WinIIr

Version 2.0, Joel Bouchat, July 2020..

WinIIr is a rather specific tool for people who have to design digital "**I**nfinite **I**mpulse **R**esponse" filters.

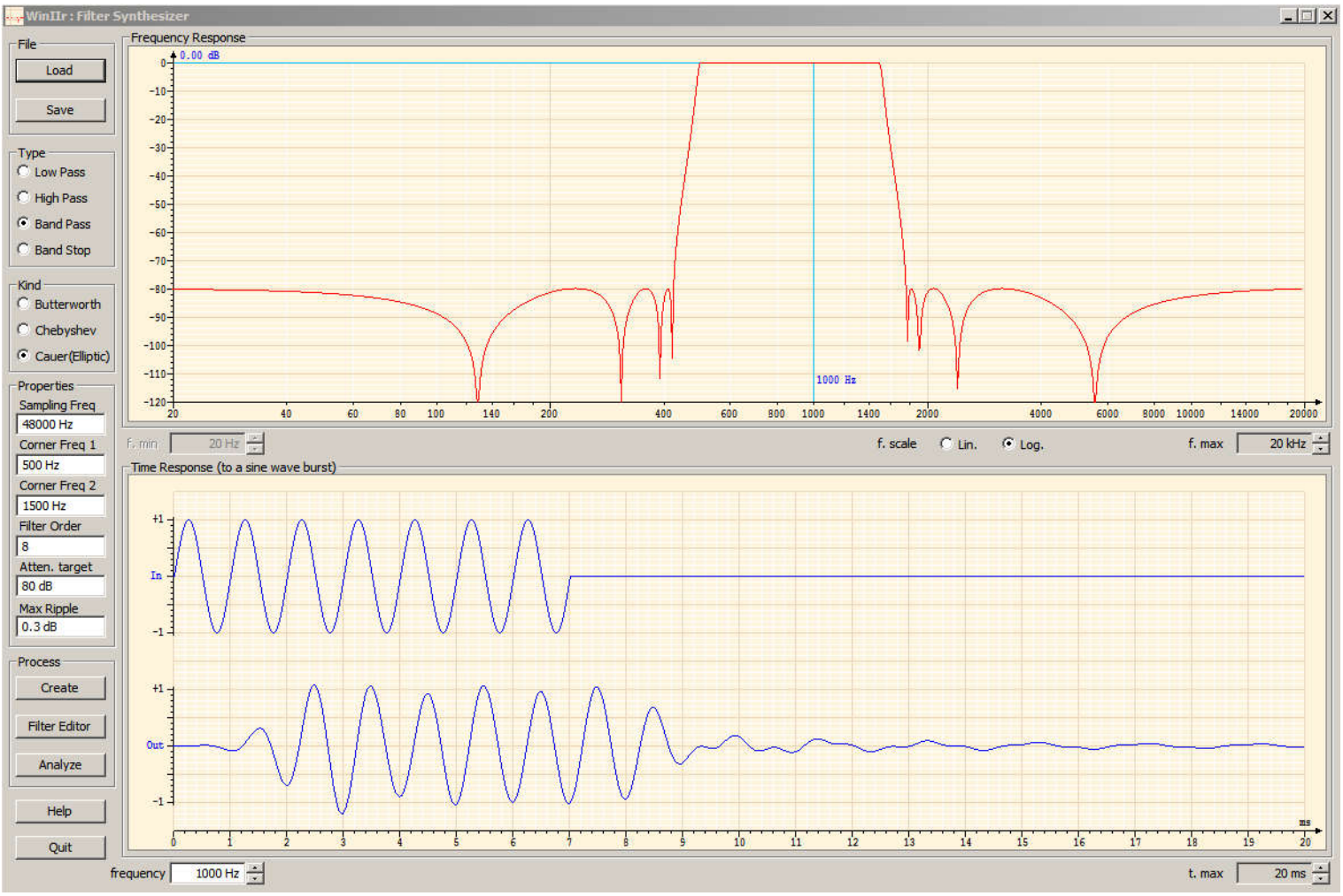
There are several applications available for this purpose, starting with "MATLAB" ©.

Anyway, WinIIr offers certain advantages:

- This software is completely **free** !
- The synthesis is precise and correct. It delivers the same results as "MATLAB. This is not always the case for some packages available on the Web…
- It is ergonomic: optimizing a filter while examining its Frequency and Time Responses is easy.
- The filter is calculated and implemented using the "**Biquad**" architecture. This is the only way for creating high order filters with only a "double float" precision (64 bits). The maximum order is **16**, even for band-pass filters.
- Rather than the classical "Step Response", the Time Response diagram shows the response of the filter to a "**Sine Wave Burst**" at the "**Marker**" frequency. This way, estimating the group delay at a specific frequency is easy. The "**Marker**" frequency can be modified by displacing a cursor on the screen or by entering the frequency in an edition box.
- The filters are stored on the disk in a **human readable text format**, Therefore the files are easy to maintain.
- The application includes **its own text editor**, accessible through the "**Filter Editor**" button. The description is refreshed in real-time during the design.
- The description file also includes **the "c" source code** for implementing the filter.

The Main Dialog Box:

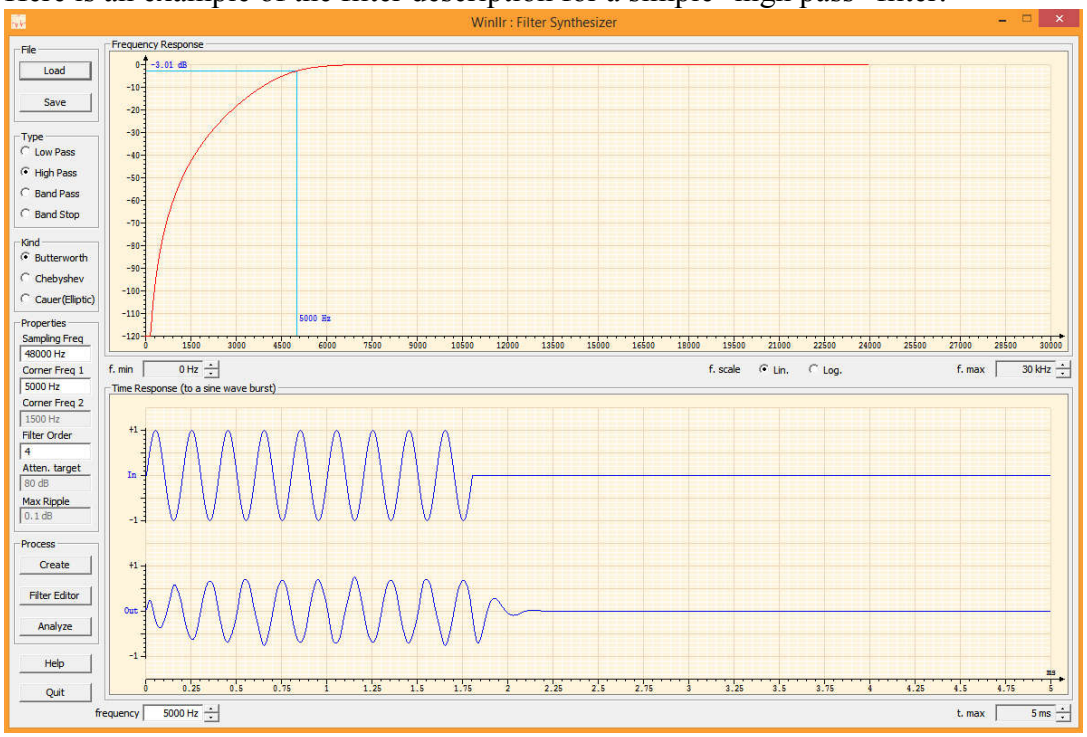
Here is a example of the main application screen for an "elliptic" band pass filter of order 8:



The available features are self-explanatory for people accustomed to filter design.
Four types of filters are available, as usual: low pass, high pass, band pass and band stop, with three possible implementations: Butterworth, Chebyshev and Cauer (Elliptical). For the Chebyshev and Cauer implementations, the maximum ripple must be specified. For the Cauer filter, specifying the attenuation is also required.
The upper diagram displays the **Frequency Response** on a linear or logarithmic frequency scale.
The lower diagram displays the **Time Response** at the selected **Marker** frequency.
The scale of each abscissa is selected by a small spin control.

The Code Editor:

The Editor is used to access the filter description in text format. It is accessible through the "**Filter Editor**" button:
Here is an example of the filter description for a simple "high pass" filter:



Here is the synthesis result accessible through the **Editor Window**:

```
WinIIr Filter Designed  :   2 jul. 2020 at 13:13
-----------------------------------------------

PARAMETERS
----------
Type : High Pass, fc = 5000 @ fs = 48000
Kind : Butterworth, Order 4

SYNTHESIS
---------

Poles
-----
Pole  1: +0.507771412797 +0.149103686251 j
Pole  2: +0.507771412797 -0.149103686251 j
Pole  3: +0.643452720137 +0.456155021735 j
Pole  4: +0.643452720137 -0.456155021735 j

Zero
----
Zero  1: 1

Polynomes
---------
 Denominator                        Numerator
 a0 =  1.0000000000000000e+000,  b0 =  4.1737200862861579e-001
 a1 = -2.3024482658687364e+000,  b1 = -1.6694880345144631e+000
```

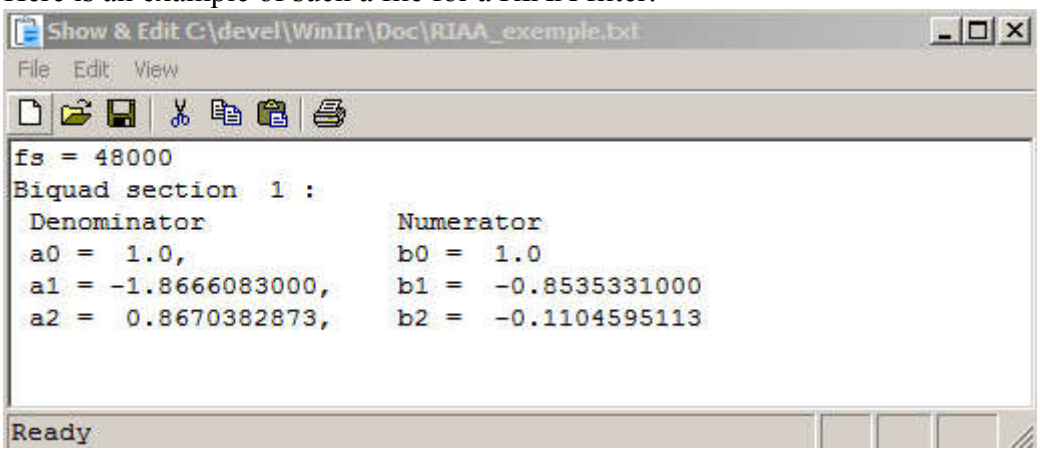
```
 a2 =  2.2090801109022578e+000,  b2 =  2.5042320517716945e+000
 a3 = -9.9219365650392699e-001,  b3 = -1.6694880345144631e+000
 a4 =  1.7423010478293127e-001,  b4 =  4.1737200862861579e-001

Biquads
-------
 Biquad section  1 :
 Denominator                    Numerator
 a0 =  1.0000000000000000e+000,  b0 =  5.7390163562541840e-001
 a1 = -1.0155428255941767e+000,  b1 = -1.1478032712508368e+000
 a2 =  2.8006371690749693e-001,  b2 =  5.7390163562541840e-001

 Biquad section  2 :
 Denominator                    Numerator
 a0 =  1.0000000000000000e+000,  b0 =  7.2725356179509404e-001
 a1 = -1.2869054402745597e+000,  b1 = -1.4545071235901881e+000
 a2 =  6.2210880690581649e-001,  b2 =  7.2725356179509404e-001

-----------------------------------------------------------------

Implementation (C code for the Biquads)
---------------------------------------

//IIR Type : High Pass, fc = 5000 @ fs = 48000
//    Kind : Butterworth, Order 4

#define NBSECTIONS 2

typedef struct
{
    double a1;
    double a2;
    double b0;
    double b1;
    double b2;
} BIQUAD;

double m_qBuf0[NBSECTIONS];
double m_qBuf1[NBSECTIONS];

//Coefficients

static BIQUAD m_biquad[NBSECTIONS] = {
 { -1.0155428255941767e+000, 2.8006371690749693e-001,
    5.7390163562541840e-001,-1.1478032712508368e+000, 5.7390163562541840e-001 },
 { -1.2869054402745597e+000, 6.2210880690581649e-001,
    7.2725356179509404e-001,-1.4545071235901881e+000, 7.2725356179509404e-001 }
};

//Code

// Biquad second order filter section:
// each Biquad section uses the transposed Direct Form II structure shown below:

//   x(n)                              y(n)
//    >-------->b0---->|+|--------------->
//       |              ^             |
//       |              |             |
//       |            z-1            |
//       |              ^             |
//       |              |             |
//       |---->b1---->|+|<-- -a1<----|       (a0 is always equal to 1)
//       |              ^             |
//       |              |             |
//       |            z-1            |
//       |              ^             |
//       |              |             |
//       ------>b2---->|+|<-- -a2<-----
```

```
// Coefficients b0, b1 and b2 multiply the input signal x[n] and are referred to
// as the feedforward coefficients.
// Coefficients a1 and a2 multiply the output signal y[n] and are referred to
// as the feedback coefficients. They must be negated.

//  y[n] = b0 * x[n] + d1;
//    d1 = b1 * x[n] - a1 * y[n] + d2;
//    d2 = b2 * x[n] - a2 * y[n];

// "input" is the input sample
// "n" is the biquad cell index
// returns the output sample
inline double biQuadSection(double input, int n)
{
    double a1 = m_biquad[n].a1;
    double a2 = m_biquad[n].a2;
    double b0 = m_biquad[n].b0;
    double b1 = m_biquad[n].b1;
    double b2 = m_biquad[n].b2;

    // Direct form II implementation
    double a = input -  m_qBuf0[n] * a1 -  m_qBuf1[n] * a2;
    double b = a * b0 + m_qBuf0[n] * b1 +  m_qBuf1[n] * b2;
    m_qBuf1[n] = m_qBuf0[n];
    m_qBuf0[n] = a;

    return b;
}

// Filtering using "Biquad" Second Order Sections
// "input" is the input sample
// "t" = is the "time". Only t = 0 is used to init the filter
// returns the output sample
double biQuadfiltering(double input, int t)
{
    int n;

    if(t == 0)
    {
        for(n = 0; n < NBSECTIONS; n++)
        {
            // init the filter
            m_qBuf0[n] = 0.0;
            m_qBuf1[n] = 0.0;
        }
    }
    // The biquad implementation is recursive
    double out = input;
    for(n = 0; n < NBSECTIONS; n++)
    {
        out = biQuadSection(out, n);
    }
    return out;
}
```

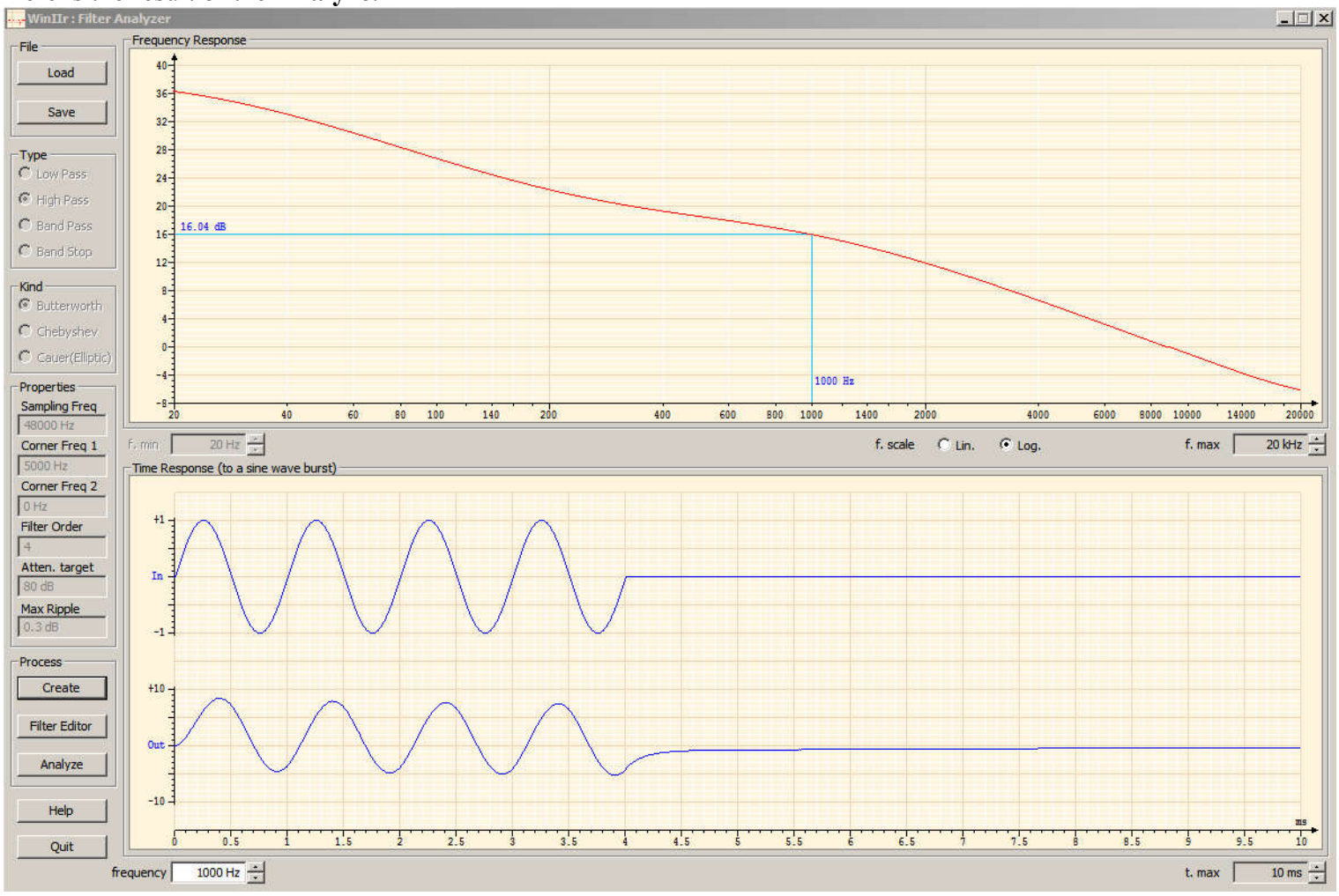
Analyzing a given filter:

WinIIr can also be used for **analyzing** a specific filter.
For this purpose, you must "Load" a text file containing the sampling frequency and the biquad parameters.

Here is an example of such a file for a RIAA filter:



Here is the result of the **Analyze**:



The process is interactive, you can modify the filter parameters inside the **Filter Editor** Window and **Analyze** it again to see the consequences of the modifications.