tee-to-Pi** menu. Note of course, that this is not the simple Tee-to-Pi conversion, since the center branch is a resonant one; it is the *Brune* equivalence instead (see for instance ref[62]). In this operation the series-resonant shunt branch is converted to a parallel-resonant series branch or vice versa. The program performs this conversion and even lets us specify, whether we wish to use capacitive or inductive side branches (permitted only for bandpass filters):



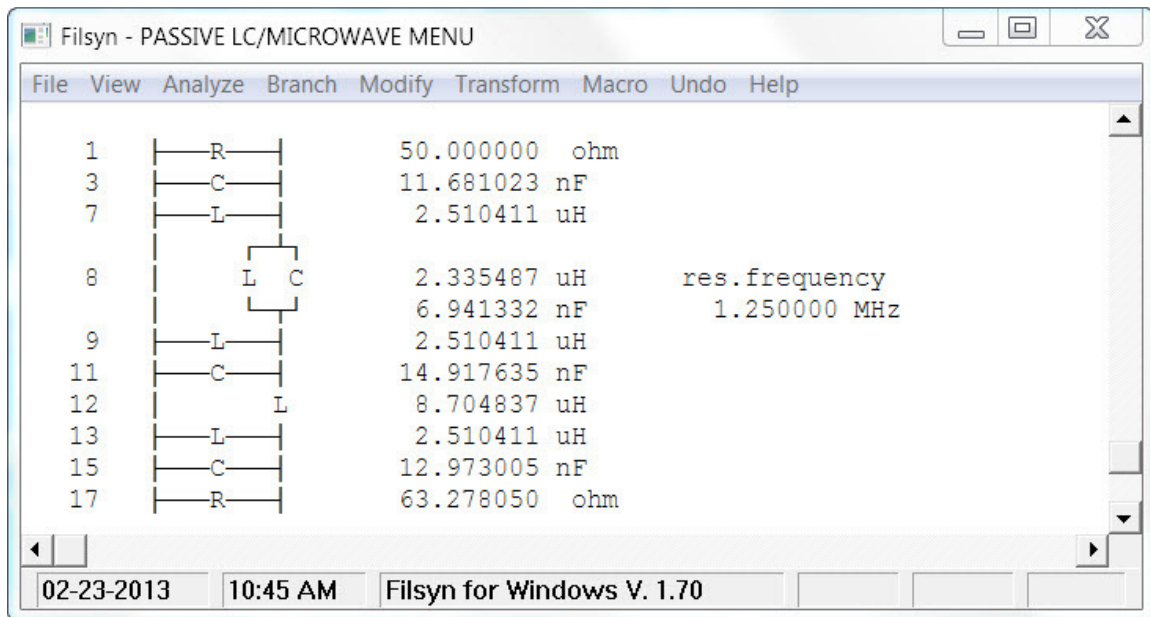
After that, we can distribute L_7 to have parallel branches in all shunt locations, just as we have done in our first example above. The first step is again computing the necessary transformation values:



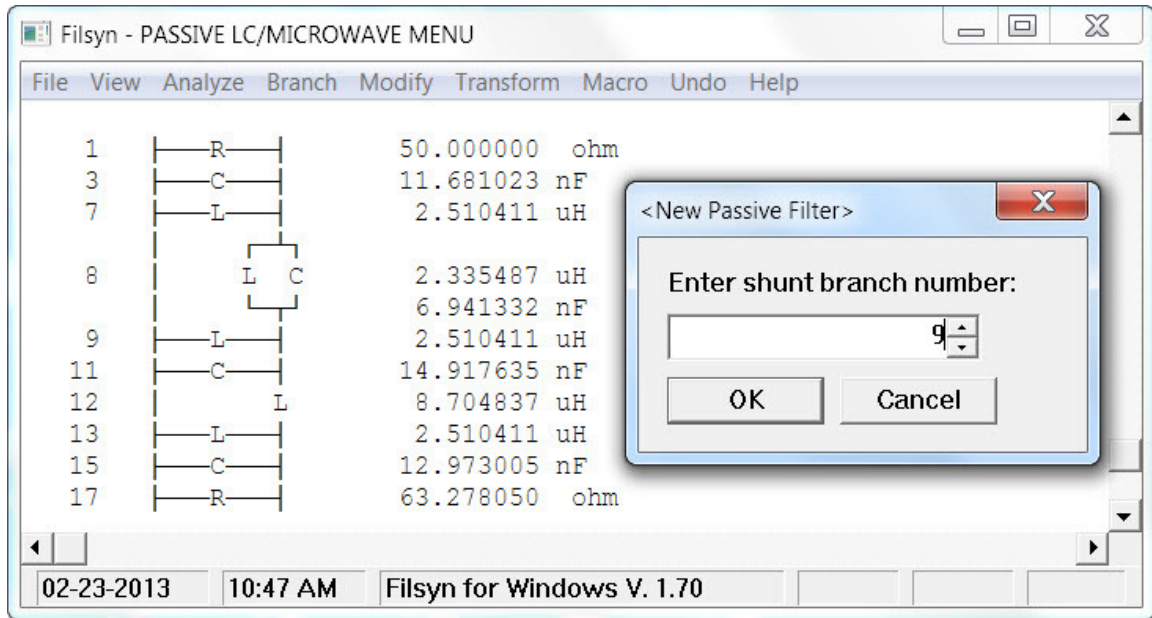
This is followed by the first transformation step:



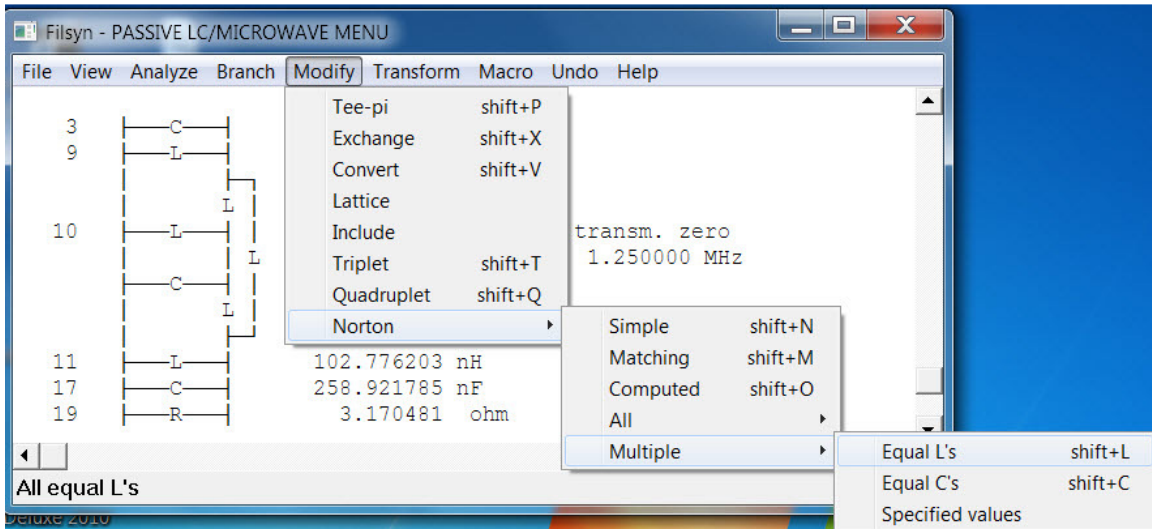
One more transformation step and some cleanup gives us the final circuit below where all shunt inductors now have the same value:



While this might be the final desired circuit, we have an additional option available to us. We can replace the resonant coupling branch by a coupling branch that is connected to nonadjacent resonators. There are two kinds of structures available, triplets and quadruplets. Triplets work on three consecutive resonators and whatever is between them, while quadruplets use four resonators and whatever is between them. Triplets can handle only one resonant coupling branch, while quadruplets need two of these. In both cases they simply need the serial number of the center branch. In the above case this is clearly branch 9 (or 11) and we can ask for a triplet using the **Modify->Triplet** option:

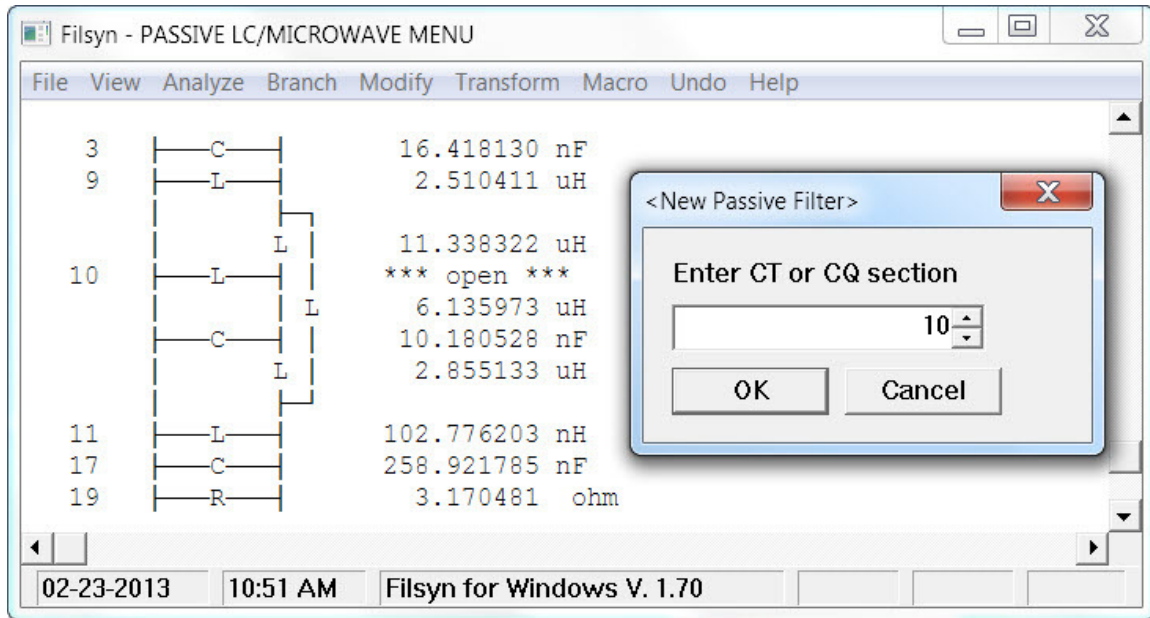


yielding the following circuit (only the modified part of the circuit is shown here):

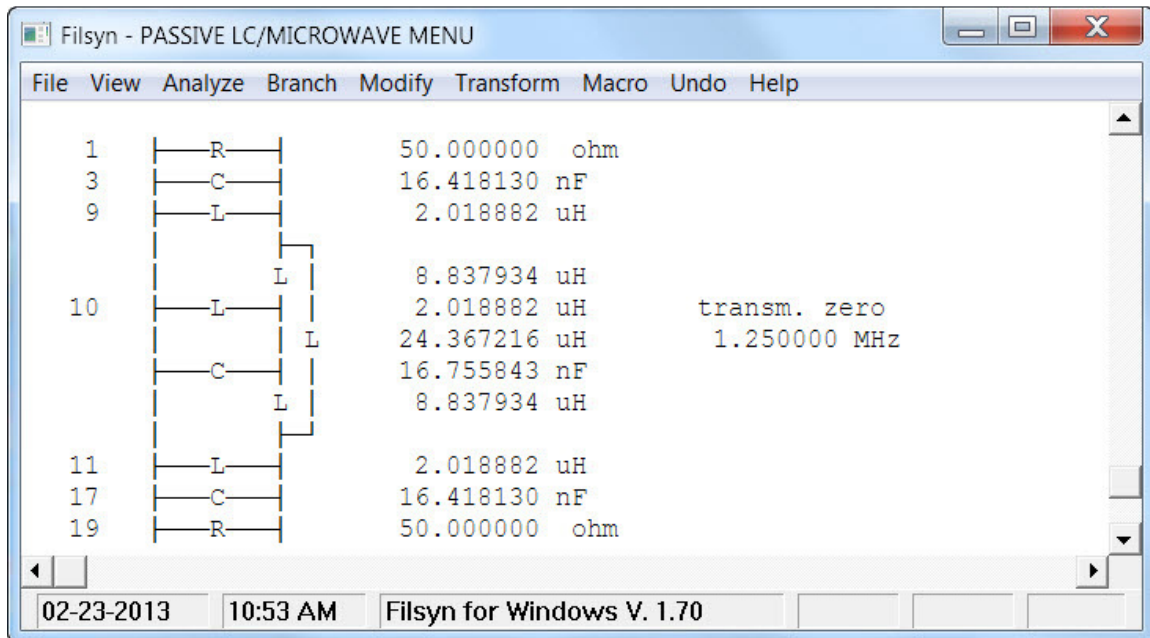


One of the side branches will turn out to be negative in these operations, but those can always be absorbed into already existing positive branches as is observable above.

Finally we need to restore the equal values of the shunt L's using the **Modify->Norton->Multiple->Equal L's** menu option above which leads to:



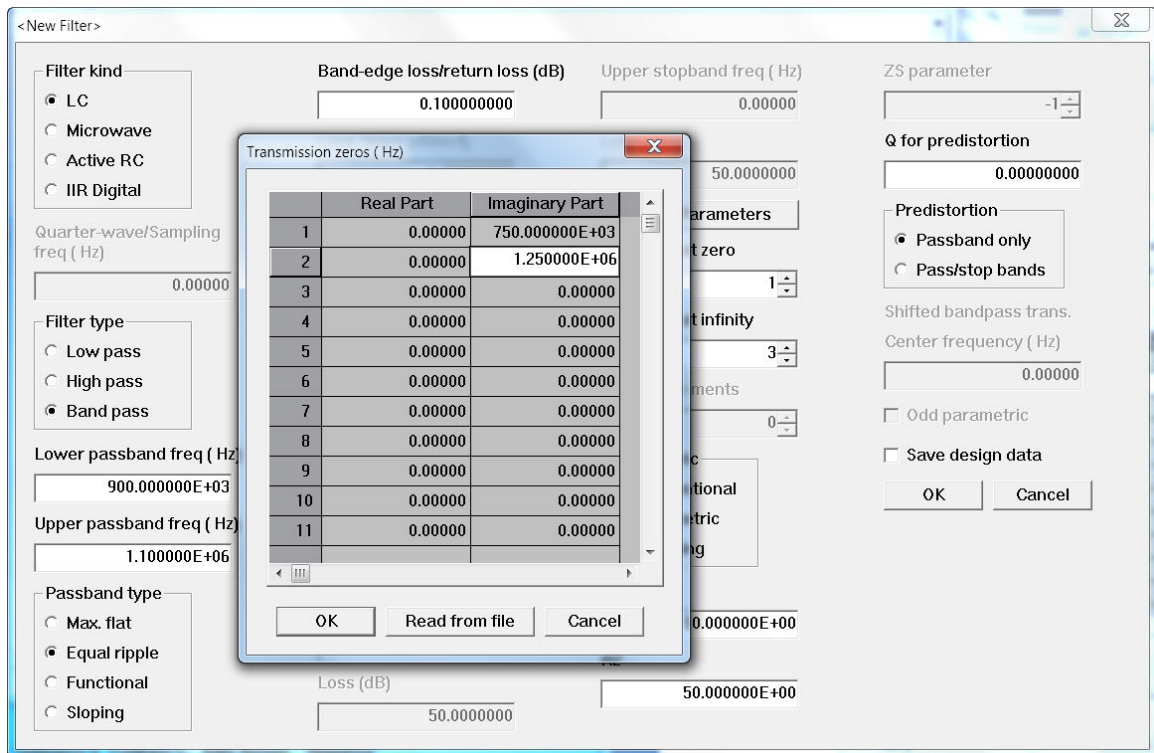
The final set of element values are these:



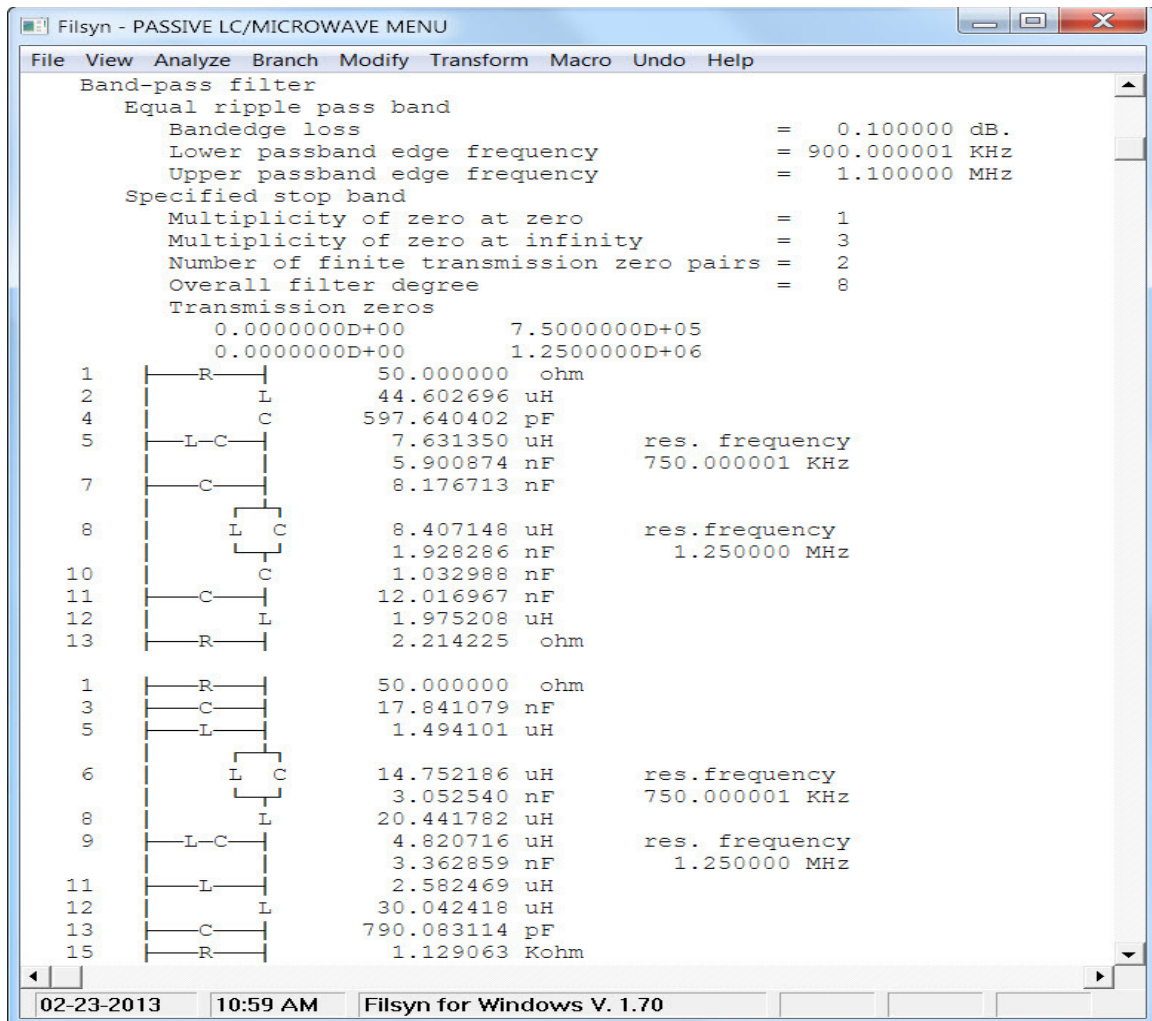
Analyzing either one of the circuits confirms the accuracy of the implementations.

4. Example 3

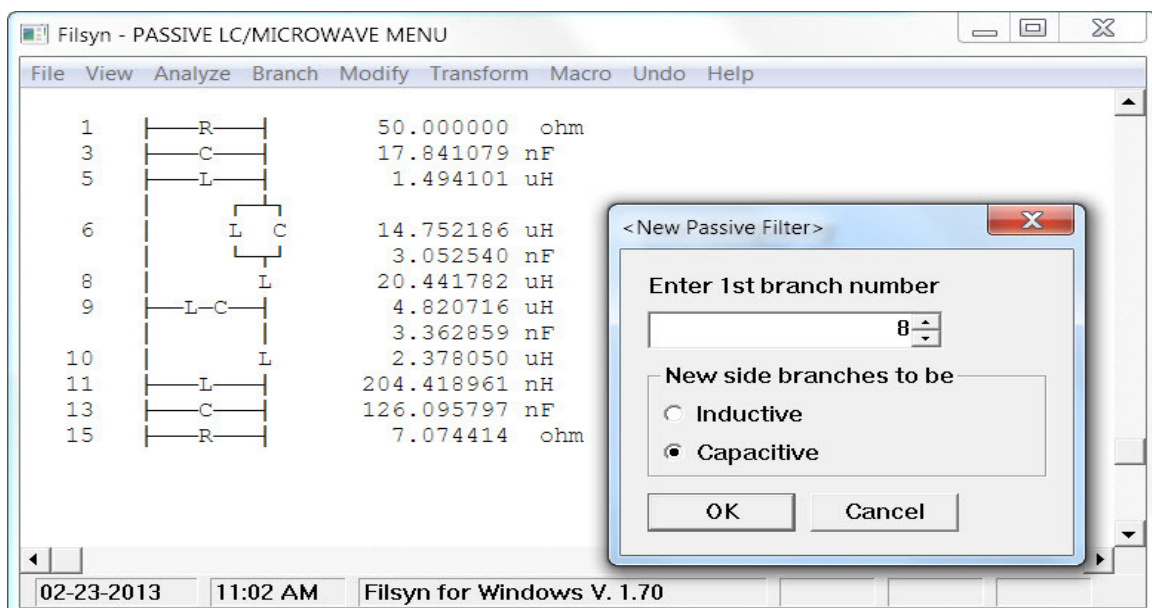
For the quadruplet implementation, we need to start with a more complex design. We use the same passband again, but include two finite transmission zeros, one below the passband, one above:



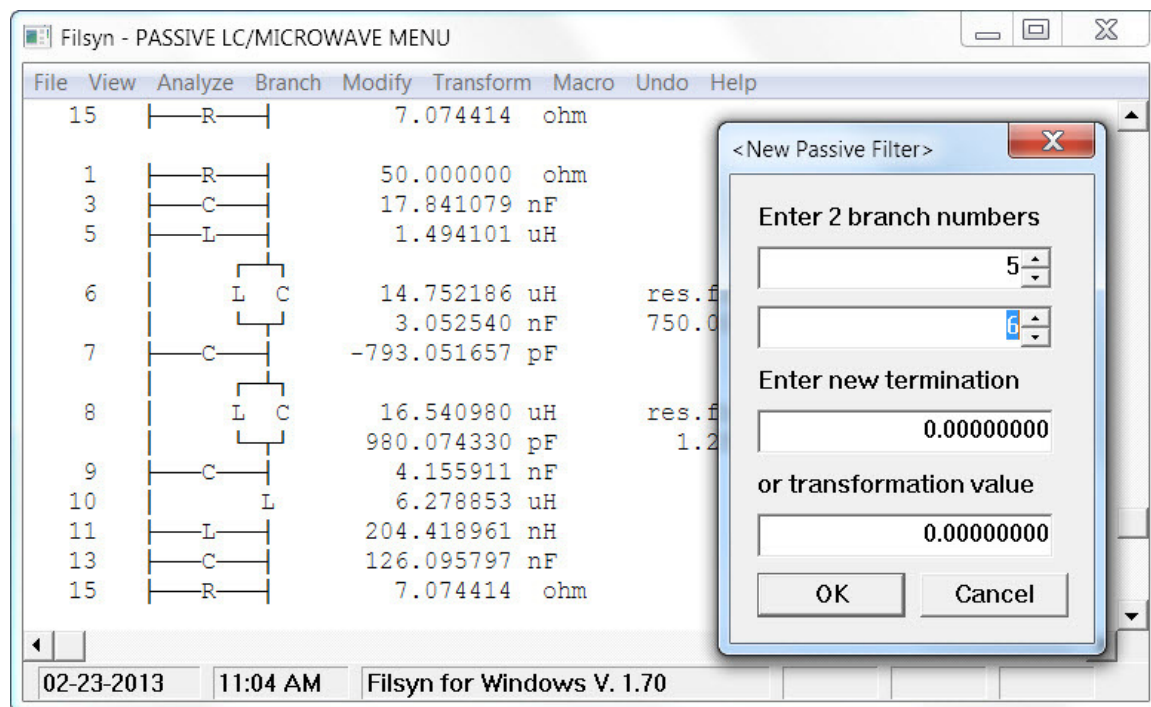
Getting the circuit we again opt for the dual:



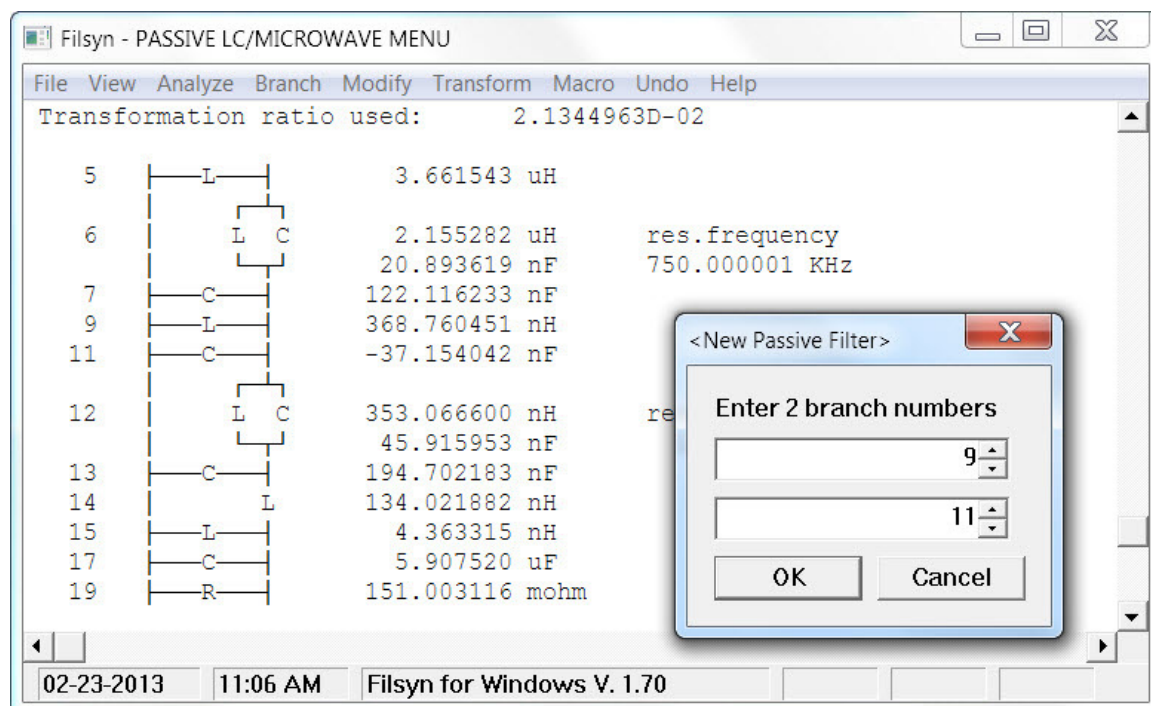
Interchanging L_{11} and L_{12} there is a Tee section starting at branch 8 that we convert to Pi:



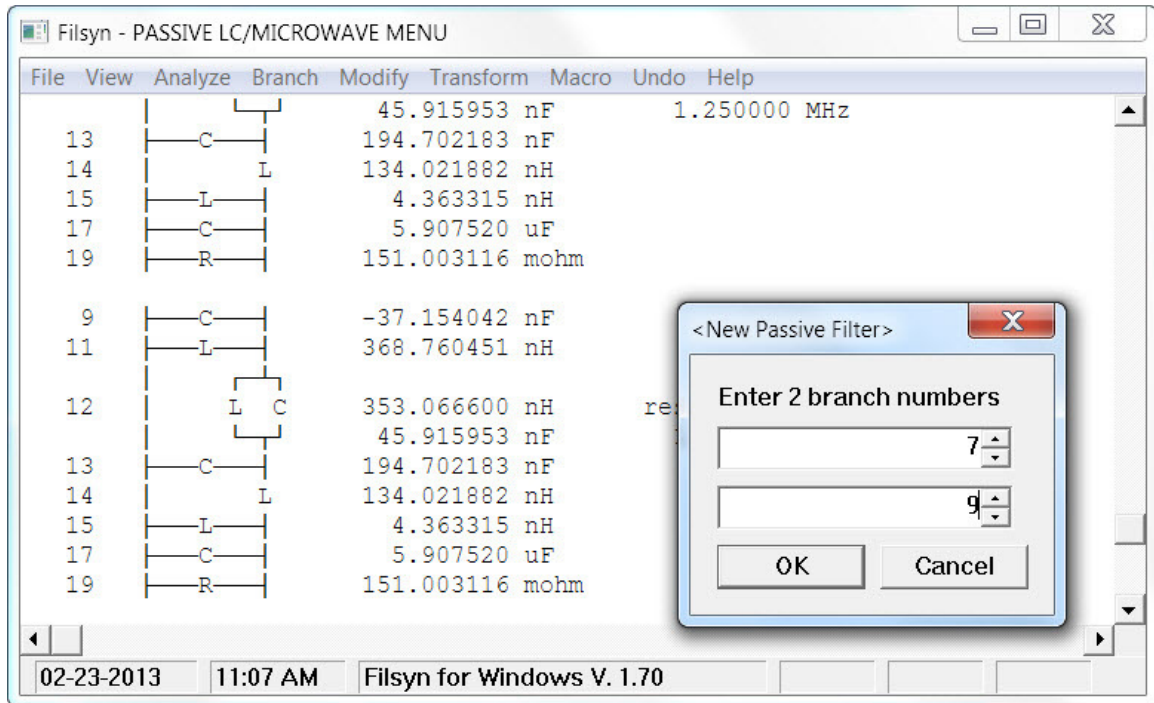
This conversion yields a structure with parallel resonant circuits in the series branches. Next we move L_5 one branch down using **Modify->Norton->Simple** menu again:



Clicking on the **OK** button and then accepting the use of the maximum transformation ratio available, we get:

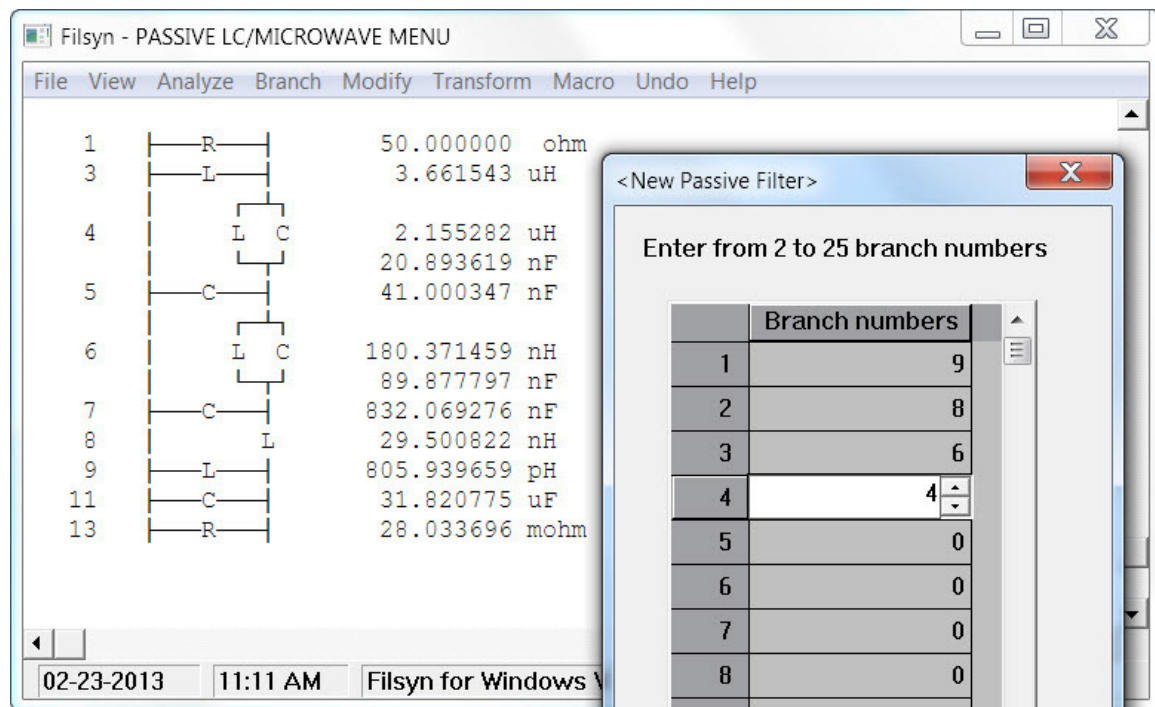


The problem is that the step generates capacitors across C_7 and in order to continue we need to interchange branches 9 and 11, using **Branch->Interchange** first (shown above) and then combine the two parallel C's using **Branch->Combine**:

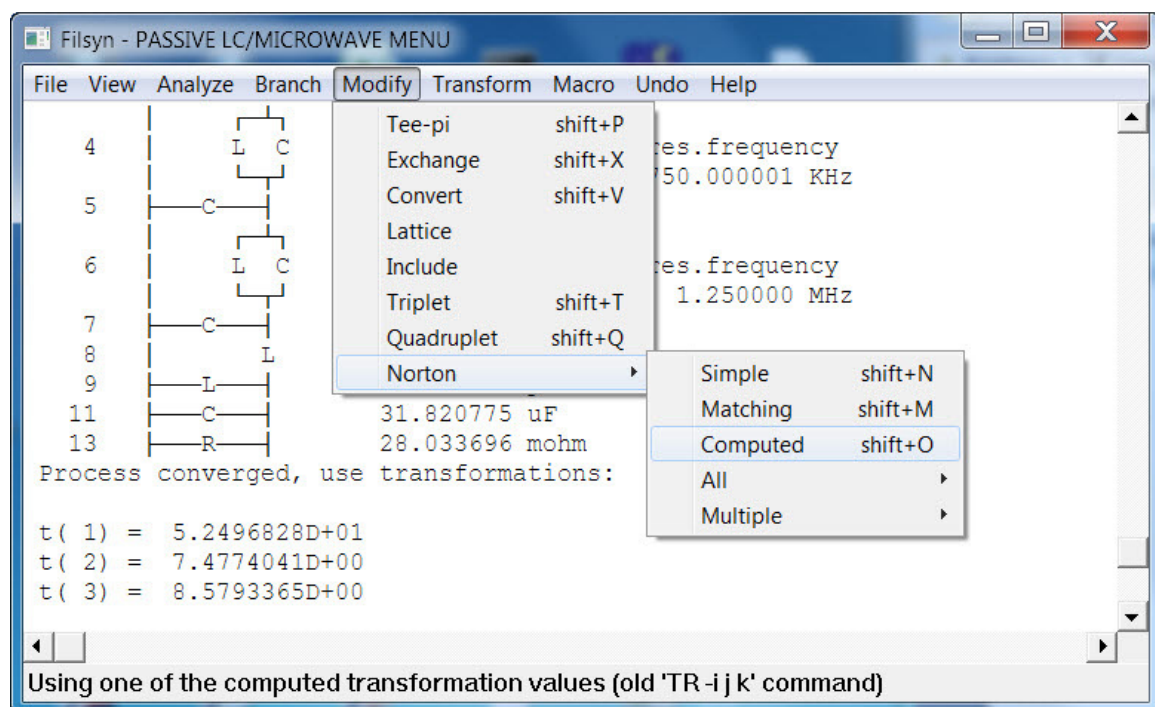


We also need to perform the same step again, since the first one did not remove L_5 completely. Then we have to simplify the resulting structure just as we have done above. Next we need to repeat the operation on the new inductor L_{11} moving it to the other side of branch 12 and so on, and so forth. This is not difficult, only tedious, but eventually we get to the circuit with only a single shunt L at the end. Do not worry about the negative C_5 , that will disappear in the sequel.

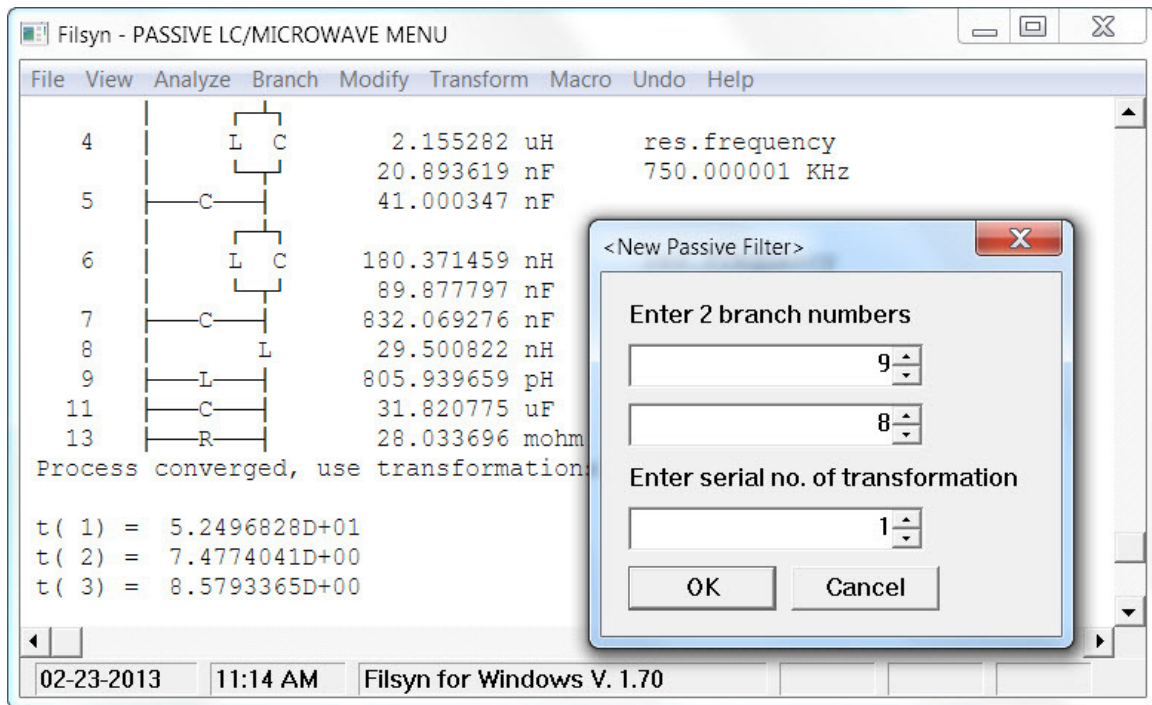
At this step we again need to compute a set of transformation values to redistribute the single inductor L_{19} to each shunt capacitor of equal value. First step is to compute the necessary transformation ratios using **Modify->Norton->All->All**. Note that the order of the branch numbers is important, but whether it is going forward or in reverse is immaterial.



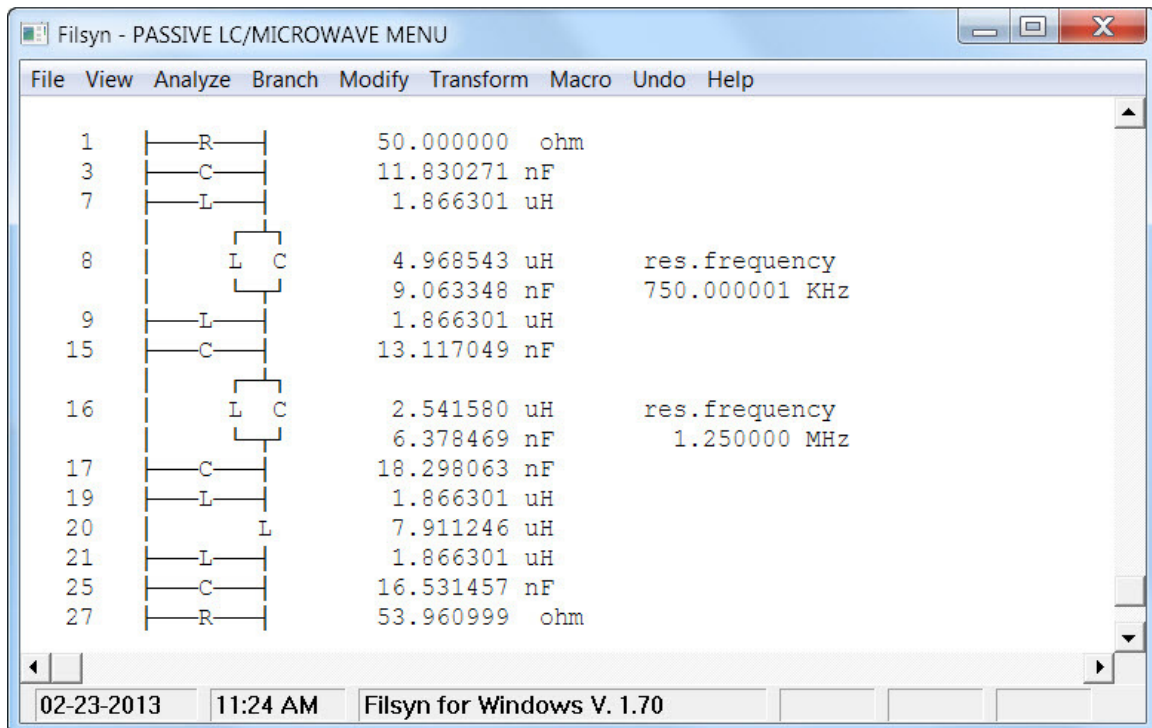
Having the ratios, called 'transformations' below, we need to apply them one at a time. To use these factors we called the **Modify->Norton->Computed** menu:



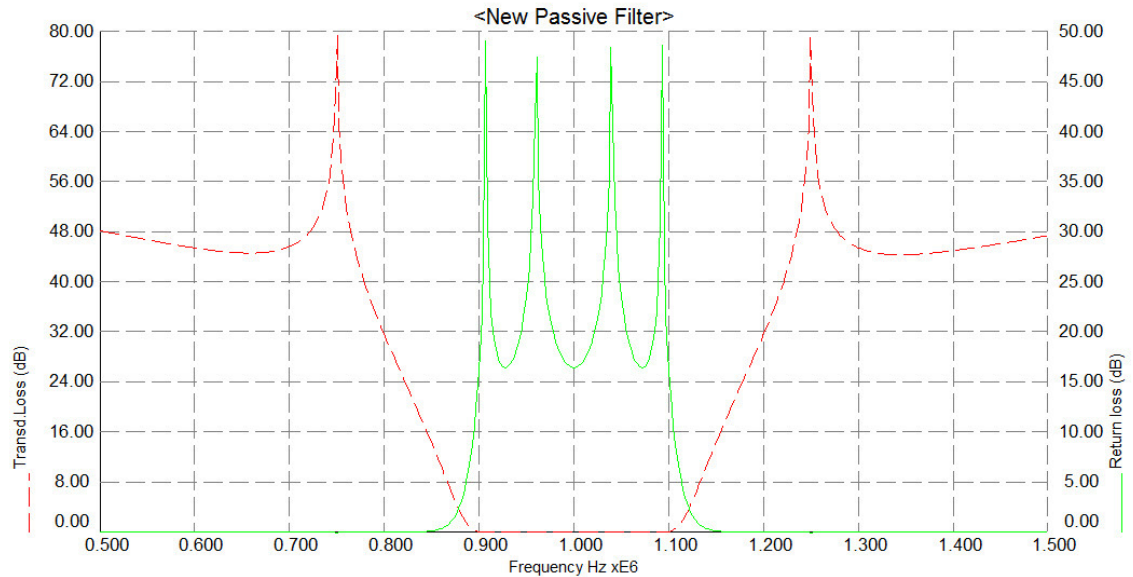
The data necessary are:



where we specified the needed branch numbers for the first step. After two more steps, we get the final circuit:



All shunt L's have again the same value and the frequency domain analysis shows perfect behavior.

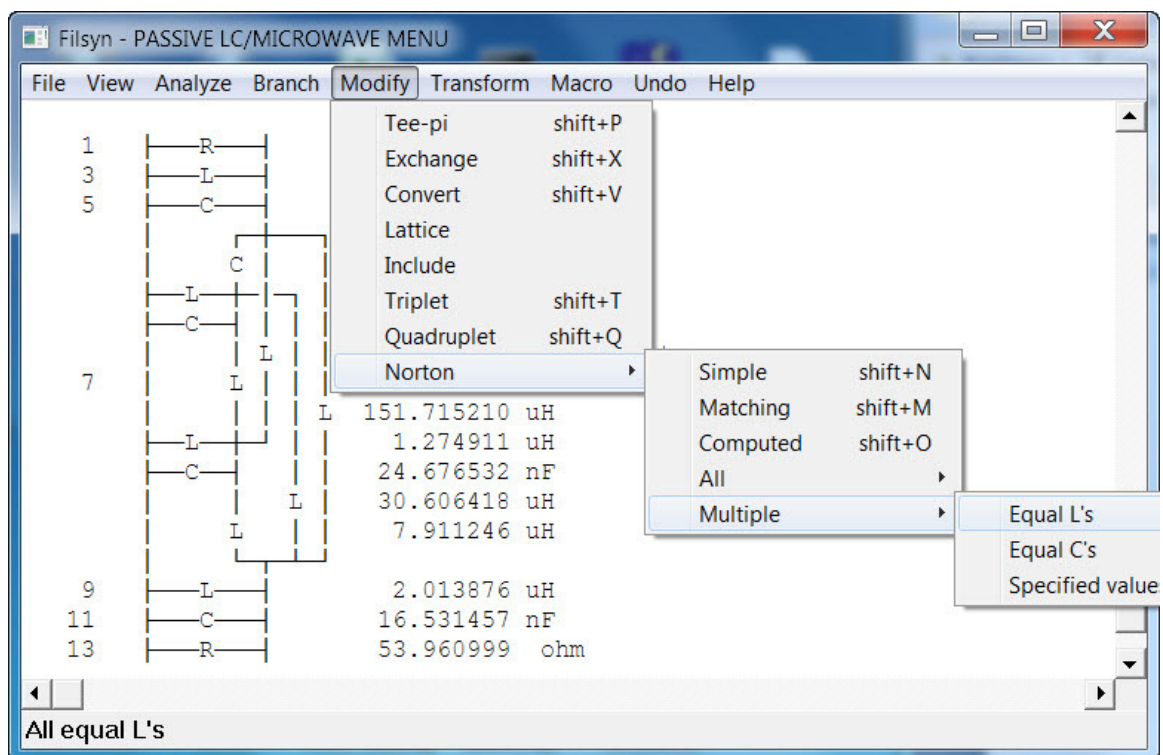


Now we are in the position to demonstrate the quadruplet implementation. Here we have four shunt resonators with two resonant coupling branches which is what we need. Calling the **Modify->Quadruplet** menu and specifying branch 16 as the center:

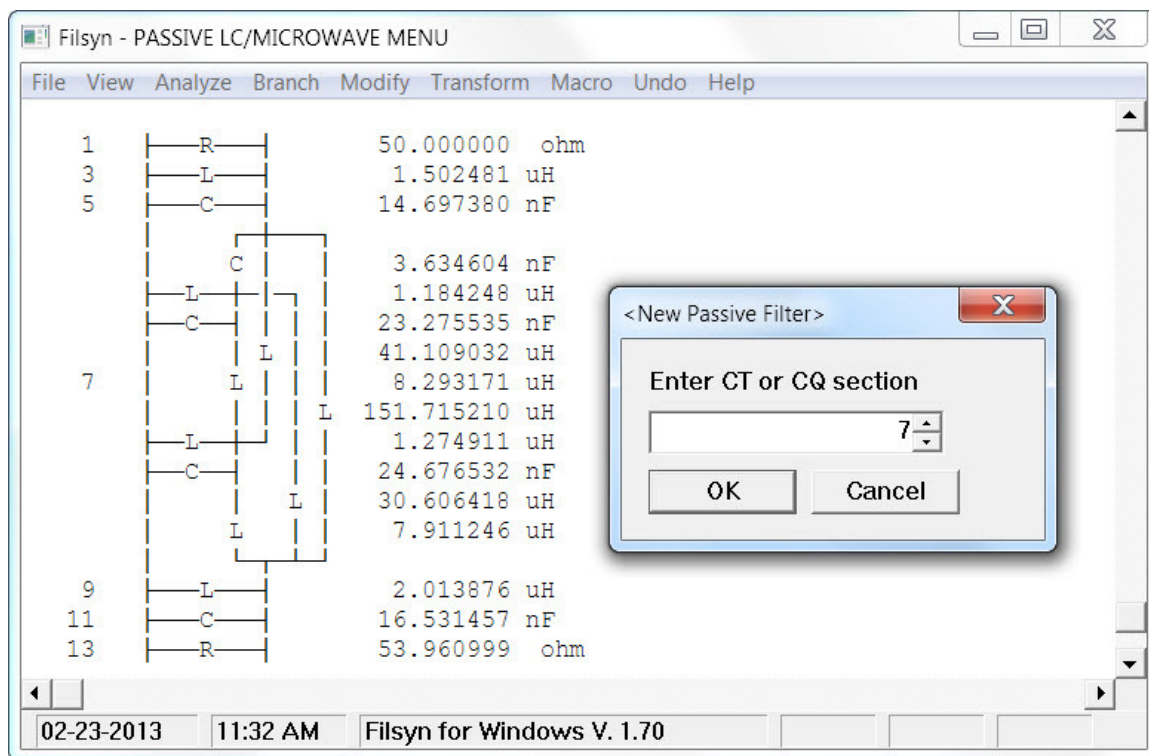
Branch	Component	Value	Unit
1	R	50.000000	ohm
3	C	11.830271	nF
7	L	1.866301	uH
8	L	4.968543	uH
9	C	9.063348	nF
15	L	1.866301	uH
16	C	13.117049	nF
17	L	2.541580	uH
19	C	6.378469	nF
20	L	18.298063	nF
21	L	1.866301	uH
25	C	7.911246	uH
27	R	1.866301	uH
		16.531457	nF
		53.960999	ohm

res.frequency
750.000001 KHz

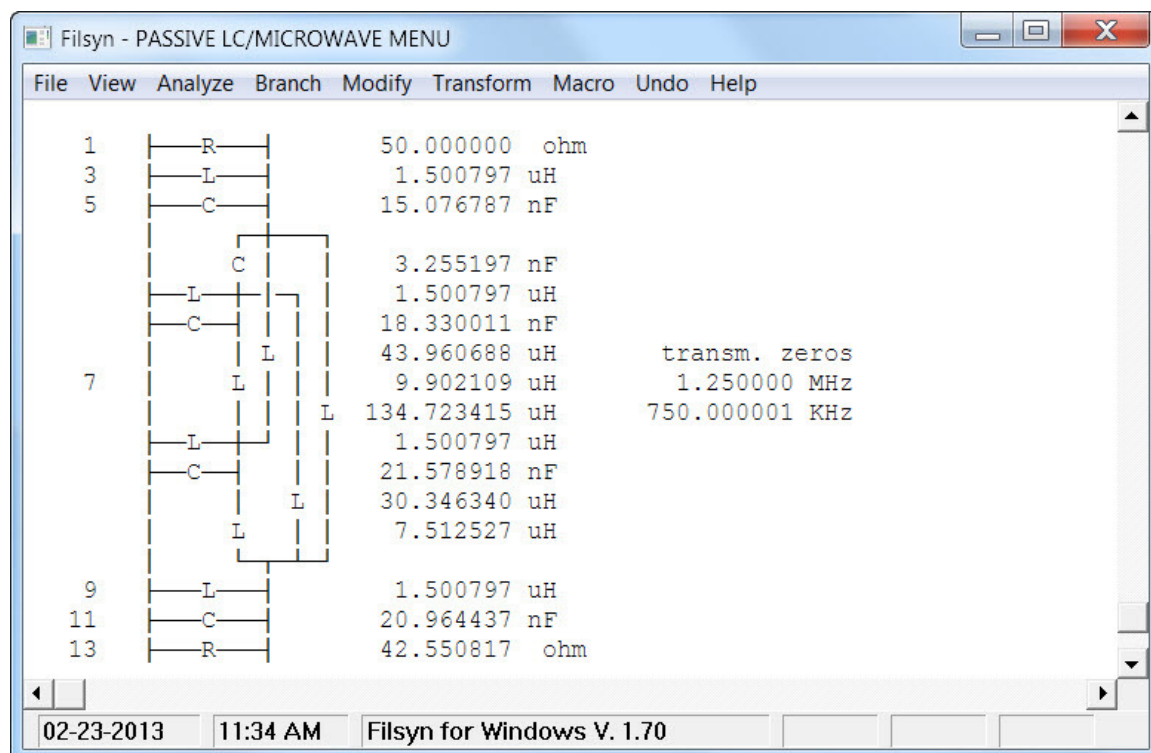
We get a form of the quadruplet, but with unequal shunt L's. However we can correct that by using again the **Modify->Norton->Multiple->Equal L's** menu:



Branch 7 is of course, the quadruplet and its analysis agrees with the previous one.



The resulting circuit has all shunt L's equal and its analysis is identical to the one shown before.



5. Comments

As one can see, the design of a coupled resonator filter is not difficult but can be very tedious especially for those who use the program only occasionally. Therefore we have built into the **Filscript.exe** the capability of performing nearly all operations necessary for this design procedure. In order to make this procedure extendible, we wrote external (compiled) subroutines that the Filscript.exe program can call to perform the necessary operations. The names of these routines (we have about 240 of these now in the “Scripts” folder) tell you what filter they were written for. The names start either with the letter P – for parametric – or the letter C – for conventional – i.e. nonparametric. This is followed by four integers and finally by “syn” and an “.wbc” extension:

$P\alpha\beta\gamma\delta\text{syn.wbc}$

or

$C\alpha\beta\gamma\delta\text{syn.wbc}$

Here

α is the number of zeros at zero frequency

β is the number of zeros in the lower stopband

γ is the number of zeros in the upper stopband and

δ is the number of zeros at infinity.

For example, the files for our second and third examples above would be “C1013syn.wbc” and “C1113syn.wbc” respectively.

Note that if any of these integers is greater than 9, then we use the corresponding hexadecimal letter instead. None of the numbers may go over 13. If you encounter an instance where a script file is missing we shall be happy to write one.

All of these scripts are used in the **Filters with finite zeros** menu of the filscript.exe program and they work equally well for microwave filters.

APPLICATION NOTE 14

ACTIVE RC FILTER USING FDNR

1. Introduction

In addition to the three synthesis methods presented in *Chapter 6* of the manual, one can obtain other structures with the help of **Filsyn**. In particular, consider the design technique using FDNR's (Frequency Dependent Negative Resistors). As a starting point, this requires a passive LC containing the minimum number of capacitors; preferably with all capacitors grounded. As explained by *Bruton* [4] (see also ref. [9]), each capacitor will be converted into an FDNR using a pair of operational amplifiers. The inductors will become resistors and the two terminating resistors are replaced by capacitors. This is equivalent to dividing all branch impedances by the complex frequency variable ' s '.

Lowpass and highpass filter structures are fixed and the only step we can perform is to take the dual of the computer-generated structure to yield the minimum number of capacitors. Everything else is described by *Bruton*. In the bandpass case, however, we may use **Filsyn** to provide the most convenient structure for this conversion. We will demonstrate this procedure by an example.

2. Filter Design

The filter specification is unimportant except that it should be a parametric bandpass. We selected four finite transmission zeros (two above the passband and two below) and the minimum values (unity) for the multiplicities of those at zero and infinite frequencies. Everything else is selected arbitrarily and the data input session is as shown below.

<New Filter>

Filter kind

- ☒ LC
- ☐ Microwave
- ☐ Active RC
- ☐ IIR Digital

Quarter-wave/Sampling freq (Hz)

0.00000

Filter type

- ☐ Low pass
- ☐ High pass
- ☒ Band pass

Lower passband freq (Hz)

1.000000E+03

Upper passband freq (Hz)

2.000000E+03

Passband type

- ☐ Max. flat
- ☒ Equal ripple
- ☐ Functional
- ☐ Sloping

Band-edge loss/return loss (dB)

0.100000000

Loss Slope (dB/oct)

6.00000000

Flat loss (dB)

0.00000000

Function Type

- ☒ E
- ☐ F

Multiplier

0.00000000

Stopband

- ☐ Monotonic
- ☐ Equal min
- ☐ Placer
- ☒ Specified

Lower stopband freq (Hz)

0.00000

Loss (dB)

50.0000000

Upper stopband freq (Hz)

0.00000

Loss (dB)

50.0000000

Detail Parameters

of zeros at zero

1

of zeros at infinity

1

of unit elements

0

Parametric

- ☐ Conventional
- ☒ Parametric
- ☐ Matching

R1

1.000000E+00

R2

1.000000E+00

ZS parameter

-1

Q for predistortion

0.00000000

Predistortion

- ☒ Passband only
- ☐ Pass/stop bands

Shifted bandpass trans. Center frequency (Hz)

0.00000

☐ Odd parametric

☐ Save design data

OK Cancel

The transmission zeros are entered after clicking on the **Detail Parameters** button:

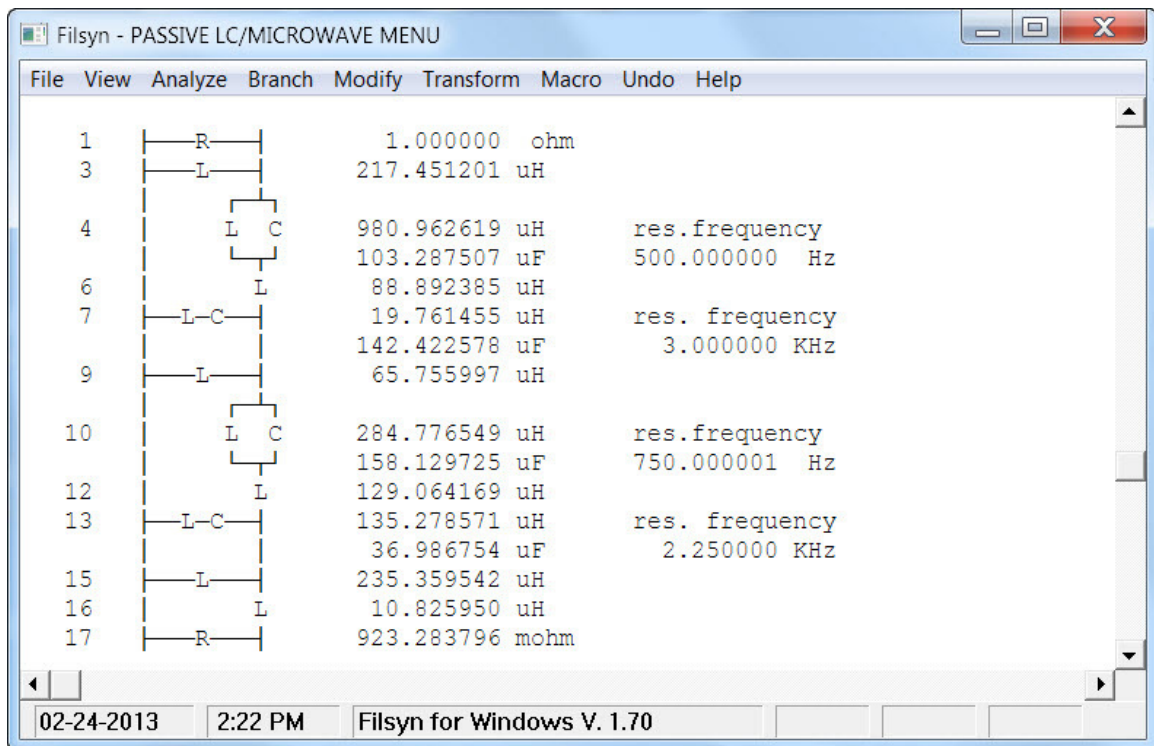
Transmission zeros (Hz)

	Real Part	Imaginary Part
1	0.00000	500.000000E+00
2	0.00000	750.000000E+00
3	0.00000	2.250000E+03
4	0.00000	3.000000E+03
5	0.00000	0.00000
6	0.00000	0.00000
7	0.00000	0.00000
8	0.00000	0.00000
9	0.00000	0.00000
10	0.00000	0.00000
11	0.00000	0.00000

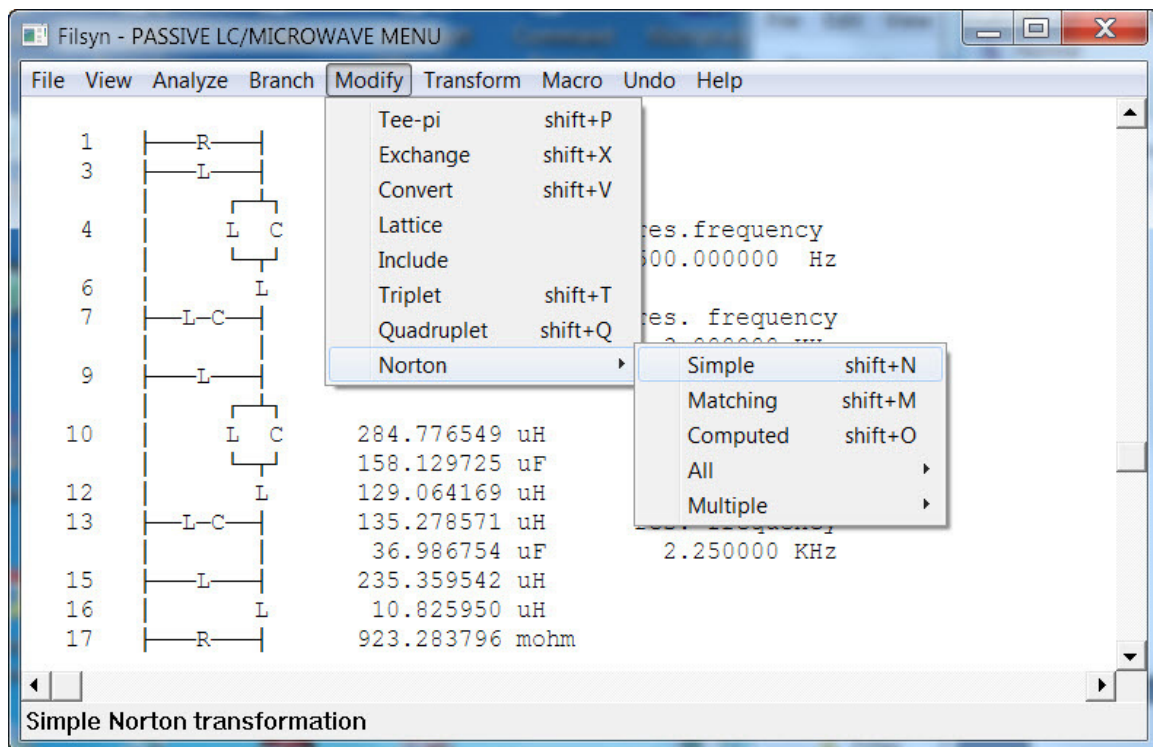
OK Read from file Cancel

The summary printout is of no concern, we proceed to the passive synthesis directly. The circuit required can be obtained several different ways; the simplest is the auto synthesis of a structure with the minimum number of inductors. These inductors should appear in parallel LC circuits in the series branches. This structure can be realizable only if the zeros are all above the passband, but the program lets us specify this structure in the manual mode anyway, if negative element values are permitted.

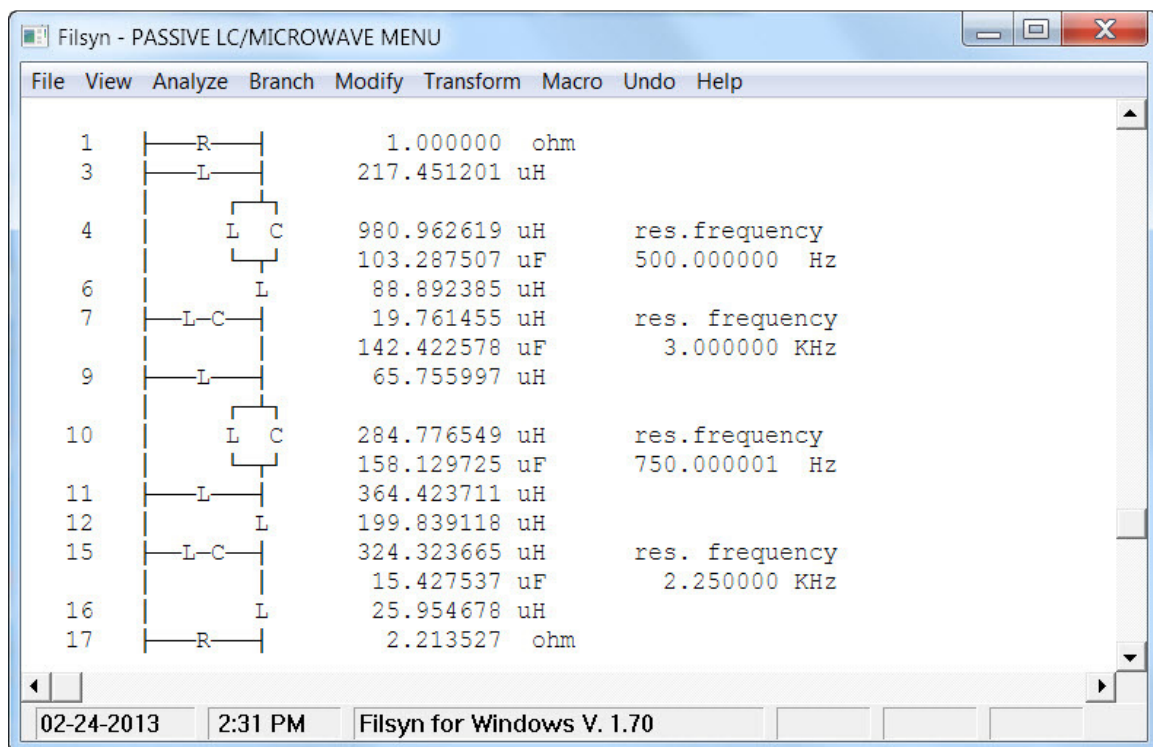
Using the computer-selected method of synthesis instead, the procedure is uneventful. The computer-generated circuit however needs a series of modifications in order to convert the structure into the one we need. Since the starting structure has the minimum number of inductors and many more capacitors, we first take its dual as shown below. Note that it now has only four capacitors, the absolute minimum needed.



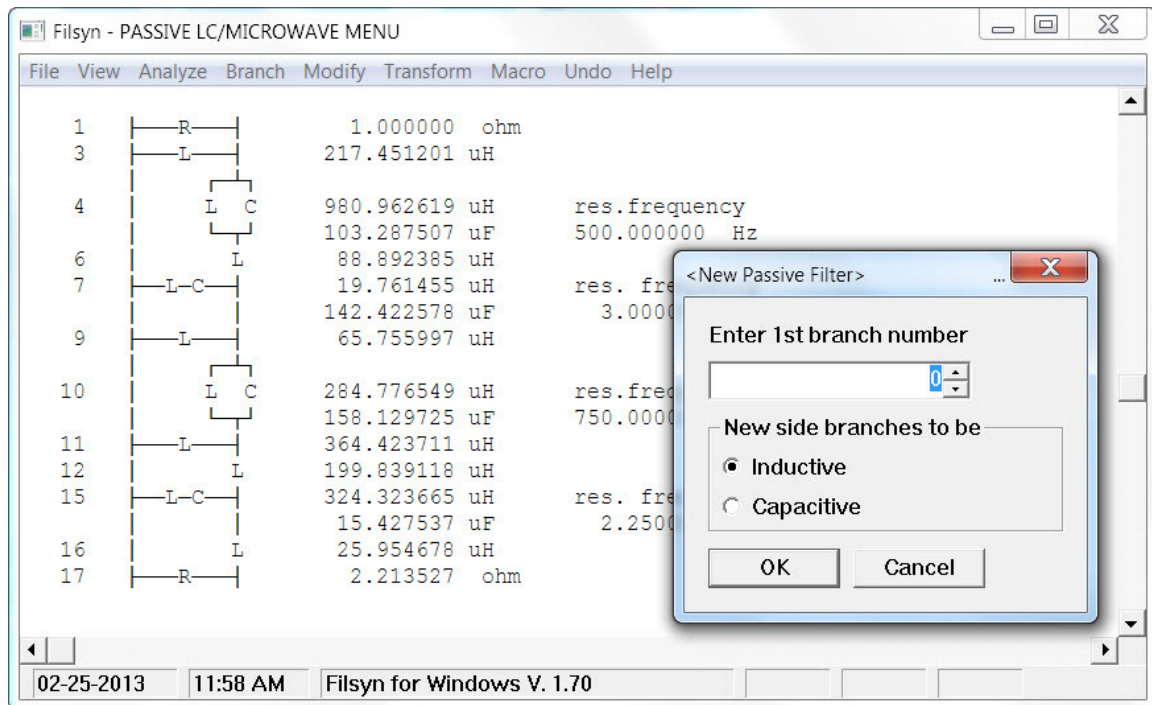
Our next step is to eliminate the parallel resonant circuits in the series branches. This can be accomplished by the **Modify->Tee-pi** conversion menu, but in order to do that, we need shunt inductors on both sides of the series resonant circuits. Our first step is therefore to invert the L-section L_{12} and L_{15} using the **Modify->Norton->Simple** menu item as shown below and specifying these branch numbers at the subsequent prompt as well as accepting the maximum available transformation yielding:



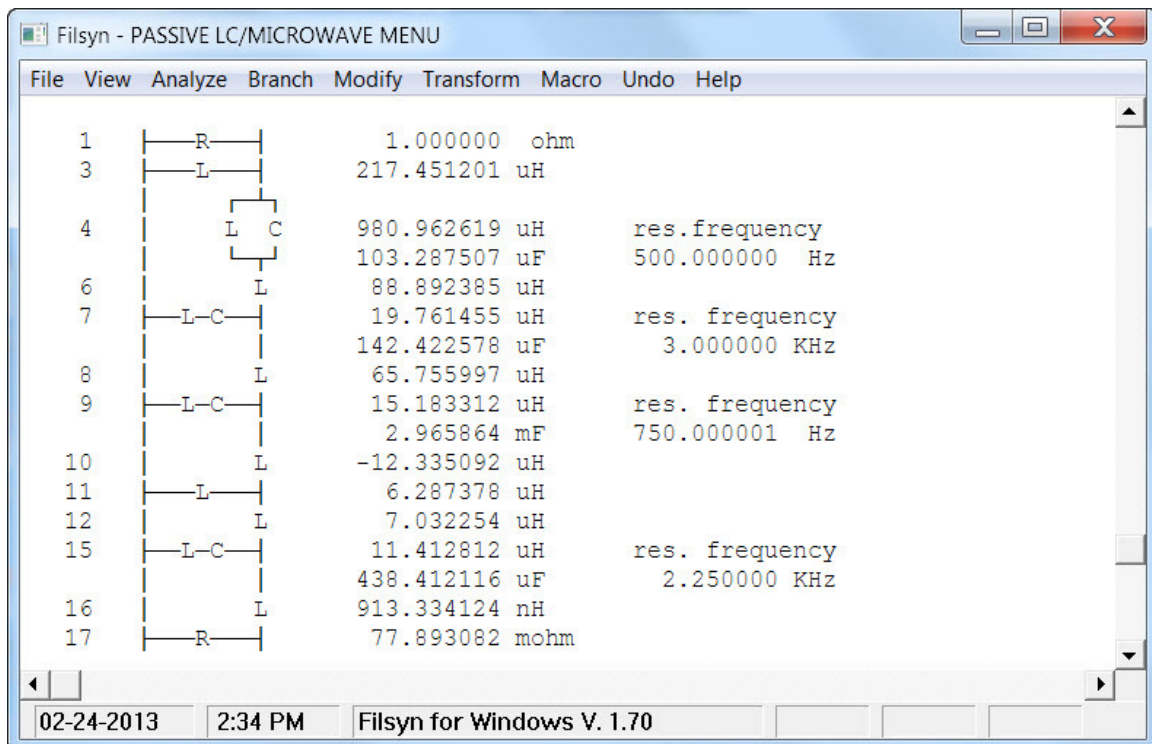
Now we have a pi-section (branches 9, 10 and 11), that we can convert to a tee:



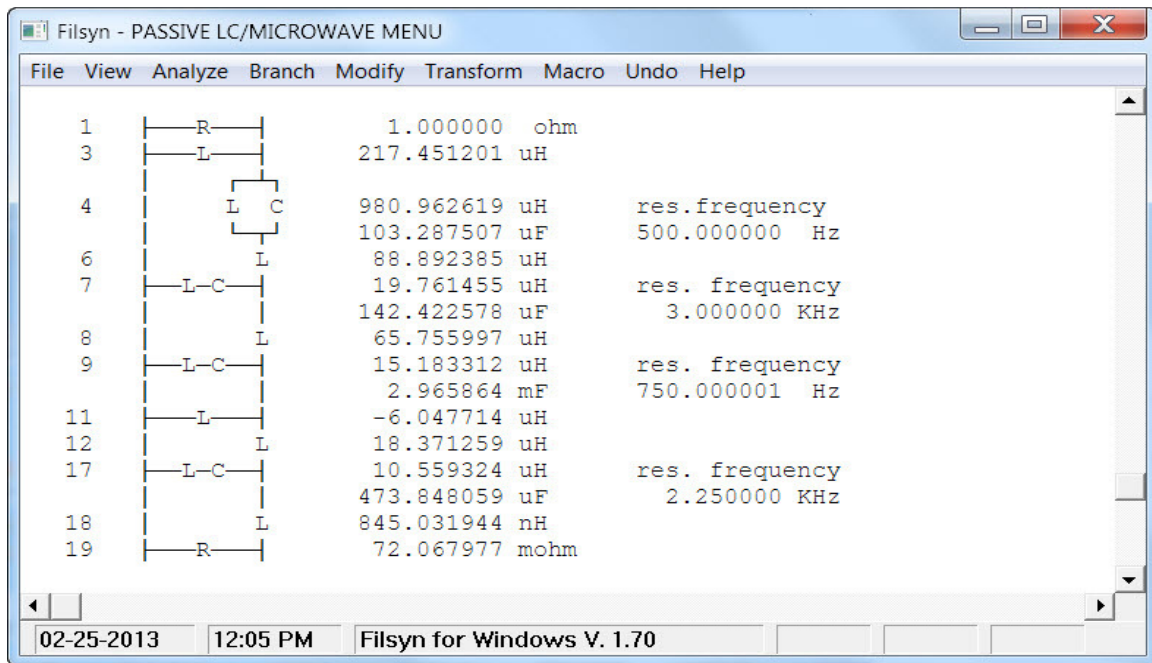
When we select the **Modify->Tee-pi** menu item, we need to enter the number of the first branch of the tee (or pi):



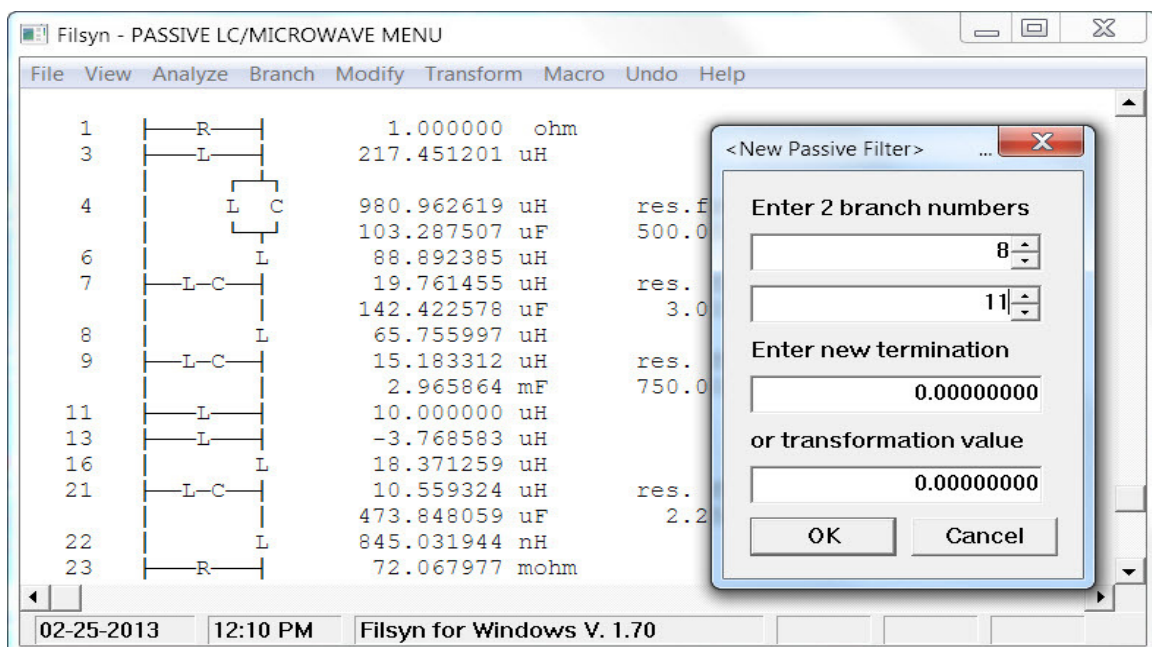
The resulting circuit has the proper resonant circuit, but it has a negative inductor, which will be converted in the process to a negative resistor. This is no problem as long as the negative resistor is grounded at one end, which this one is not.



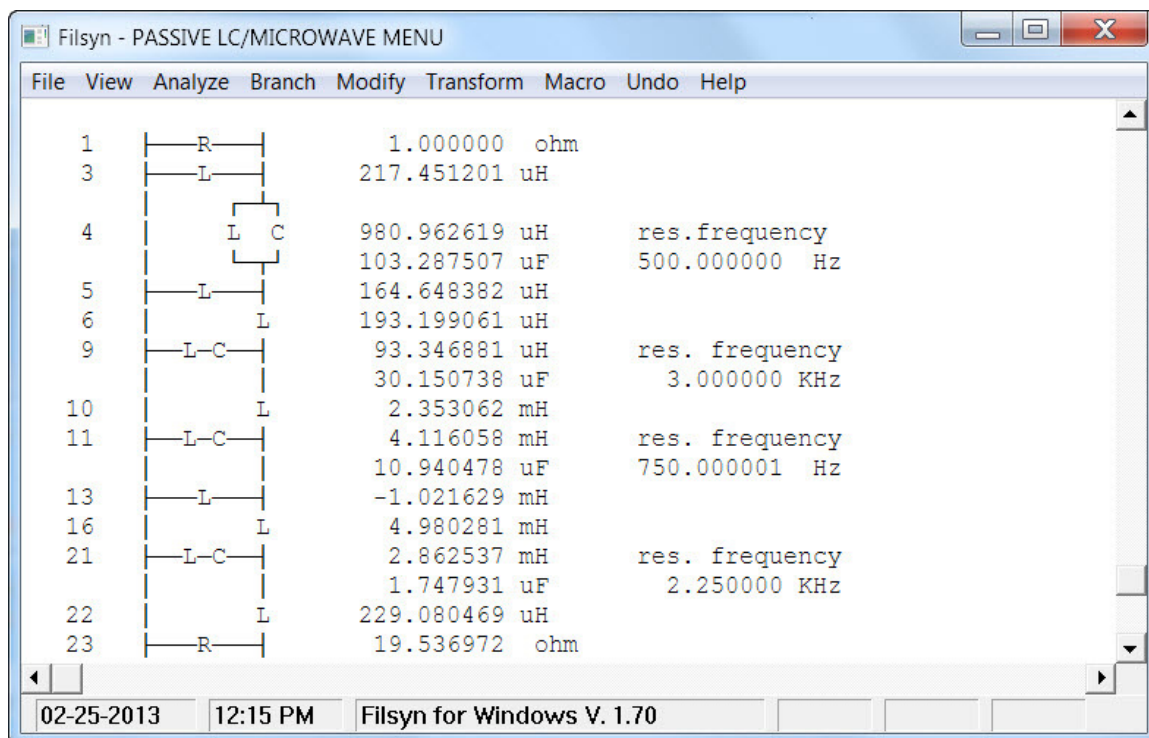
However, flipping around the L section of branches 10 and 11 and combining two series branches into one yields the circuit below, where the negative inductor is indeed grounded.



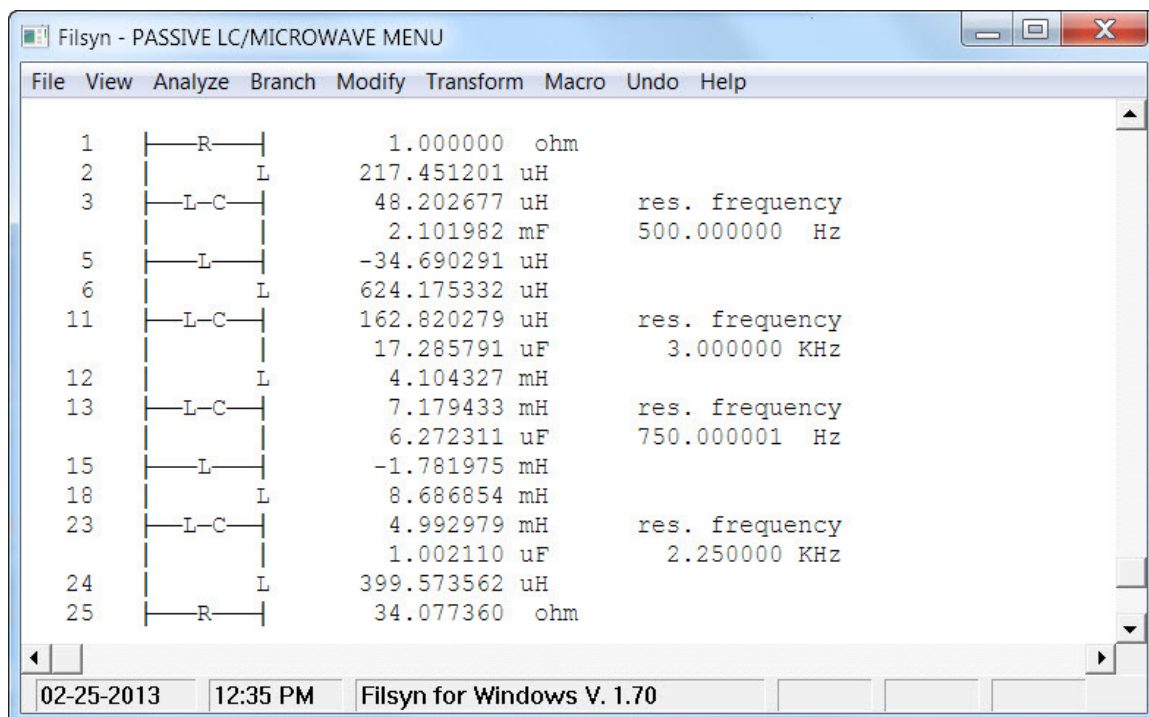
In order to convert branch 4 also to a series resonant branch in shunt, we need a shunt inductor on both sides of it. This can be done as follows. First we introduce a pair of inductors (we selected values of -10uH and +10uH to have them cancel each other) parallel to branch 11. Next we combine the two negative branches to a single branch and take the positive inductor and L_8 and flip this L section around as we have done previously.



This then creates another L section which is also flipped around, finally yielding the circuit below containing another pi section at the front end.



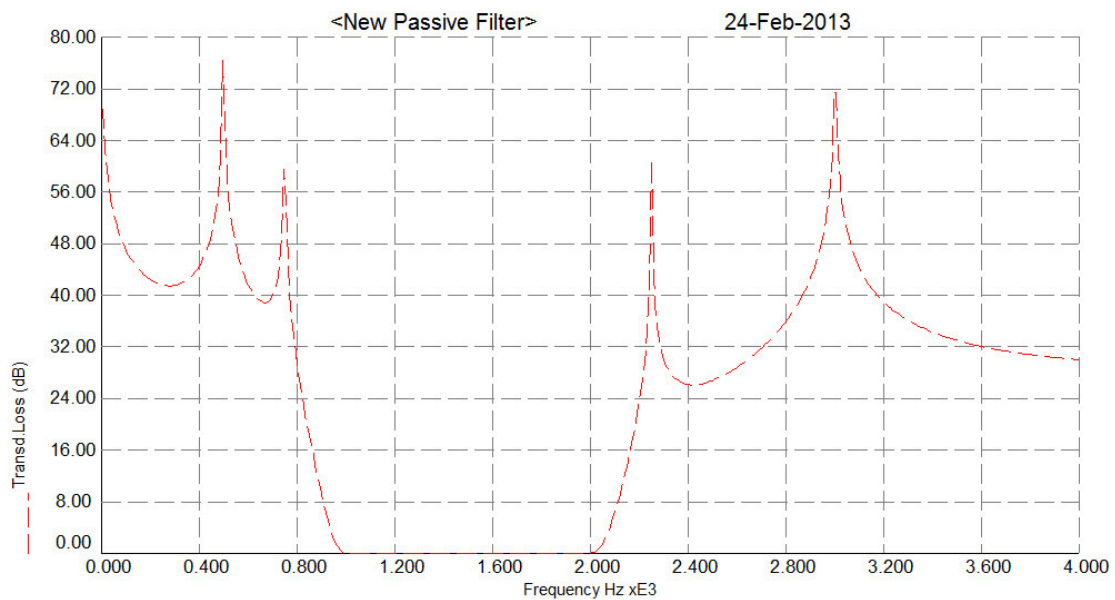
Repeating the tee-to-pi conversion and doing some additional cleanup L section conversions we obtain the final circuit:



As required, the circuit has only four capacitors and they are all grounded at one end (after interchanging the inductors and capacitors in the resonant circuits). The negative inductors are also both grounded at one end, so their implementation also poses no problem. Note also that the number of components is the minimum.

Further steps may be taken to introduce additional shunt inductors across resonant branches and to change the impedance level along the ladder circuit. Since these steps involve arbitrary selections of component values, we leave them to the user.

A final frequency domain analysis confirms that the performance of this filter is exactly as expected. Conversion of this circuit to the appropriate active RC form is straightforward following the reference, and is therefore not considered here.



We have recorded the complete set of steps performed above using the **Macro** menu item and show it here:

```
WinWaitExist ("Filsyn - PASSIVE", 2)
SendMenusTo ("Filsyn", "View Suppress")
SendKey ("{UP} ~")
IntControl (35, 100, 0, 0, 0)
WinWaitExist ("Filsyn - PASSIVE", 2)
SendMenusTo ("Filsyn", "Transform Dual")
WinWaitExist ("Filsyn - PASSIVE", 2)
SendMenusTo ("Filsyn", "Modify Norton Simple")
WinWaitExist ("<New Passive", 2)
SendKey ("12 {TAB} 15 {TAB} 0.000000D+00 {TAB} 0.000000D+00 ~")
SendKey ("~")
WinWaitExist ("Filsyn - PASSIVE", 2)
SendMenusTo ("Filsyn", "Modify Tee-pi")
WinWaitExist ("<New Passive", 2)
```

```
SendKey (" 9 {TAB} {UP} ~")
WinWaitExist("Filsyn - PASSIVE", 2)
SendMenusTo("Filsyn", "Modify Norton Simple")
WinWaitExist ("<New Passive", 2)
SendKey("10 {TAB} 11 {TAB} 0.000000D+00 {TAB} 0.000000D+00 ~")
SendKey("~")
WinWaitExist ("Filsyn - PASSIVE", 2)
SendMenusTo ("Filsyn", "Branch Combine")
WinWaitExist ("<New Passive", 2)
SendKey ("12 {TAB} 14 ~")
WinWaitExist ("Filsyn - PASSIVE", 2)
SendMenusTo ("Filsyn", "Branch Insert")
WinWaitExist ("<New Passive", 2)
SendKey("{UP 9} {DOWN} {TAB} 11 {TAB} -1.000000E-05 ~")
WinWaitExist ("Filsyn - PASSIVE", 2)
SendMenusTo ("Filsyn", "Branch Combine")
WinWaitExist ("<New Passive", 2)
SendKey ("11 {TAB} 13 ~")
WinWaitExist ("Filsyn - PASSIVE", 2)
SendMenusTo ("Filsyn", "Branch Insert")
WinWaitExist ("<New Passive", 2)
SendKey("{UP 9} {DOWN} {TAB} 9 {TAB} 1.000000E-05 ~")
WinWaitExist("Filsyn - PASSIVE", 2)
SendMenusTo("Filsyn", "Modify Norton Simple")
WinWaitExist ("<New Passive", 2)
SendKey(" 8 {TAB} 9 {TAB} 0.000000D+00 {TAB} 0.000000D+00 ~")
SendKey("~")
WinWaitExist("Filsyn - PASSIVE", 2)
SendMenusTo("Filsyn", "Modify Norton Simple")
WinWaitExist ("<New Passive", 2)
SendKey(" 6 {TAB} 9 {TAB} 0.000000D+00 {TAB} 0.000000D+00 ~")
SendKey("~")
WinWaitExist ("Filsyn - PASSIVE", 2)
SendMenusTo ("Filsyn", "Modify Tee-pi")
WinWaitExist ("<New Passive", 2)
SendKey (" 3 {TAB} {UP} ~")
WinWaitExist("Filsyn - PASSIVE", 2)
SendMenusTo("Filsyn", "Modify Norton Simple")
WinWaitExist ("<New Passive", 2)
SendKey(" 4 {TAB} 5 {TAB} 0.000000D+00 {TAB} 0.000000D+00 ~")
SendKey("~")
WinWaitExist ("Filsyn - PASSIVE", 2)
SendMenusTo ("Filsyn", "Branch Combine")
WinWaitExist ("<New Passive", 2)
SendKey (" 6 {TAB} 8 ~")
WinWaitExist ("Filsyn - PASSIVE", 2)
SendMenusTo ("Filsyn", "View Suppress")
```



```
SendKey ("{DOWN} ~")  
WinWaitExist ("Filsyn - PASSIVE", 2)  
SendMenuTo ("Filsyn", "View Show")  
SendKey ("~")
```

The advantage of making this macro recording is that it is now very easy to change the – arbitrary – value that we inserted as the parallel inductors ($\pm 10\text{mH}$) to see if the element values can be made more convenient. These values are the (highlighted) last numbers on the second line following the “**Branch Insert**” menu items. Since this macro file is a pure text file, it can be edited in any text editor. Trying a few values we find, for instance, that using $\pm 2\ \mu\text{H}$ values instead of the $\pm 10\ \mu\text{H}$ we used before, gives us an output termination of 852 ohms instead of the 34 ohms we had before and a couple more tries would get us pretty close to the 1 Kohms we need. This procedure can be further simplified if we save the design data entered at the main data entry window, which makes it very simple to restart the design from the top.

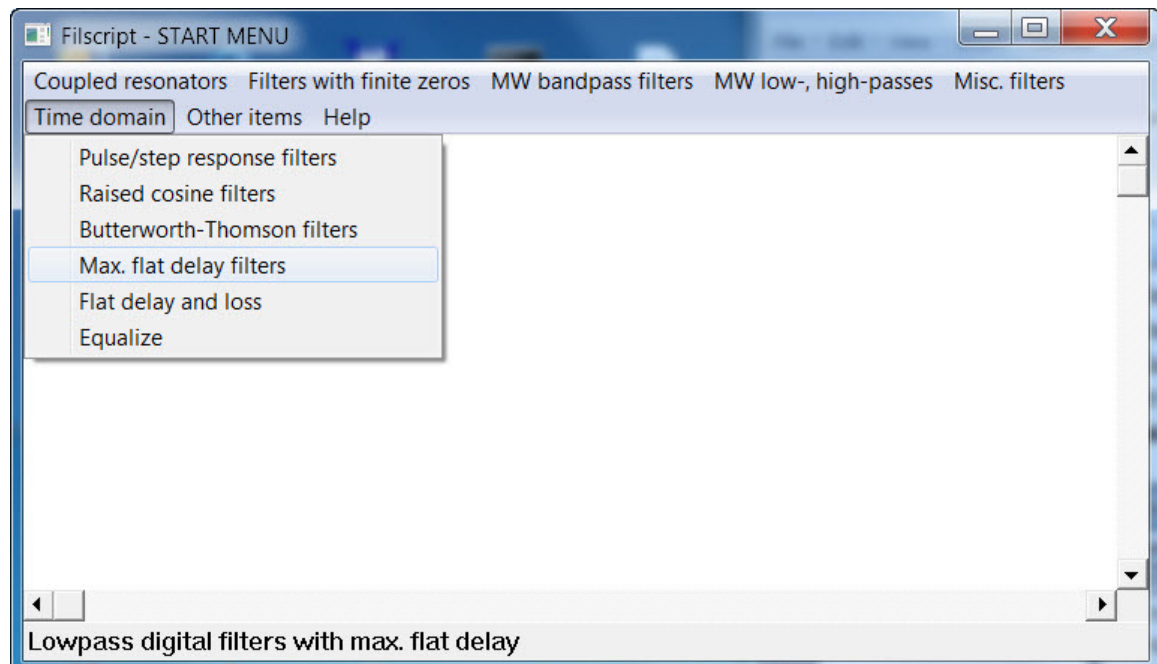
APPLICATION NOTE 15

MICROWAVE OR DIGITAL LOW- OR HIGH-PASS FILTERS WITH FLAT DELAY AND LOSS

1. Indroduction

Thiran (ref.[55]) has presented a method for computing the transfer function poles that yield maximally-flat delay in IIR digital filters. Since the zeros of the function are still arbitrary, the function can be made to have other properties, like flatter passband loss or equal minima stopband. Another fact not recognized before, is the possibility of applying this design feature to microwave filters due to the similarity of these filter's transfer function forms. Historically even more interesting is the fact, that *Abele* (see ref.[63]) several years earlier has found a method to generate microwave filter functions which had maximally flat delay. Of course this can also be applied to IIR digital filters as well. The two methods have not been critically compared, but superficially they seem to be quite different. Both methods need an arbitrary parameter, but the one for *Thiran* is an integer smaller than the degree, while the one for *Abele* is a real number greater than the degree. For further information see ref.[64].

In any case, we have selected *Thiran's* method here to generate a whole group of functions, combining microwave or IIR digital filters with maximally flat delay in the passband, coupled with either flat passband of varying flatness or equal minima type stopband. Furthermore these IIR digital lowpass filters may even be convertible to a similar highpass design. All of these options are now available in the **Filsyn** program in a script that can be reached by calling the **Filscript** executable and selecting the **Time domain** -> **Max. flat delay filters** menu option:



The data input screen follows:

The **filter degree** must be between 5 and 30. Below degree 5 hardly any of the features make any sense and toward degree 30 we will probably run into numerical problems. In fact, in the microwave implementation the program will offer to switch to the manual synthesis mode which is more useful if numerical problems arise.

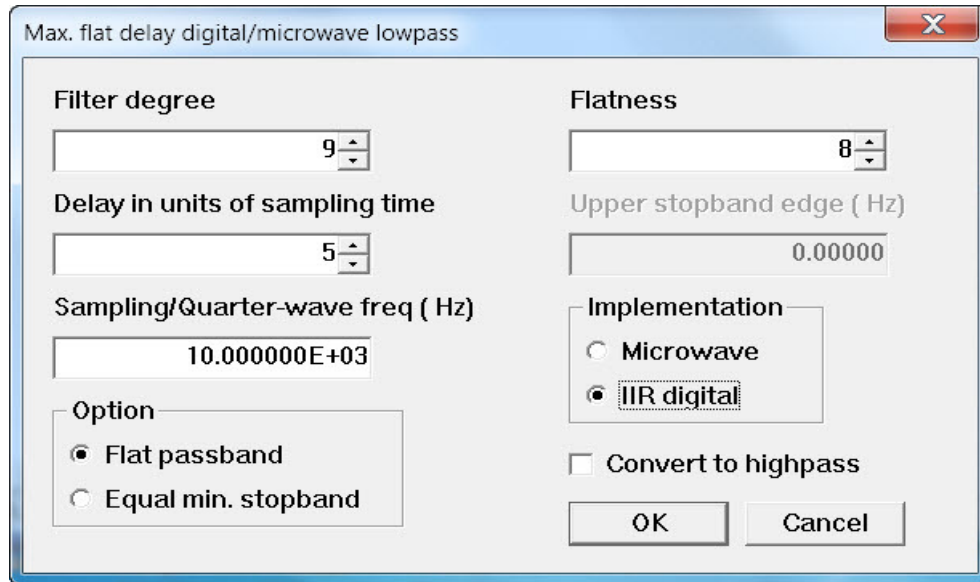
The **delay in units of sampling time** is an integer that influences the delay and must be an integer less than the degree. The filter delay at zero frequency will be:

$$\text{Delay} = (K + N/2)/f_s$$

where N is the filter degree, K is this integer parameter and f_s is the sampling frequency. This is exact for IIR digital implementation, but only approximate for microwave filters where the delay will be somewhat smaller. The **flatness** controls the passband loss and must be an integer, a multiple of 4 and less than the degree. The rest of the data are evident.

2. Example 1

Consider the following digital filter specification: it's a 9th degree filter with an 8th order flat passband and a delay parameter of 5. Sampling frequency is 10 KHz. Note that the more flat the passband is the less attenuation it provides in the stopband, hence this filter is not going to be much of one.



Max. flat delay digital/microwave lowpass

Filter degree: 9

Flatness: 8

Delay in units of sampling time: 5

Upper stopband edge (Hz): 0.00000

Sampling/Quarter-wave freq (Hz): 10.000000E+03

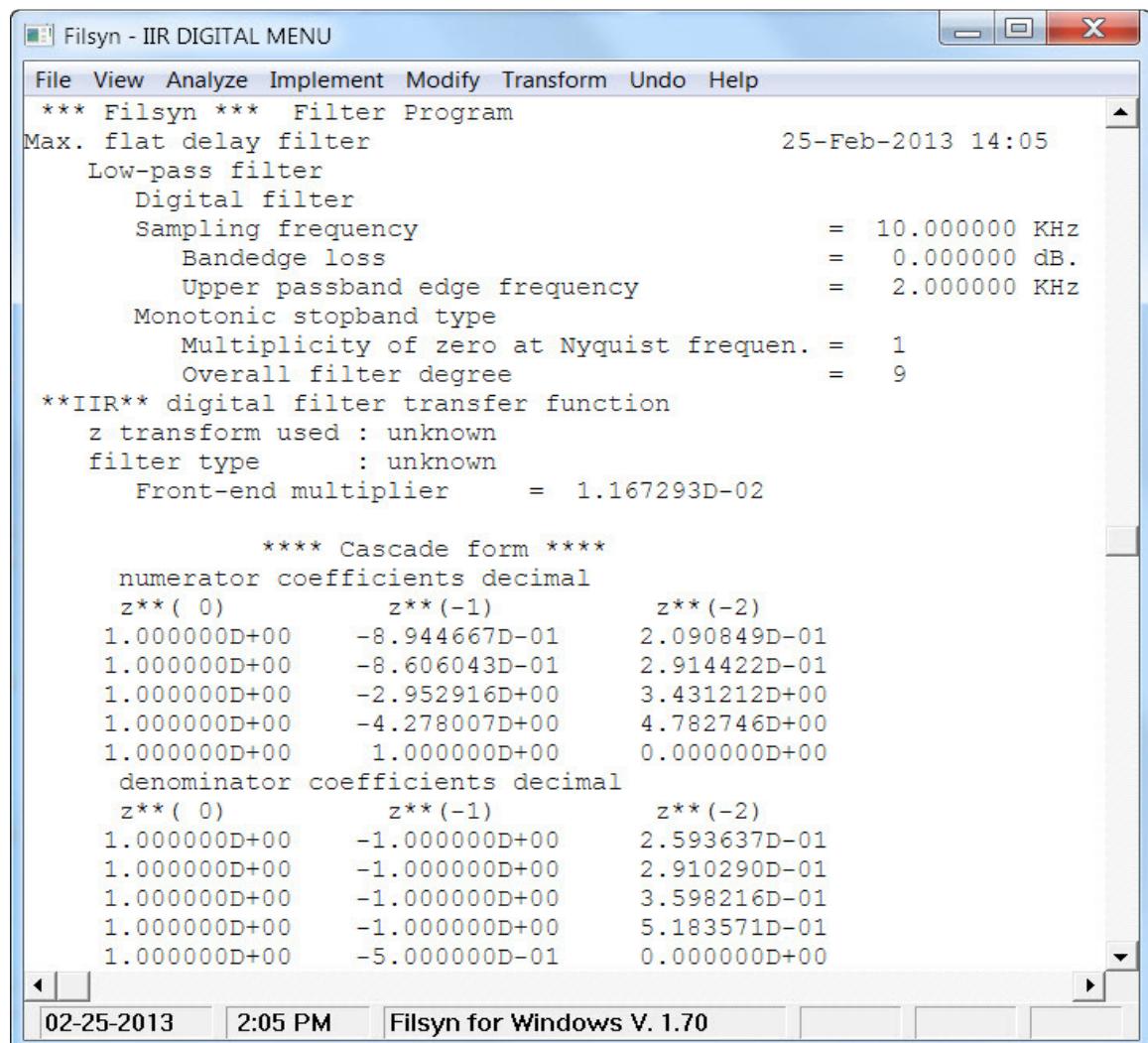
Implementation: ☒ IIR digital, ☐ Microwave

Option: ☒ Flat passband, ☐ Equal min. stopband

☐ Convert to highpass

OK Cancel

After the script stops running, we get the filter data:



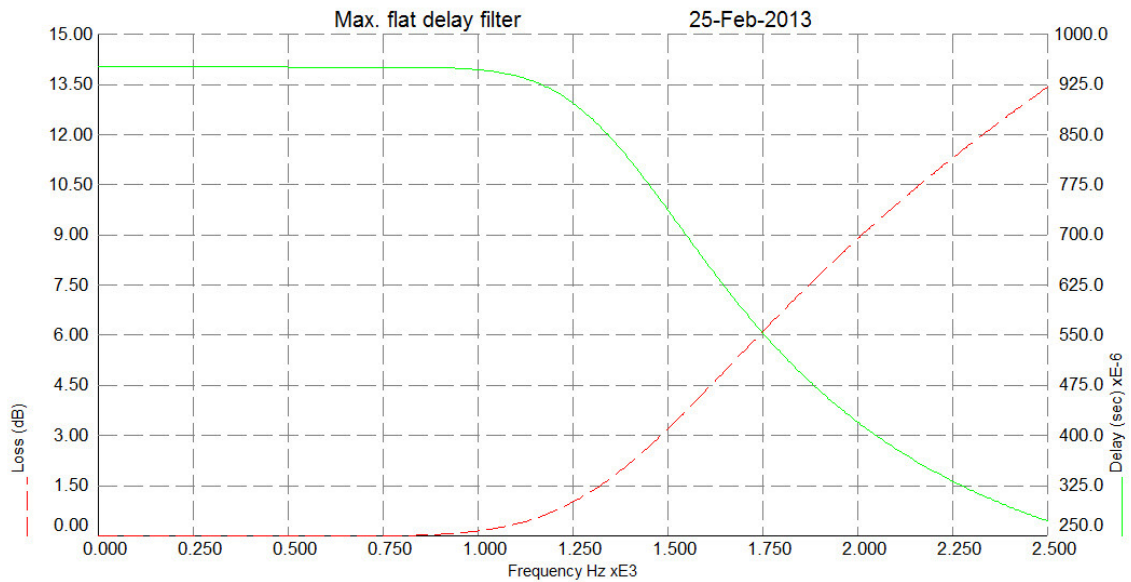
```

File View Analyze Implement Modify Transform Undo Help
*** Filsyn *** Filter Program
Max. flat delay filter                                     25-Feb-2013 14:05
  Low-pass filter
    Digital filter
      Sampling frequency                                = 10.000000 KHz
      Bandedge loss                                    = 0.000000 dB.
      Upper passband edge frequency                    = 2.000000 KHz
      Monotonic stopband type
        Multiplicity of zero at Nyquist frequen.      = 1
        Overall filter degree                         = 9
    **IIR** digital filter transfer function
      z transform used : unknown
      filter type      : unknown
      Front-end multiplier = 1.167293D-02

      **** Cascade form ****
      numerator coefficients decimal
        z**( 0)      z**(-1)      z**(-2)
        1.000000D+00 -8.944667D-01 2.090849D-01
        1.000000D+00 -8.606043D-01 2.914422D-01
        1.000000D+00 -2.952916D+00 3.431212D+00
        1.000000D+00 -4.278007D+00 4.782746D+00
        1.000000D+00 1.000000D+00 0.000000D+00
      denominator coefficients decimal
        z**( 0)      z**(-1)      z**(-2)
        1.000000D+00 -1.000000D+00 2.593637D-01
        1.000000D+00 -1.000000D+00 2.910290D-01
        1.000000D+00 -1.000000D+00 3.598216D-01
        1.000000D+00 -1.000000D+00 5.183571D-01
        1.000000D+00 -5.000000D-01 0.000000D+00
  
```

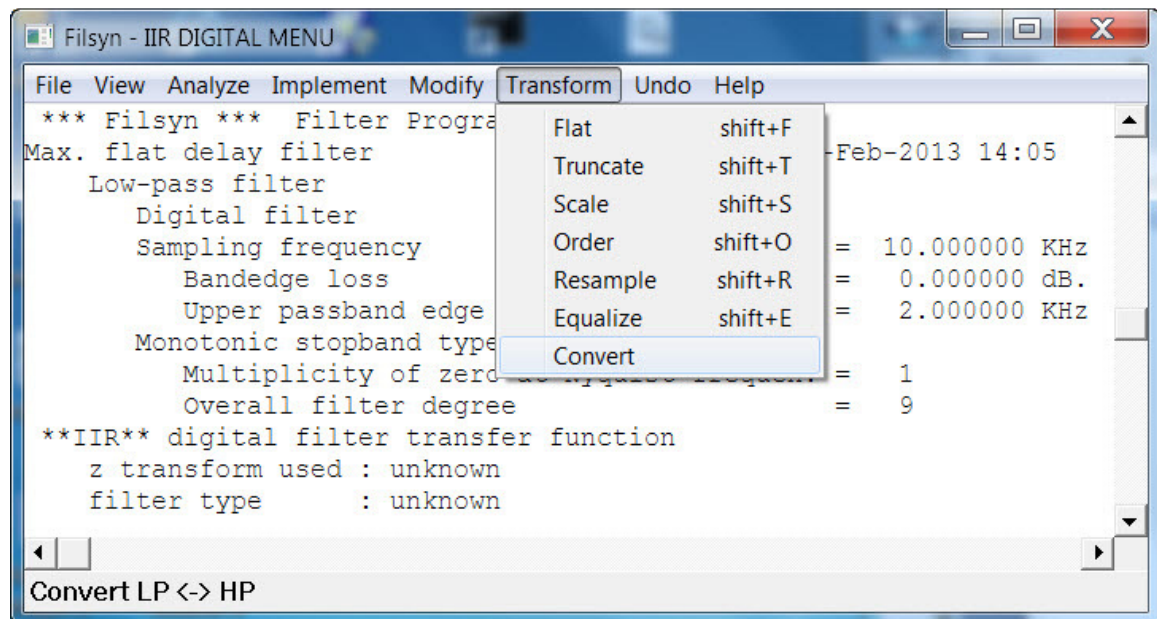
02-25-2013 2:05 PM Filsyn for Windows V. 1.70

Analyzing this in the frequency domain we get:



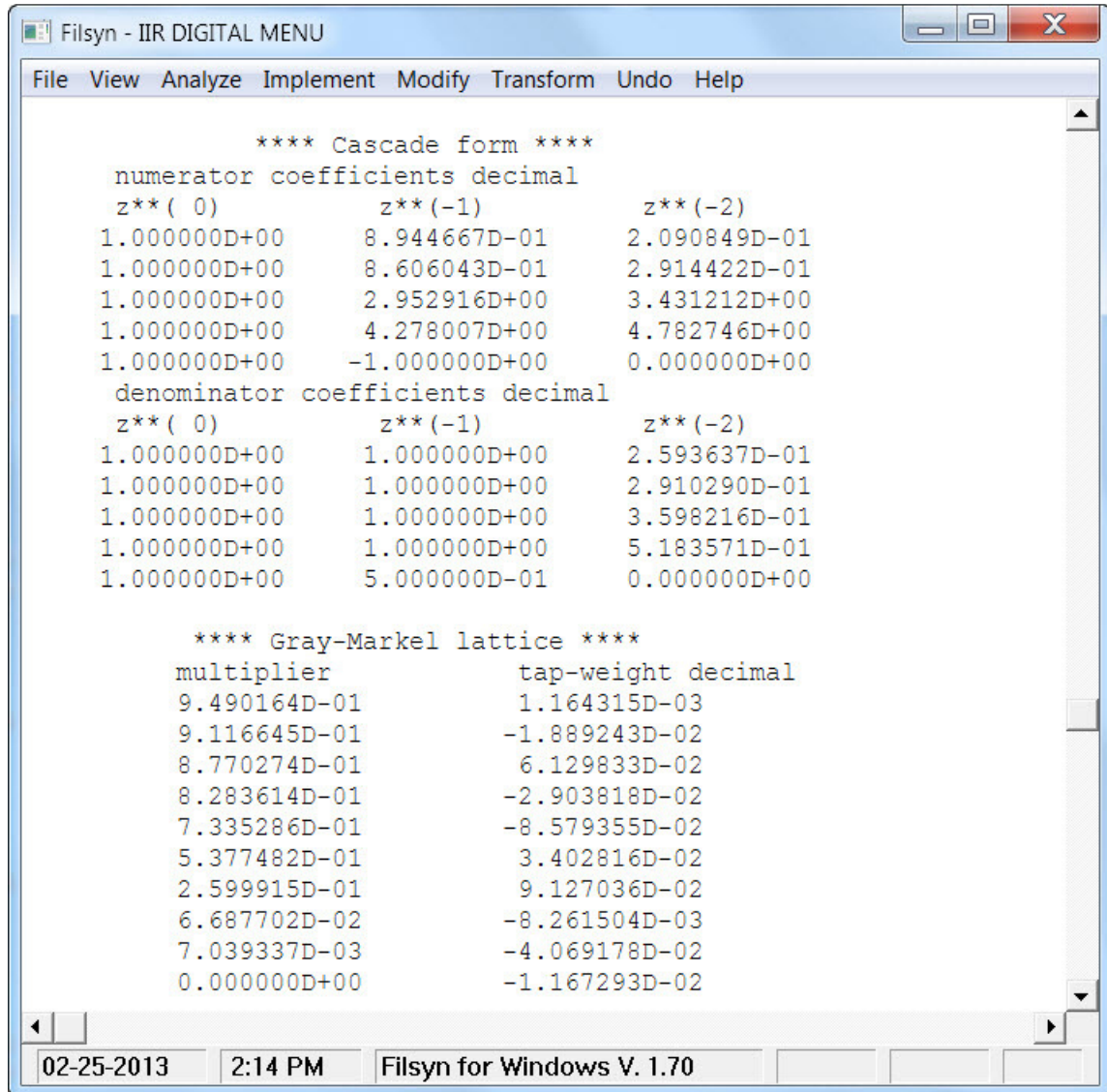
The flat delay is exactly 0.95 msec and the loss is also very flat. The loss is not much, at 2.5 KHz it is barely 13 dB.

The interesting thing is that we can flip this filter from lowpass to highpass and we can perform this by simply changing the signs of the linear terms of both the numerator and denominator factors. Of course this is not restricted to this filter but is applicable to *any* IIR digital filter. As a result, we have introduced a new menu item **Convert** in the **Transform** set:



If we select the **Convert** to highpass option in the menu above, this conversion will be done automatically. Note also that this operation is bilateral, i.e. it works also the other direction; it can even be applied to bandpass filters where it inverts the frequency scale. Applying it twice leaves the function unchanged.

Applying this menu item we get the new function:



```

Filsyn - IIR DIGITAL MENU
File View Analyze Implement Modify Transform Undo Help

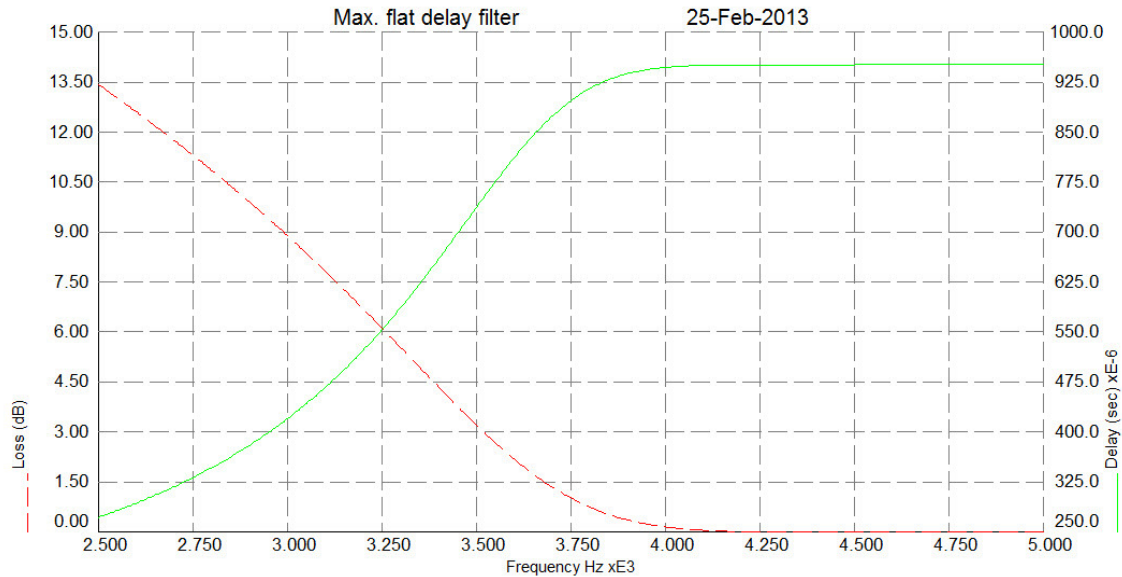
**** Cascade form ****
numerator coefficients decimal
z**( 0)      z**(-1)      z**(-2)
1.000000D+00  8.944667D-01  2.090849D-01
1.000000D+00  8.606043D-01  2.914422D-01
1.000000D+00  2.952916D+00  3.431212D+00
1.000000D+00  4.278007D+00  4.782746D+00
1.000000D+00 -1.000000D+00  0.000000D+00
denominator coefficients decimal
z**( 0)      z**(-1)      z**(-2)
1.000000D+00  1.000000D+00  2.593637D-01
1.000000D+00  1.000000D+00  2.910290D-01
1.000000D+00  1.000000D+00  3.598216D-01
1.000000D+00  1.000000D+00  5.183571D-01
1.000000D+00  5.000000D-01  0.000000D+00

**** Gray-Markel lattice ****
multiplier      tap-weight decimal
9.490164D-01    1.164315D-03
9.116645D-01    -1.889243D-02
8.770274D-01    6.129833D-02
8.283614D-01    -2.903818D-02
7.335286D-01    -8.579355D-02
5.377482D-01    3.402816D-02
2.599915D-01    9.127036D-02
6.687702D-02    -8.261504D-03
7.039337D-03    -4.069178D-02
0.000000D+00    -1.167293D-02
    
```

02-25-2013 2:14 PM Filsyn for Windows V. 1.70

where we also computed the Gray-Markel representation of this new function. The analysis results shown below confirm that we have a highpass indeed, the function is flipped around 2.5 KHz.

By the way, you may notice that in the cascade implementation *all* the linear terms in the denominator represent a real part of 0.5 for *all* roots. These numbers are correct but difficult to explain.



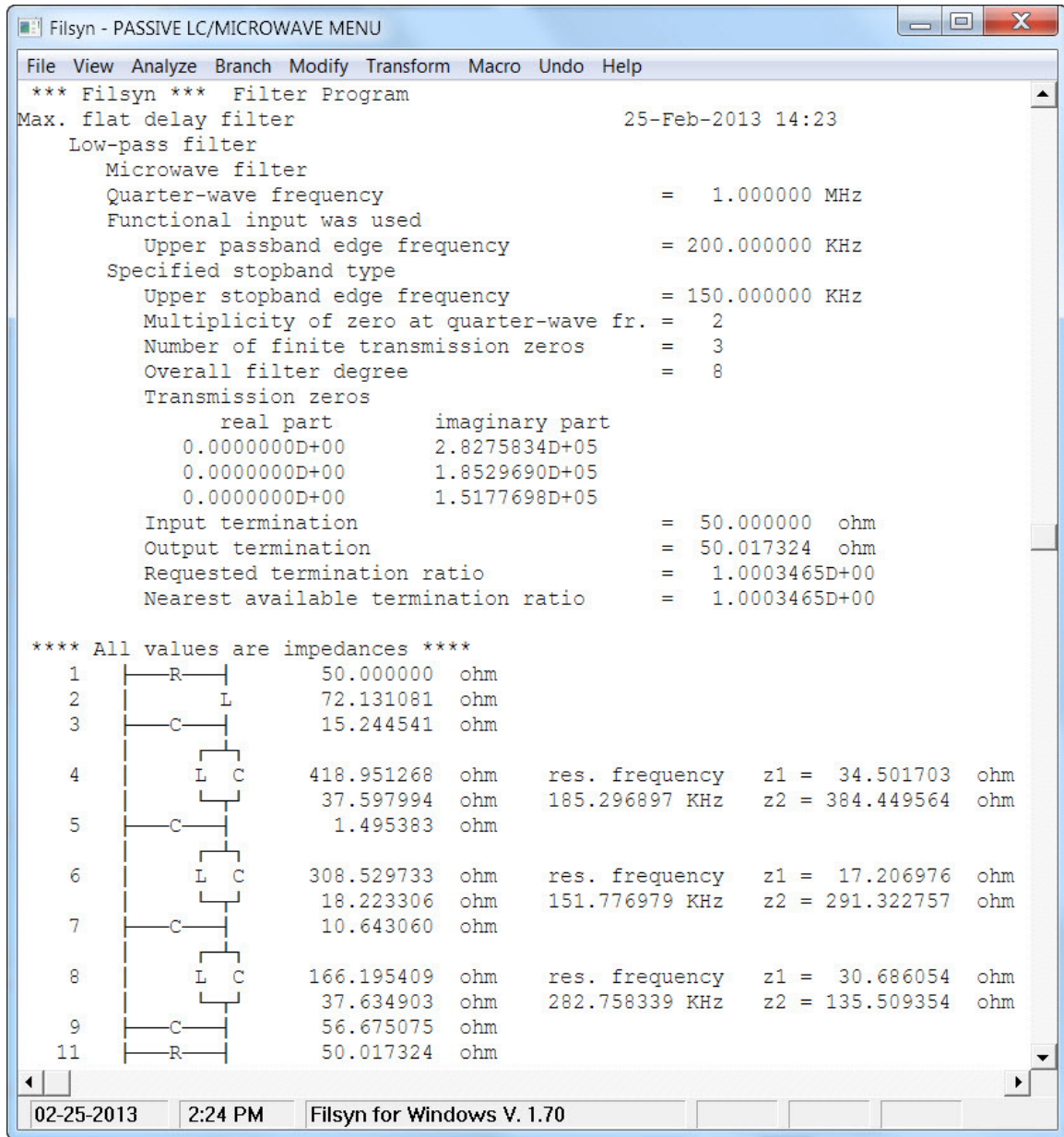
3. Example 2

Consider now the following microwave filter set of specification:

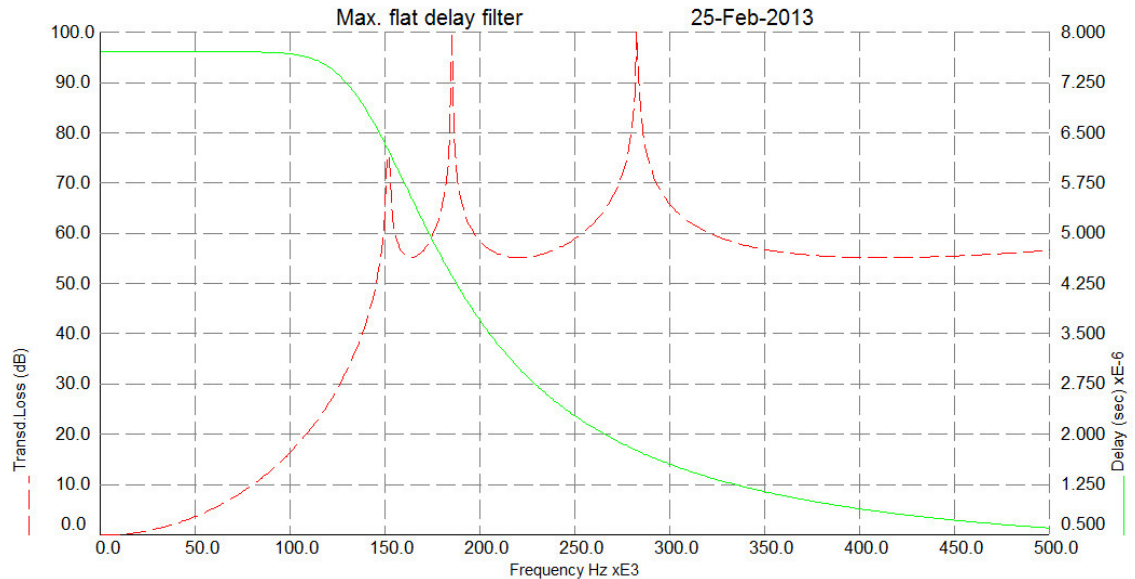
The screenshot shows a dialog box titled 'Max. flat delay digital/microwave lowpass'. It contains the following settings:

- Filter degree: 8
- Delay in units of sampling time: 4
- Sampling/Quarter-wave freq (Hz): 1.000000E+06
- Flatness: 0
- Upper stopband edge (Hz): 150K
- Implementation: ☒ Microwave, ☐ IIR digital
- Option: ☐ Flat passband, ☒ Equal min. stopband
- Buttons: OK, Cancel

This is an 8th degree filter that will have about 8 μ sec ($4 + 8/2$) delay at zero frequency and an equal minima type stopband from 150 Hz up. The design stops at the point:



At one point the manual synthesis option was offered, but as seen above, it was not needed, the computer selection was accurate enough. The analysis shows very accurate results as well:



The delay is a flat 7.70 μ sec close to what we predicted; the stopband loss has an equal minimum of 55 dB.

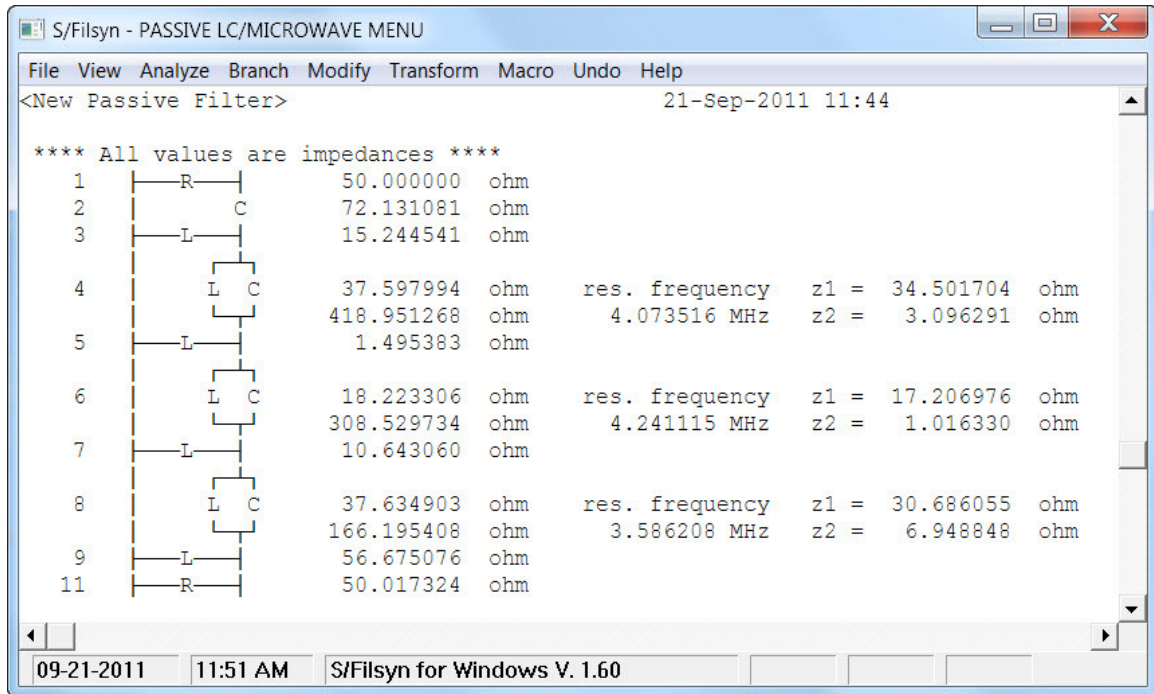
These examples only present a few of the options and combinations, some of which are not available, but if not, they are clearly indicated.

Comment: We have already mentioned above and in *Application Note 6* that IIR digital and microwave lowpass filters can be converted into highpass and vice versa and that this conversion for digital filters is very simple and has already been implemented in **Filsyn**. For microwave filters we mentioned the mathematical process of inverting the variable and performing the synthesis again as a way of obtaining the conversion. However there is a much simpler process available which is to take the ladder or lattice implementation and replace all inductors (short-circuited stubs) by capacitors (open-circuited stubs) of the same impedance and vice versa.

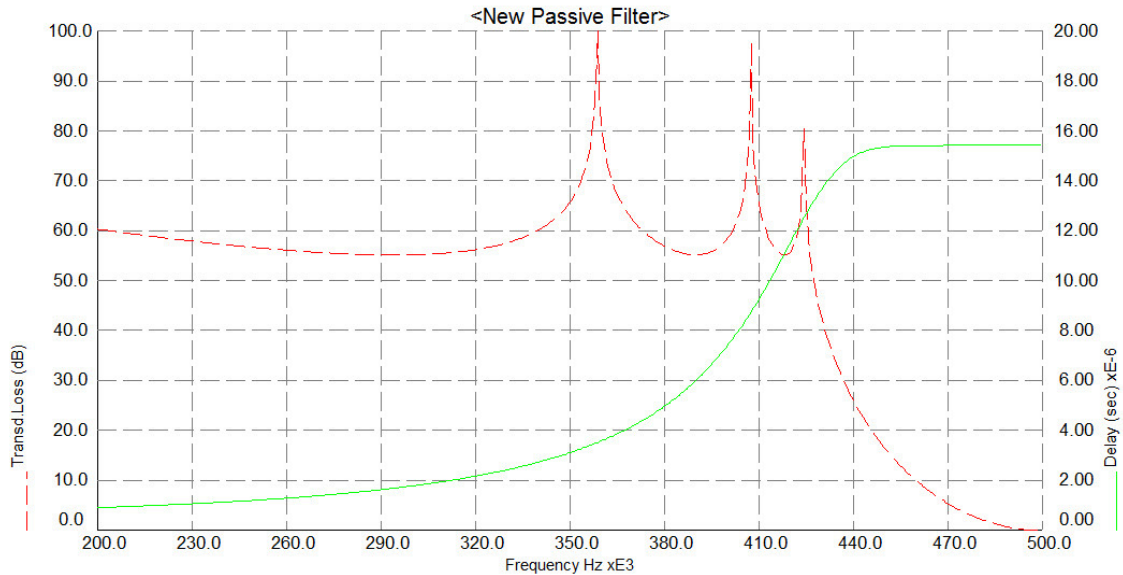
This procedure has *not* been implemented yet, because if the circuit contains twin-T sections, then the converted structure may contain twin-T sections of the form that the program does not recognize yet.

For your information we consider here the microwave lowpass design of example 2 above. We manually convert this to a highpass (actually a bandpass) as per the description above. Note that we need to double the quarter-wave frequency to 20 MHz. We then get:

MW or IIR low- or high-pass filter with flat delay and loss



and this yields the following highpass behavior:



This is exactly the same as the lowpass, flipped around.

APPLICATION NOTE 16

ANALOG LOWPASS FILTER WITH FLAT DELAY AND LOSS

1. Introduction

Occasionally one might need a passive LC or active RC lowpass filter that has a flat delay and loss in the passband but has no requirements concerning its stopband behavior. The transfer function of such a circuit can be generated as follows. We start with a constant delay polynomial, generated in the **Delayline/equ** segment of the program. This should be a maximally flat approximation and will be the denominator polynomial of the transfer function we need. This is followed by generating a numerator polynomial that does not contribute to the delay but compensates the magnitude of the denominator. Such a polynomial will have complex quadruplets of zeros and consequently will have a degree that is a multiple of four but less than that of the denominator.

The mathematical details of this procedure are described in reference 52. In order to implement this procedure, we have developed a script file that performs it and built it into the `filscript.exe` script procedure. Numerical problems may occur in which case the program will recommend that the input data should be changed. The reason for these problems is that a number of complex computations is performed on various polynomials of which we need to have complex zeros, but this is never guaranteed.

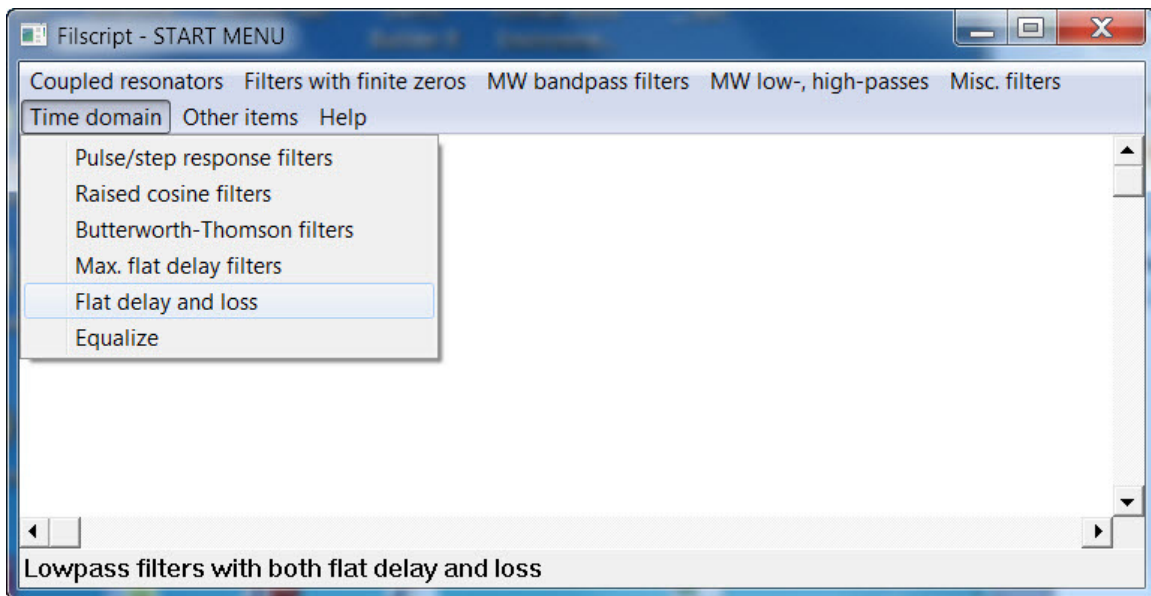
Concerning the implementation of these filters, while the active-RC implementation has no problems, the passive LC implementation will necessarily have bridged-T or twin-T sections in them to realize the complex zeros we generated and some of these will contain a negative inductor. However they will appear bounded by two other positive inductors and can be implemented as a pair of coupled inductors. Finally occasionally we will find other negative components in the computer-generated circuit. One option in this case is to use the manual synthesis method to eliminate these and this option is facilitated by a prompt that offers this option at the proper location. If this does not succeed, the only other option we have is to use the active-RC implementation if that is feasible.

2. Data input

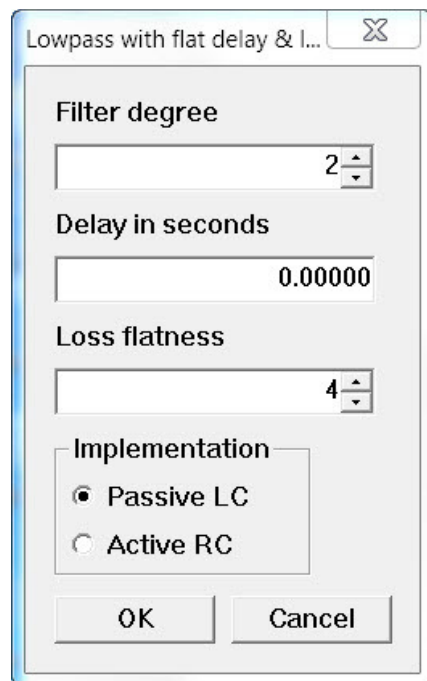
The input data menu is quite simple:

The overall degree must be between 6 and 20; filters of degree lower than six cannot be used in this process and beyond 20 is unlikely to be needed. The starting delay will be maximally flat. Next we need to specify the required loss flatness which must be a multiple of four and of course a minimum of four since otherwise we have no flat loss at all. The maximum flatness is determined by the fact that it must be lower than the overall filter degree. Finally we select the implementation, either passive LC or active RC.

The **Start** menu has this script under the **Time domain** header with the name **Flat delay and loss**:



Selecting this menu item we get the data entry window:



Using as an introductory example an 11th overall degree, 200 μ sec delay, 8th order flatness and implement it in passive LC form:

Lowpass with flat delay & l... X

Filter degree

11

Delay in seconds

200.000000E-06

Loss flatness

8

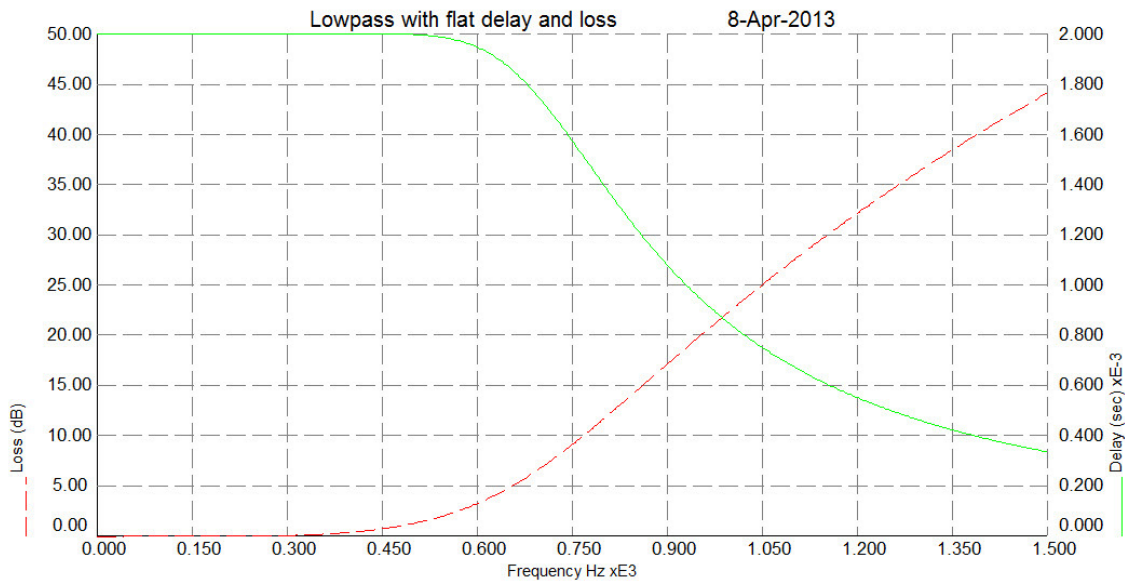
Implementation

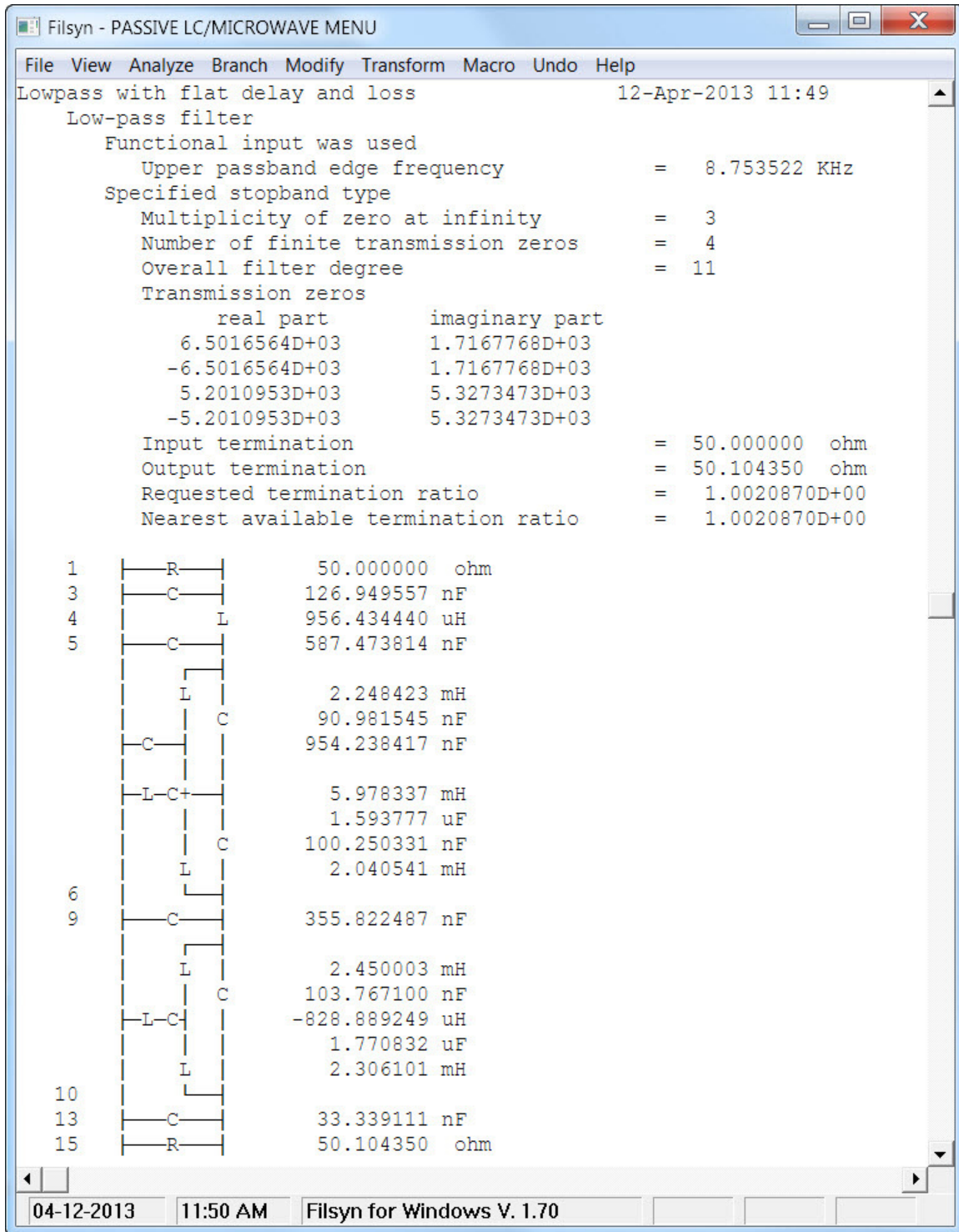
☒ Passive LC
 ☐ Active RC

OK

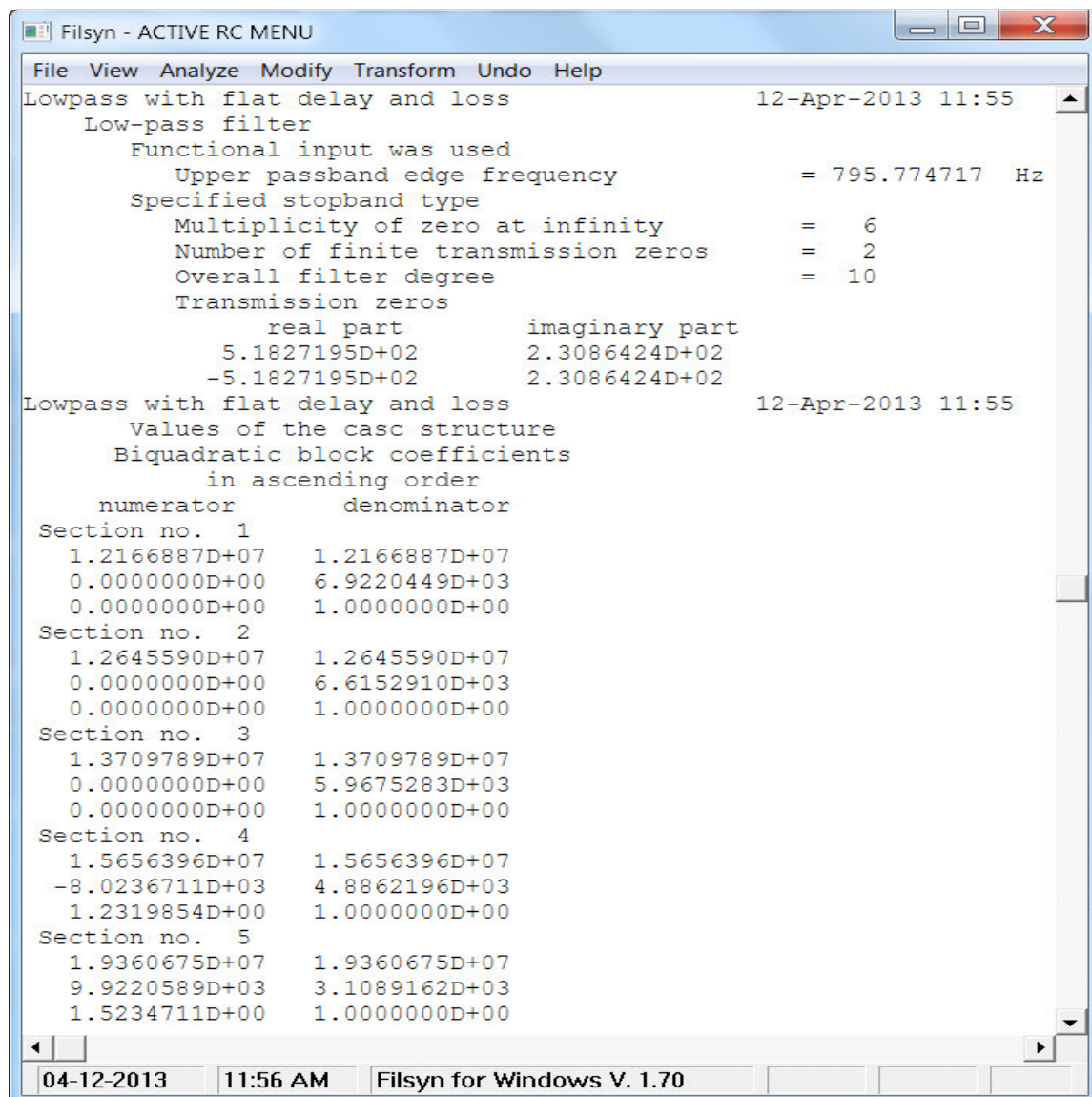
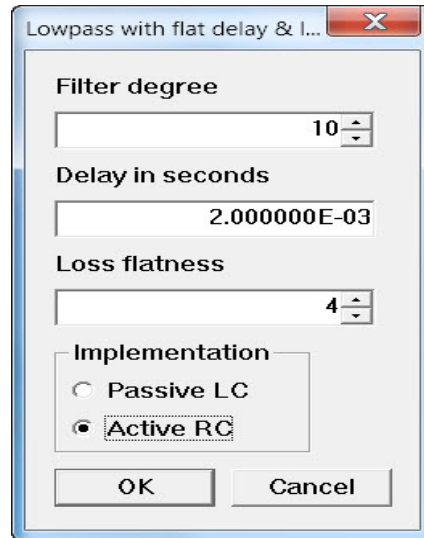
Cancel

You will notice the large number of steps the program goes through before it stops at the point shown on the next page. There are a large number of computational steps to perform, hence occasional errors can easily accumulate. The resulting implementation in this case is realizable and the analysis result shown is quite impressive.

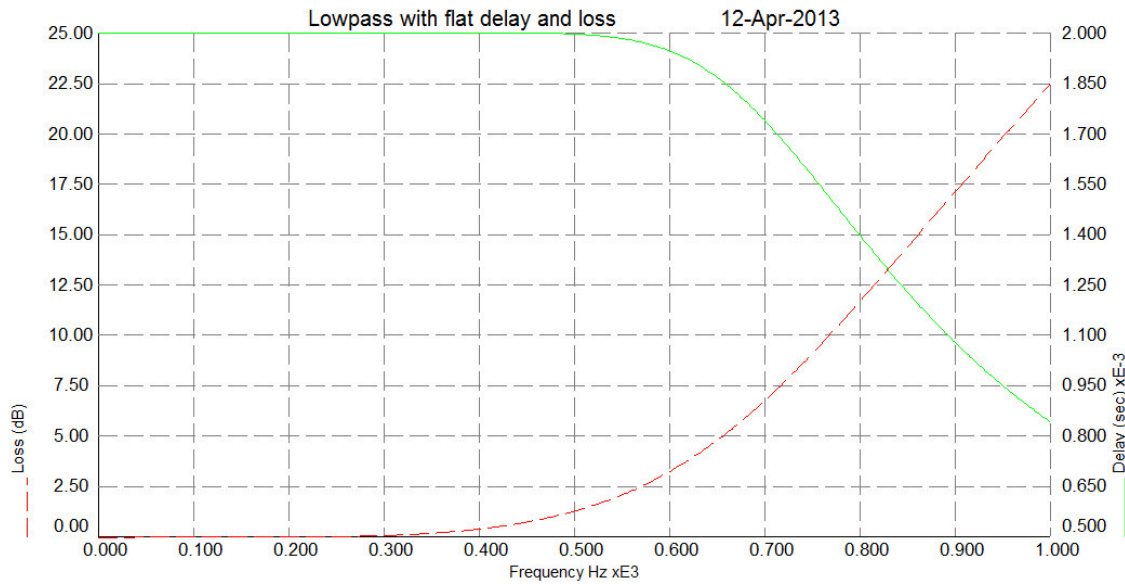




We will repeat this design except specifying a 10th degree filter and a 4th order loss flatness and active RC implementation:



yielding the above set of filter data and the analysis results shown below:



3. Comments

As we have pointed it out before, the delay flatness acceptable in this script is only the maximally flat (Bessel) type, while the equal ripple type is not available. The reason for this is that while the mathematical procedure of this design method is fully understood and works properly, there is no guarantee that the resulting transfer function numerator will yield an acceptable loss shape over the complete frequency axis. The surprising and so far unexplained fact is that the results are nearly always excellent for maximally flat starting functions and nearly always unacceptable for the equal ripple type ones.

INDEX**A**

Abele 356, 582
 active RC 23, 35, 44, 48, 150, 188, 243, 263, 279, 388, 411, 468, 489, 519
 admittance 103, 175, 365, 416, 424
 allpass 40, 168, 191, 360
 amplifier 504
 operational 152, 357, 503, 577
 summing stage 509
 transconductance 151, 503
 analysis 5, 29, 34, 39, 49, 107, 151, 221, 236, 245, 402, 422, 542
 frequency domain 19, 26, 55, 73, 108, 157, 191, 207, 212, 246, 256, 269, 299, 495, 523, 572
 time domain 3, 35, 107, 157, 188, 236, 241, 247, 552
 impulse-response 36, 107, 157, 167, 192, 220, 238
 step-response 36, 107, 157, 220, 238
 antimetric 364
 asymmetric 385

B

band elimination 119, 170
 band-reject 23, 31, 39, 119, 168, 445
 bandpass 1, 12, 18, 24, 33, 59, 63, 78, 83, 98, 137, 208, 234, 297, 30
 Bessel 36, 42, 271, 389, 393, 407
 bilinear 82, 159, 167, 171, 191, 264, 411, 516
 biquad(ratic) 167, 190, 357, 498, 532
 bridged-T 41, 95, 109, 118, 245, 293, 426
 Brune 108, 562
 Butterworth 3, 24, 29, 36, 96, 120, 162, 404, 415, 445
 -Thomson 3, 36, 41

C

Cauer 42
 cascade 22, 27, 40, 135, 147, 158, 167, 170, 190, 196, 201, 213, 236, 250, 269, 454, 500, 513, 524, 591

characteristic function 39, 82, 90, 95, 191, 386, 411, 533
 Chebyshev 24, 31, 36, 42, 98, 120, 405, 445, 536
 combline 17, 297
 computer-generated 54, 99, 128, 351, 361, 386, 393, 418, 435, 448, 455, 577, 596
 conventional (non-parametric) 11, 83, 100, 169, 378, 385, 410, 454, 575
 coupled resonators 1, 4

D

delay equalizer 38, 45, 48, 95, 105, 151, 174, 187, 209, 243, 254, 289, 351, 426, 468, 473, 526
 delay line 39, 45, 59, 77, 174, 179, 243, 271, 287, 296
 differential allpass 40, 168, 191
 diplexer 19, 413, 425
 documentation 38, 42, 104, 107, 151, 157, 187, 237, 298, 300, 308, 314, 327

E

edge-coupled 17, 115, 309
 elliptic 24, 31, 41, 59, 64, 92, 120, 170, 177, 195, 320, 340, 387, 396, 399, 406, 415, 433, 445
 equal minima 39, 42, 83, 272, 339, 348, 387, 393, 405, 433, 490, 498, 587, 592
 equal ripple 39, 42
 delay 74, 87, 90, 271, 275, 412, 430, 486, 601
 loss 48, 137, 213, 219, 234, 297, 327, 386, 405, 535
 equalizer, delay: see 'delay equalizer'

F

flatness 29, 339, 587, 596, 601
 Fleischer-Laker 159, 164
 follow-the-leader (FLF) 40, 148, 165, 367, 511, 525, 532
 functional input 39, 41, 48, 82, 95, 105, 152, 187, 243, 318, 331, 338, 344, 364, 386, 393, 411, 492, 535

G

Gauss(ian) 3, 31, 41, 407
global units 4, 36, 43
Gray-Markel 40, 168, 190, 199, 346, 591

H

highpass 1, 16, 19, 24, 35, 92, 98, 109,
137, 143, 148, 167, 194, 234, 361, 377,
387, 404, 414, 422, 468, 473, 477, 483,
577, 587
Hilbert 219, 228

I

immittance 54, 361, 377
impedance 6, 9, 18, 21, 27, 103, 107,
110, 135, 146, 174, 292, 309, 362, 375,
393, 403, 409, 416, 423, 443, 466, 577,
584, 594
 inverter 447, 451
 matching 27, 137, 352, 533
 transformation 109, 377, 387,
423, 492
implementation 21, 48, 54, 59, 74, 90,
100, 138, 147, 157, 168, 174, 183, 190,
206, 213, 243, 254, 269, 279, 287, 297,
307, 331, 426, 435, 468, 482, 492, 503,
516, 523, 542, 566, 583, 591, 596
impulse
 invariant 167, 190
 response 36, 39, 107, 157, 167,
221, 234, 238
interdigital 17
interstage 27, 41
inverse Chebyshev 31, 405
iteration 68, 210, 275, 322, 329, 455,
487, 500, 516, 526

K

Kaiser 234
Kaiser-Sandberg 190

L

ladder 6, 27, 40, 54, 59, 91, 95, 109,
174, 181, 361, 387, 394, 434, 445, 454,
467, 540, 544, 584, 594

lattice 27, 40, 67, 70, 74, 109, 168, 174,
216, 245, 454, 460, 594
 Gray-Markel 40, 168, 190, 199,
346, 591
leapfrog (LF) 40, 147, 166, 246, 280,
502, 508, 516, 520, 532
Levy 108, 533
linear phase 87, 133, 219, 228, 233, 236,
240, 271, 287
 bandpass 78
 highpass
 lowpass 486
Long-Trick 190
lowpass 18, 29, 33, 59, 63, 74, 83, 90,
96, 100, 119, 122, 126, 137, 143, 147,
162, 167, 171, 177, 184, 194, 220, 230,
240, 250, 271, 320, 334, 361, 387, 396,
399, 404, 412, 417, 421, 425, 431, 445,
451, 468, 476, 485, 492, 533, 549, 577,
587, 594
lowpass-to-highpass 167
lower sideband 434

M

Matlab 104, 132
maximally flat 39, 42, 96
 delay 36, 250, 407, 495, 587,
596, 601
 loss 31, 386, 404, 533
Minnis 18
monotonic 29, 39, 42, 98, 100, 377, 404
multiplexer 40, 100, 403, 413, 415, 423

N

natural modes 48, 87, 243, 296, 411,
481, 495
Norton 109, 115, 118, 134, 353, 381,
387, 423, 457, 462, 542, 559
 See also: impedance
 transformation 109, 377, 423,
492
Nyquist 174, 191, 220, 234, 240, 470

O

optimize 61, 166, 193, 295, 297, 300,
317, 326, 336, 349, 502, 520, 541

optimization 54, 62, 75, 95, 149, 194,
210, 213, 277, 292, 297, 300, 305, 321,
330, 334, 340, 350, 396, 416, 425, 435,
455, 533

P

Papoulis 29, 41
parametric 5, 11, 41, 61, 83, 100, 169,
364, 378, 385, 423, 492, 541, 575
placer routine 1, 21, 40, 50, 59, 64, 67,
388, 396, 399, 426, 435, 454, 468, 499,
517, 520, 541
poles 48, 83, 166, 254, 262, 326, 334,
338, 361, 411, 468, 473, 479, 490, 603
 transfer function 243, 250, 318,
320, 350, 587
polynomial 41, 82, 87, 95, 147, 167,
236, 246, 259, 364, 368, 377, 386, 410,
596
postprocessor 37, 297, 301, 309
predistortion 41, 91, 95, 364, 396, 400,
403, 409, 425
preprocessor 318, 321, 344
prewarp 164, 174, 263, 269, 516, 518

Q

quadratic 147, 167, 193, 207, 262, 468,
477
quadruplet 12, 41, 95, 109, 118, 243,
321, 334, 364, 411, 427, 564, 573, 596
quarter-wave 16, 21, 40, 143
 frequency 98, 100, 126, 146, 174,
177, 181, 378, 388, 411, 477, 594
quasi-elliptic 12, 15, 21, 148, 184

R

Remez 48
reflection coefficient 41, 51, 89, 389,
409, 411
response 21, 107, 188, 201, 238, 312,
395, 410
 frequency domain 7, 10, 94, 138,
194, 234, 302, 339, 453, 470
 time domain 3, 35, 39, 107, 157,
167, 189, 193, 219, 223, 234, 241, 553

return loss 5, 10, 51, 57, 92, 107, 316,
396, 409

Richard transformation 411

S

sampling 163, 181
 frequency (rate) 159, 167, 170,
174, 177, 181, 187, 194, 220, 226, 234,
257, 260, 294, 340, 411, 468, 473, 516,
549, 588
Scasy 151
Scompact 298, 308, 314
script files 1, 16, 41, 105, 120, 134, 356,
412, 434
shifted-bandpass 36, 47, 63, 66, 78, 271,
279
short-circuit 17, 40, 101, 361, 375, 413,
417, 423
sloping bandpass 27
sloping passband 3, 318, 410
Spice 104, 132, 151, 157, 416, 419, 422,
425
standing-wave ratio 409
step response 3, 35, 107, 157, 221, 238
SuperCompact 132
SuperStar 132, 151
susceptance 414, 420, 425
Switcap 151
switched capacitor 39, 151, 159, 188,
263, 516, 521, 527, 531
symmetric 11, 18, 174, 191, 260, 364,
385, 558
 structurally 18, 21, 70, 99, 114,
135, 537
 arithmetically 7, 59, 63, 65, 81,
405, 488
synthesis 1, 69, 98, 104, 151, 167, 174,
178, 186, 237, 264, 318, 320, 331, 390,
579
 auto(matic) /computer 6, 75, 127,
427, 435, 455, 544, 577, 579, 594
 manual 8, 20, 54, 89, 273, 276,
285, 361, 455, 588, 593, 596

T

Thiran 250, 587

Index

Thomson 3, 36, 42, 271 – see also
"Bessel"
time domain design 3
Touchstone 104, 298, 314, 416, 423
transducer loss 415
transfer function 1, 41, 82, 87, 106, 147,
150, 154, 158, 167, 188, 191, 203, 236,
239, 243, 250, 263, 318, 337, 348, 361,
387, 410, 470, 477, 516, 527, 587, 596
transmission zeros 1, 15, 18, 21, 27, 40,
46, 52, 61, 67, 82, 87, 95
triplet 12, 109, 118, 564
twin-T 41, 95, 119, 245, 426, 504, 594

U

upper sideband 434

V

voltage loss 103, 413, 416

W

wave-digital 174, 178
windowed 48, 219, 233

Z

z-transform 82, 159, 163, 167, 174, 190,
195, 411, 516
zeros 12, 50, 53, 81, 87, 89, 147, 236,
240, 247, 254, 259, 318, 334, 338, 341,
350, 372, 389, 410, 426, 431, 468, 473,
479, 493, 531, 549
 transmission 1, 15, 19, 24, 39,
47, 52, 61, 64, 82, 95, 98, 109, 149, 168,
174, 191, 201, 280, 325, 361, 366, 377,
385, 396, 400, 426, 434, 454, 488, 498,
517, 541, 555
zig-zag 63, 492