IIR Design Program Readme

User Instructions with Examples

1 Thumbnail

The program IIR_Design.exe is a GUI-based program that provides

- An automated way to obtain the design of digital filters with independently determined numbers of poles and zeros,
- Decimation by a FIR filter followed by equalization of bandpass shape with a small IIR filter implemented at the decimated sample rate are produced by this design program, and
- Outputs in the form of a report-ready half-page design report, and graphics of a pole-zero plot and a frequency response plot, with a CSV file with included full-precision design and implementation parameters.
- Input passband peak-to-peak ripple in dB is an input, as is passband frequency; stopband attenuation is determined as part of the design process and provided as an output.

The program provides for decimation with a combination of low latency and good pulse shape preservation, a mid-ground between FIR decimation filters and digital elliptic filters. This is accomplished by a FIR decimation filter that does not flatten the passband, and an IIR filter implemented post-decimation that provides a flat passband shape with a small number of low-Q poles.

The constraints implicit in defining the filter this way does affect its design; these considerations are detailed below.

Advantages and disadvantages of general FIR/IIR filters relative to FIR and elliptic filters are summarized below as Table 1.

Table 1. Summary of Filter Advantages and Disadvantages by Type

Filter Type	Advantages	Disadvantages
General IIR	• Low latency • Low delay • Low waveshape distortion	Not linear phase
FIR	Linear phase – zero waveshape distortion	• Long delay • Long latency
Elliptic	Shortest possible transition between passband and stopband	• Rapid phase variations near passband edge • Waveshape distortion for wideband signals

Table of Contents

List of Tables

List of Figures

1.1 License

This program is released for use under the two-clause “simplified” BSD license:

The license is provided on the pop-up that provides a GUI for program inputs. The source code is not released at this time, although I do plan release the source code eventually. The source code is in Fortran 2008; use of Absoft Pro Fortran 2016 features are used for the GUI and graphics.

2 References

The algorithm roughly follows

H. Martinez and T. Parks, *A new class of recursive digital filters for decimation*, ICASSP '78 (Volume 3), pp 508-511; DOI: http://dx.doi.org/10.1109/ICASSP.1978.1170440

A later journal paper with likely more detailed documentation:

H. Martinez and T. Parks, *A class of infinite-duration impulse response digital filters for sampling rate reduction*, IEEE Transactions on Acoustics, Speech, and Signal Processing (Volume:27 , Issue: 2), pp 154-162; DOI: http://dx.doi.org/10.1109/TASSP.1979.1163207

I attended the ICASSP presentation. I did not refer to the journal paper, having my own program in place well before it came out. The software and algorithms in this program are entirely my own and did not originate in code from any other source.

Applied mathematics, numerical analysis, and other classical reference data and information can be found in

Handbook of Mathematical Functions, with Formulas, Graphs, and Mathematical Tables; Milton Abramowitz and Irene A. Stegun, Editors, National Bureau of Standards Applied Mathematics Series 55, Tenth Printing, December 1972, with corrections, Library of Congress Catalog Card Number: 64-60036.

There seems to be no ISBN or DOI for this book, although facsimiles are available from Dover and other publishers. It may or may not be still available from the U.S. Government Printing Office. The 1964 edition and an errata sheet dated 1966 may be available for viewing or checkout from Government libraries. Be sure and get the 1972 edition or later (or at least the fifth printing of August 1966); there are hundreds of corrections in place in later printings that are not present in the original 1964 edition.

A modern, maintained replacement for Abramowitz & Stegun, the NIST Digital Library of Mathematical Functions (DLMF), is available on the Internet:

http://dlmf.nist.gov/

For extensive use, this online resource is best used with hard copy and a CD-ROM of the latest online material, available from NIST through Cambridge University Press here:

http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521192255

I have found that the DLMF has modernized and more comprehensive references for the mathematical and computational aspects of all topics, as well as additional topics. Tables are not included in the DLMF since computation of tables is done by desktop computers, but I find the tables in Abramowitz & Stegun invaluable in validating software that, as your own source code, provides in-application support in multiple precision, the complex domain, and other areas not available by any other means. Thus MILF and Abramowitz & Stegun are supplementary works, and if you need one, you likely are best served if you have both available to you.

3 Operation

3.1 Program Inputs

3.2 The GUI

Starting the program produces the inputs window shown in Figure 1 below.

3.3 Fast Text File User Interface

One of the output files is “Last_IIRDES_Inputs.txt” where all of the filter design inputs are listed, and document the user inputs for the last run. The GUI module looks for a file with the identical format named “IIRDES_Inputs.txt” and, if the file is present and there are no I/O errors in reading in the user-defined input parameters, that file is used and the program proceeds without pausing for input. The quantization input is taken at 1 when this happens.

You can implement a quick interface for advanced users by editing the name of the file “Last_IIRDES_Inputs.txt” to become “IIRDES_Inputs.txt” and loading it in Notepad or other simple text editor. You can edit parameters in your text editor and save the file to change the inputs and run the program by clicking on IIR_Design.exe. If you make a mistake, or the inputs fail one of the sanity checks, the program will come back and display the GUI window of Figure 1. When that happens, delete IIRDES_Inputs.txt and make a new one from the last Last_IIRDES_Inputs.txt file that is close to your desired design.

In addition to a fast user interface for experienced users, this capability can be used “headless.” Another program can use IIR_Design.exe to design programs without an operator. First, the IIRDES_Inputs.txt file is generated by another program, then the IIR_Design.exe program is invoked, then the CSV file is used to provide the design to the external program.

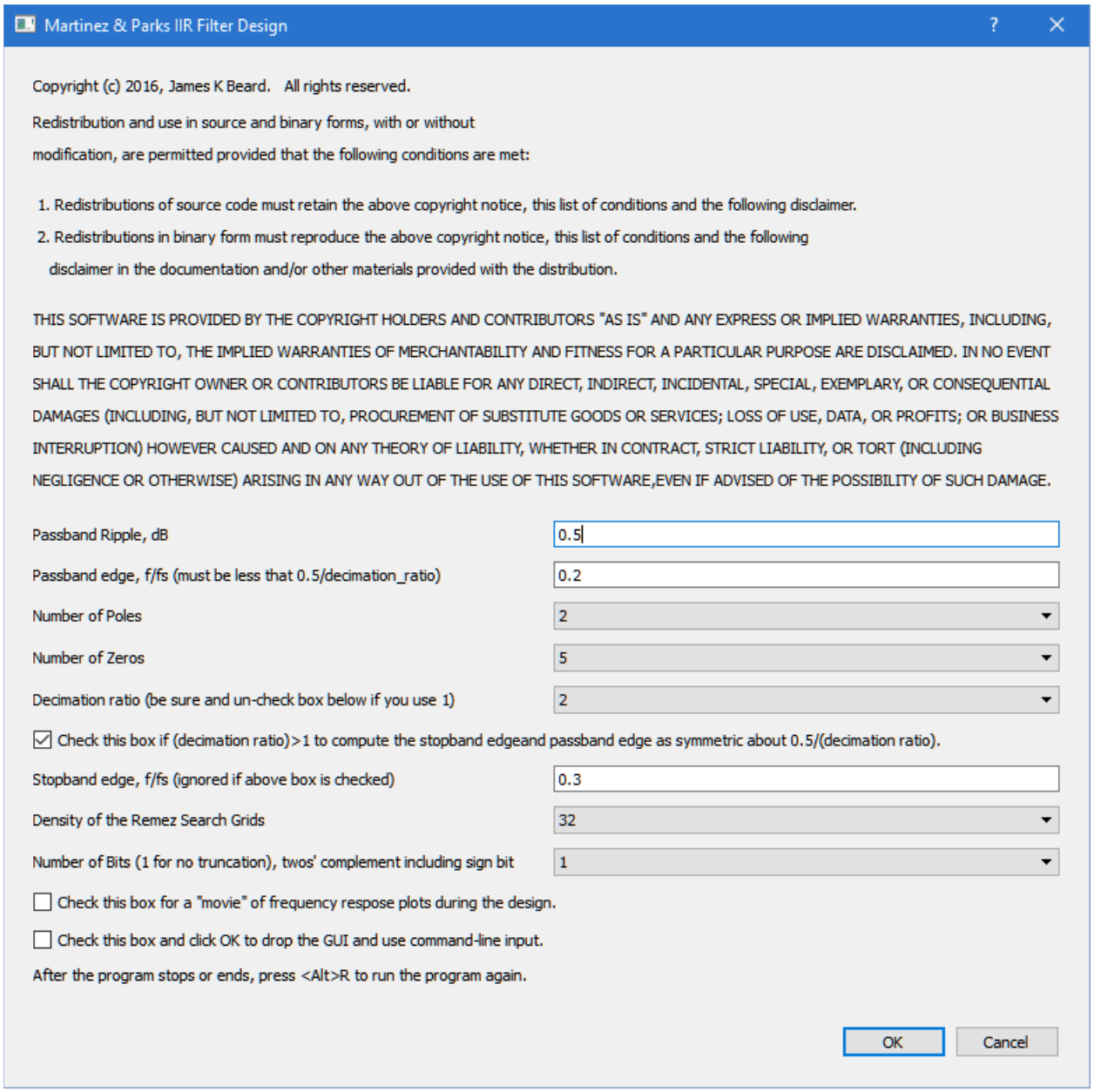

Figure 1. Program Input Window

Most users will provide the passband ripple in decibels, the passband edge, the number of poles and zeros, and the decimation ratio, and click OK. If you want to see the effect of quantization of arithmetic on the frequency response, the user can provide the number of bits as an option in the last combo box. Leaving the default of "1" does not truncate the filter weights.

3.4 Command Line Interface

There is a check box at the bottom of the input window of Figure 1. Check that box and you can use a command-line interface from the main text window in the text window that is titled IIR_Design.exe.

The individual inputs are treated in detail below.

3.5 Outputs

Clicking OK with the default inputs produces three graphics windows, two text windows, and hard copies (or data files) of a filter design. The file outputs are all prefaced with a date-and-time tag of the form yyyymmdd.hhmmss_ followed by the file names IIRDES_Design.txt, IIR_Filter_Frequency_Response_h.png, and IIR_Filter_Frequency_Response_PolesAndZeros.png.

3.5.1 Filter Design Output as a Text File

The text file for the default inputs is shown in Table 2 below.

Table 2. Filter Design for Simple Example (Default Inputs)

```
************************************************************

 IIR filter design

 Iterative Remez exchange algorithm

 Decimation ratio   2
 Poles   2
 Zeros   5
 lgrid  32

 Passband, 0.0     to 0.200000, ripple       0.500 dB
 Stopband, 0.300000 to 0.5    , attenuation  36.251 dB

 Numerator weights

 h( 3) = h( 4) =   5.569420E-01
 h( 2) = h( 5) =   3.403308E-01
 h( 1) = h( 6) =   1.127754E-01

 Denominator polynomial  (in 1/Z**(2))
 1/Z**( 0) =   1.000000E+00
 1/Z**( 2) =   8.804530E-01
 1/Z**( 4) =   1.994925E-01

 Poles
     Real       Imag        Freq     Magnitude
-0.440227  0.075452  0.472984  0.446646

 Frequencies of zeros
  0.306675  0.364778  0.500000

 Frequencies of ripple peaks
  0.000000  0.164053  0.200000  0.300000  0.327075  0.423482

************************************************************
```


3.5.2 Filter Frequency Response Plot

The frequency response graphic is produced from internal program data and also computed using the filter weights given in the program. The plot from filter weights is shown in Figure 2 below.

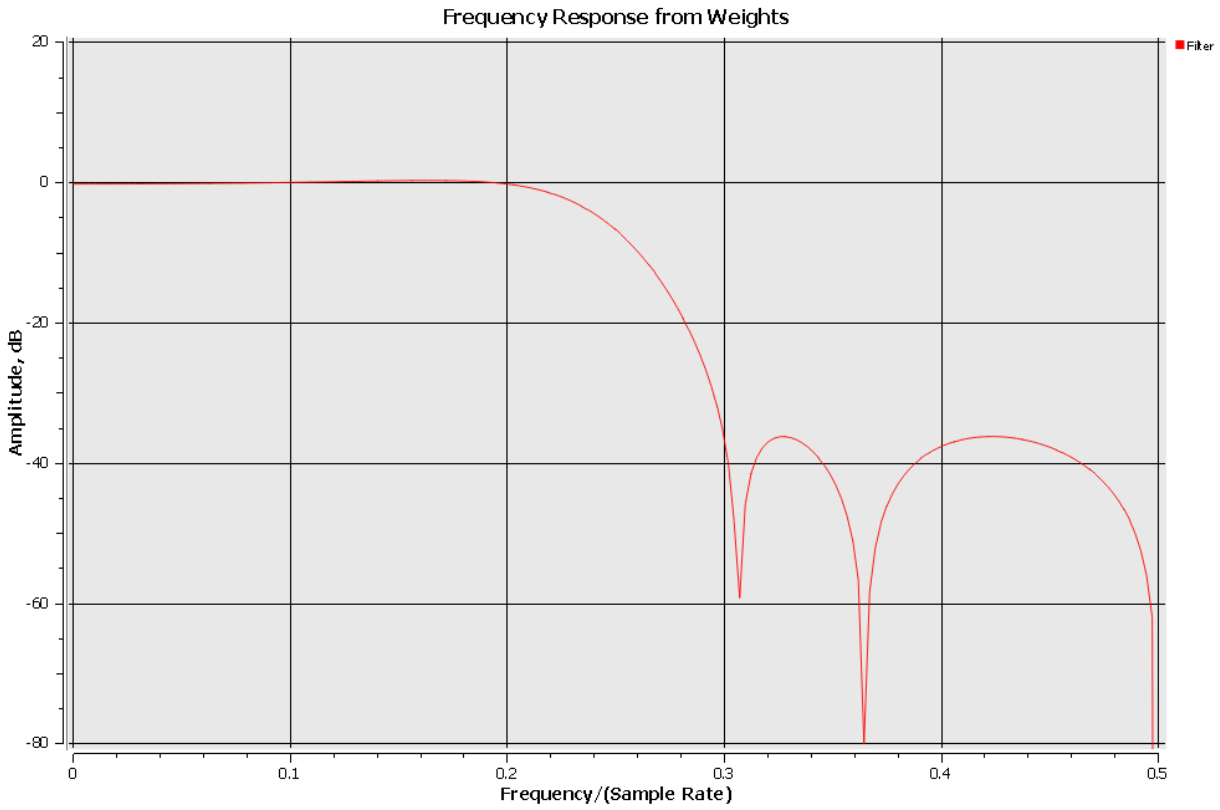

Figure 2. Frequency Response from Filter Weights for Simple Example

3.5.3 Pole-Zero Diagram Graphic

The pole-zero graphic for the example using the input parameters of Figure 1 is shown in Figure 3 below.

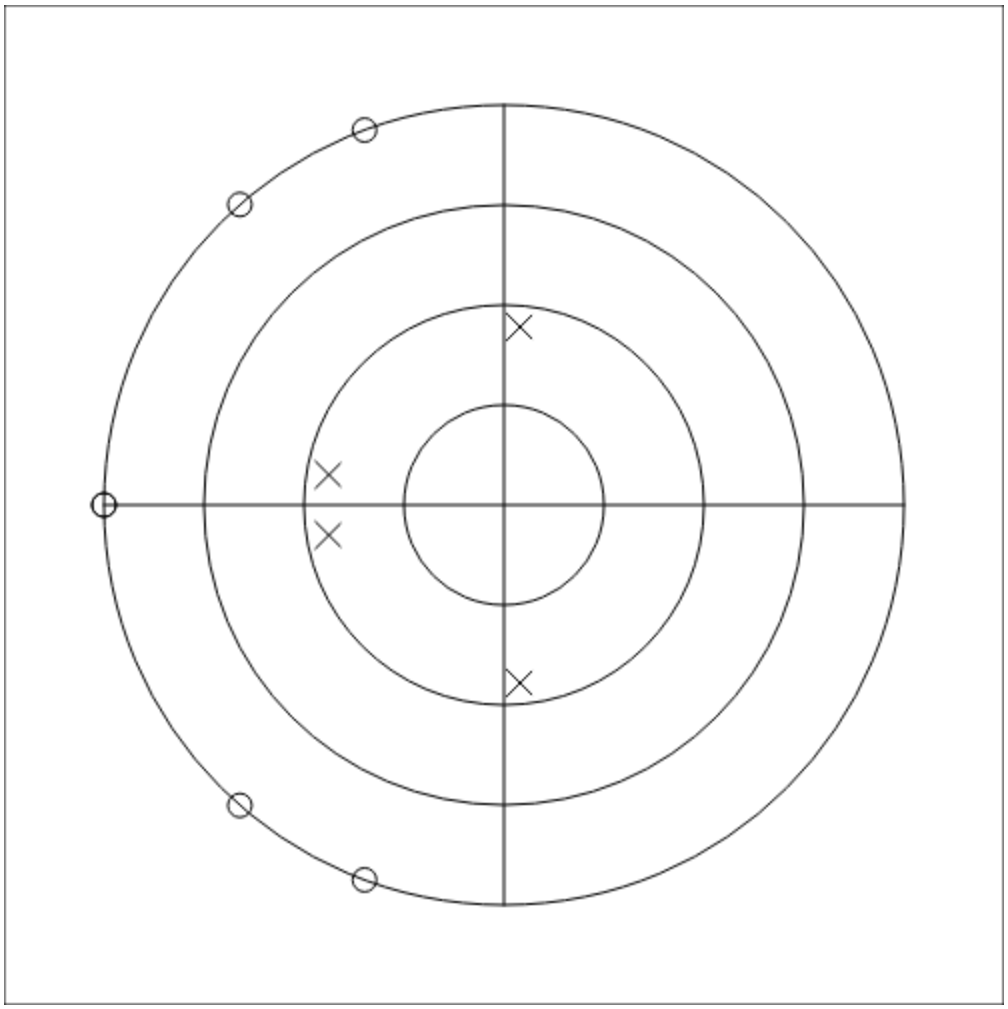

Figure 3. Z-Plane Pole-Zero Graphic for Simple Example

Note that four poles are shown in Figure 3, but the inputs specified in Figure 1 and the outputs shown in Table 2 and Figure 2 show two poles, or one pole-pair. This is because the decimation ratio is 2, and the filter is implemented as a FIR filter followed by decimation, then an IIR filter at the decimated rate, so, referred to the input, as the input frequency cycles from zero to half the sample rate (which is the Nyquist frequency for that sample rate, usually referred to as simply “Nyquist”), the output cycles at twice the rate of the input frequency variation and cycles from zero to Nyquist and continues, aliased, back to zero frequency past the pole in the lower half-plane. The effect on the overall frequency response is that the effects of the poles is in evidence at half the frequency of the poles at the output frequency, and the effect occurs twice as the input frequency is varied varies from zero to Nyquist.

Note that the pole-pair is very low Q, indicating that the effect needed to equalize the input passband to within half a decibel over the specified frequency range of up to 20% of the sample rate is small enough so that the effect on relative phase, phase delay, and wave shape is small, making this filter competitive with FIR and elliptical filters. This filter has very high performance, far less latency than a FIR filter of equivalent performance, and far better phase performance than an elliptic filter of similar performance.

3.5.4 Other Outputs

3.5.4.1 Windows in Main Application Window

After execution, the main application window has two text windows and three graphics windows. One text window, with the label "IIR_Design.exe." is the main application window and contains the application run log, and ends with the filter design summary as given in the hard copy text file shown in Table 2 on page 7. The run log is useful when trying new designs as discussed below in Section 5.3.1.

A second text window provides printer-plots of the frequency response as computed from the filter weights, as well as the frequency responses from only zeros, and from only poles. This file is not saved, although it can be saved from the File menu in the main application window. This file is useful because it provides numerical output for each point, which is useful in verifying passband ripple and stopband attenuation.

The filter design in the main text window is written as a text file, and the frequency response calculated from filter weights and the pole-zero graphics are stored as portable network graphics (PNG) files. Another graphic, the filter response computed from the cosine polynomials (see Section zzzz), is not stored. Its value is in difficult cases, where comparison between the two frequency response plots can reveal numerical difficulties (see Section 5.3).

3.5.4.2 The Last_IRRDES_Inputs.txt File

An output file, Last_IIRDES_Inputs.txt, provides validation that your inputs were what you intended. Also, if you report a problem with the program, please include this file so that I can reproduce your problem. This file has no date/time file name tag and is overwritten each time the program runs.

3.5.4.3 The Comma-Separated Values (CSV) File

A summarized version of the values in the filter design output text file is written as a CSV file. The principal difference is that all values are output in full double precision. This output simplifies implementation. In particular, CSV files are very easily read by other programs in languages such as C/C++, Python, Fortran, Matlab, Maple, and Mathematica.

4 Other Examples

4.1 Elliptic Filter

An elliptic filter can be designed by specifying a decimation ratio of 1 and the same number of poles and zeros. We demonstrate one here by specifying

- Passband peak-to-peak ripple, 0.5 dB
- Passband 0 to 0.3 times the sample rate
- Stopband beginning at 0.35 times the sample rate
- Six poles and six zeros, decimation ratio 1.

A design is achieved with a minimum stopband attenuation of 58.5 dB as shown in Table 3, Figure 4 and Figure 5 below.

Table 3. Filter Design Output for Six-Pole Elliptic Filter

```
**********************************************************

IIR filter design

Iterative Remez exchange algorithm

Decimation ratio   1
Poles   6
Zeros   6
lgrid  32

Passband, 0.0     to 0.200000, ripple       0.500 dB
Stopband, 0.250000 to 0.5    , attenuation  58.518 dB

Numerator weights

h( 4) = h( 4) =  -8.056774E-02
h( 3) = h( 5) =  -7.113430E-02
h( 2) = h( 6) =  -3.992177E-02
h( 1) = h( 7) =  -1.800310E-02

Denominator polynomial  (in 1/Z**(1))
1/Z**( 0) =   1.000000E+00
1/Z**( 1) =  -2.440268E+00
1/Z**( 2) =   3.884216E+00
1/Z**( 3) =  -3.822734E+00
1/Z**( 4) =   2.585953E+00
1/Z**( 5) =  -1.090334E+00
1/Z**( 6) =   2.379263E-01

Poles
   Real     Imag     Freq     Magnitude
 0.547620  0.302095  0.080232  0.625419
 0.385922  0.722523  0.171922  0.819130
 0.286592  0.907975  0.201340  0.952132

Frequencies of zeros
 0.253774  0.291158  0.405632

Frequencies of ripple peaks
 0.000000  0.069399  0.124688  0.162042  0.184681  0.196373
 0.200000  0.250000  0.266609  0.335254  0.500000
**********************************************************
```


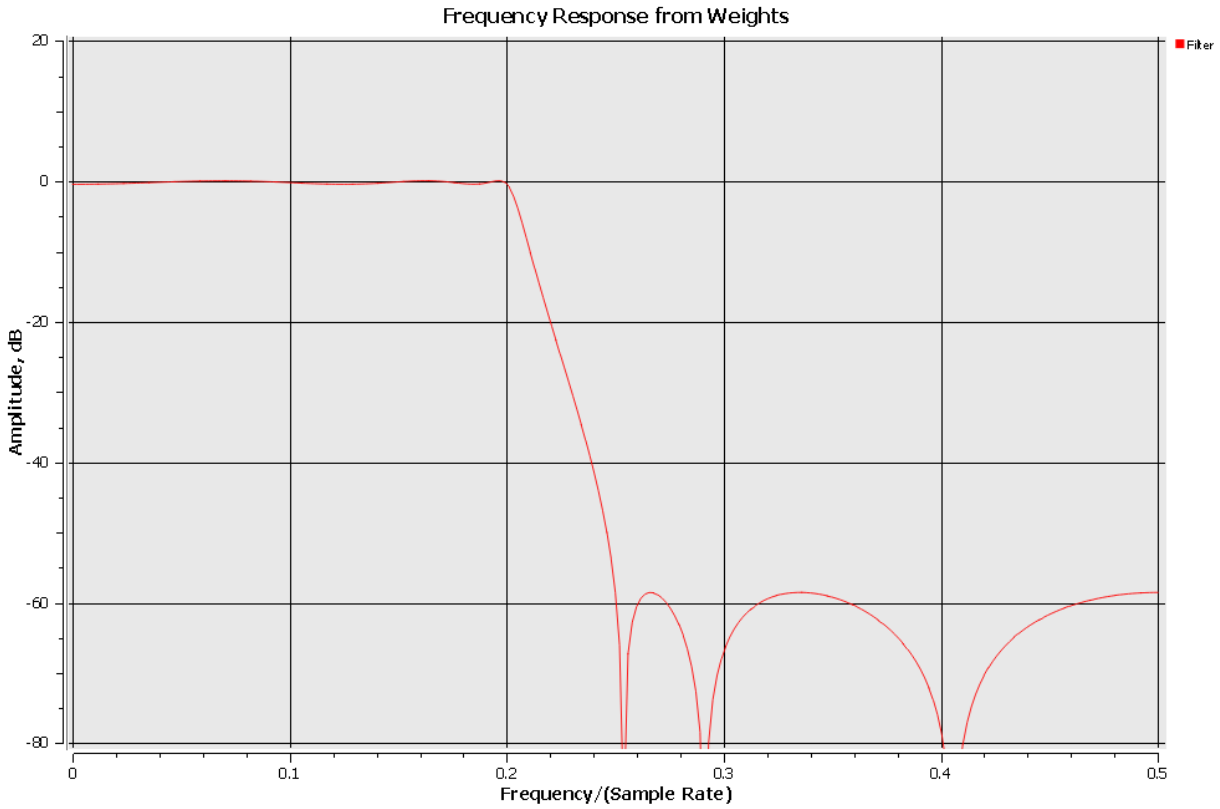

Figure 4. Frequency Response of Six-Pole Elliptic Filter

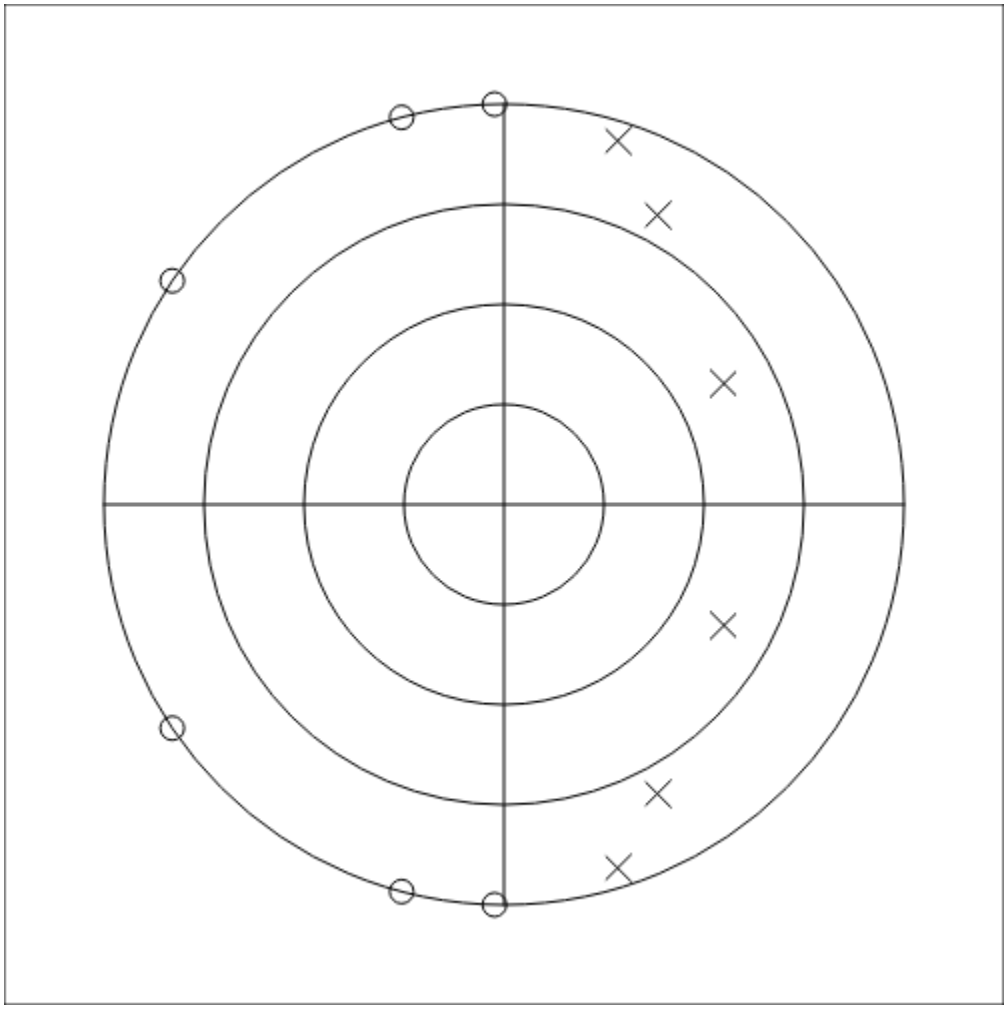

Figure 5. Z-Plane Pole-Zero Graphic for Six-Pole Elliptic Filter

Although elliptic filters can be designed using this program, the closed-form direct-to-results using elliptic functions is a more direct way to obtain elliptic filter designs because the algorithm inherently prevents input of requirements for non-viable designs; the order, passband ripple and bandwidth are specified and either the stopband edge or the stopband attenuation are specified and the free parameter is found from the elliptic function model of the filter. In this program, difficult approximations from polynomials in cosines involving huge alternating values are found iteratively by Remez exchange for both the poles (in the passband) and the zeros (in the stopband). See Section **Error! Reference source not found.** for a discussion of problems that occur in difficult designs.

4.2 High Performance Decimation Filter

A decimation filter with these requirements:

- Passband peak-to-peak ripple, 0.1 dB
- Passband 0 to 0.12 times the sample rate
- Six poles
- 25 zeros

- Decimation ratio 4

produces the outputs shown in Table 4, Figure 6 and Figure 7 below. Note that the poles nearest the passband show some resemblance to the pattern of the poles in the order 6 elliptic filter, but the decimation factor of four has the effect of scattering the poles throughout the interior of the unit circle. in particular, the higher-Q poles near the edge of the passband are replicated in the stopband and force placement of zeros near them to achieve uniform stopband attenuation.

Table 4. Filter Design for High Performance Decimation Filter Output

```
************************************************************
IIR filter design
Iterative Remez exchange algorithm

Decimation ratio   4
Poles   6
Zeros  25
lgrid  32

Passband, 0.0     to 0.120000, ripple       0.100 dB
Stopband, 0.130000 to 0.5    , attenuation  53.415 dB

Numerator weights

h(13) = h(14) =   9.654521E-01
h(12) = h(15) =   9.183356E-01
h(11) = h(16) =   8.345940E-01
h(10) = h(17) =   7.194558E-01
h( 9) = h(18) =   5.883853E-01
h( 8) = h(19) =   4.522331E-01
h( 7) = h(20) =   3.328854E-01
h( 6) = h(21) =   2.260664E-01
h( 5) = h(22) =   1.419618E-01
h( 4) = h(23) =   7.914429E-02
h( 3) = h(24) =   4.341311E-02
h( 2) = h(25) =   1.920846E-02
h( 1) = h(26) =   6.133561E-03

Denominator polynomial  (in 1/Z**(4))
1/Z**( 0) =   1.000000E+00
1/Z**( 4) =   3.135232E+00
1/Z**( 8) =   3.763480E+00
1/Z**(12) =   2.145588E+00
1/Z**(16) =   5.878888E-01
1/Z**(20) =   7.641267E-02
1/Z**(24) =   7.630879E-03

Poles
   Real     Imag     Freq     Magnitude
-0.056332  0.145018  0.308968  0.155574
-0.599646  0.135660  0.464590  0.614800
-0.911638  0.055118  0.490389  0.913303

Frequencies of zeros
 0.130331  0.133216  0.140584  0.156147  0.187369  0.241786
 0.309279  0.354643  0.370984  0.377288  0.389047  0.426883
 0.500000

Frequencies of ripple peaks
 0.000000  0.048770  0.083823  0.103653  0.113969  0.118681
 0.120000  0.130000  0.131431  0.136117  0.146908  0.169220
 0.211364  0.276269  0.336355  0.365424  0.374277  0.381057
 0.403358  0.459753
************************************************************
```


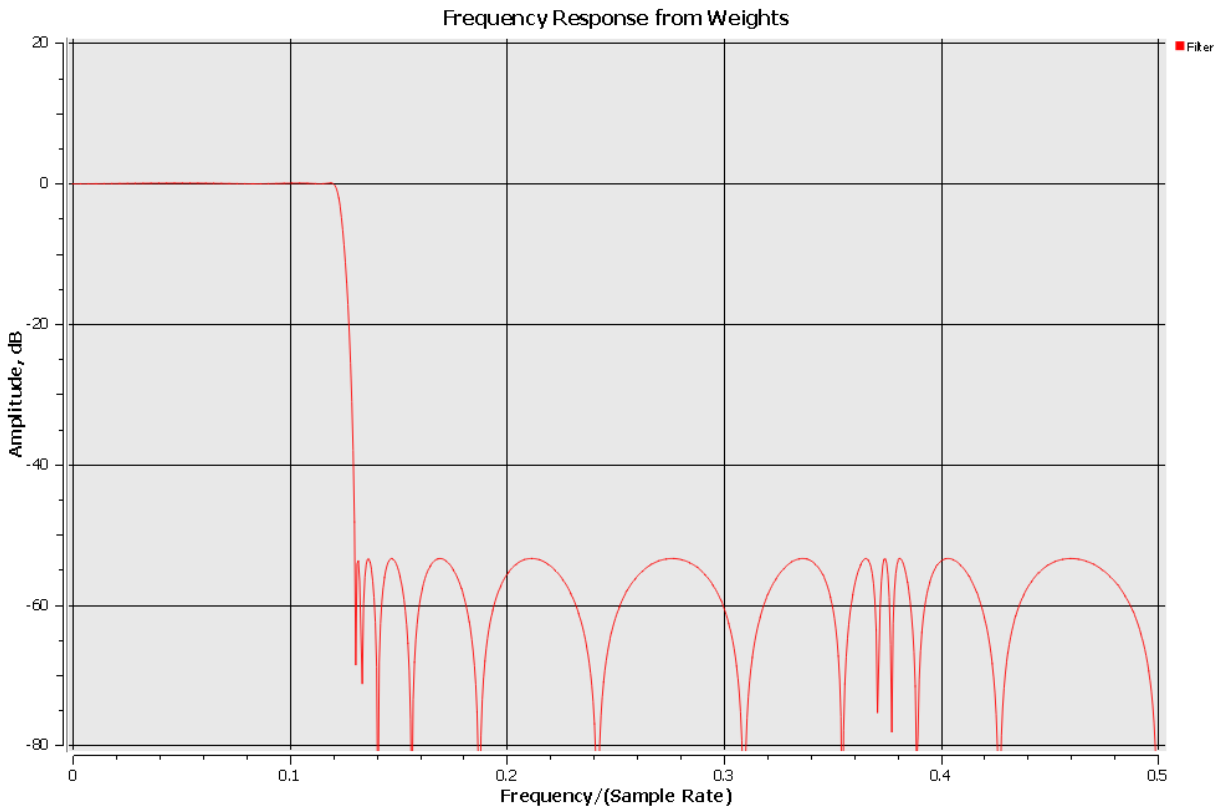

Figure 6. Frequency Response for High-Performance Decimation Filter

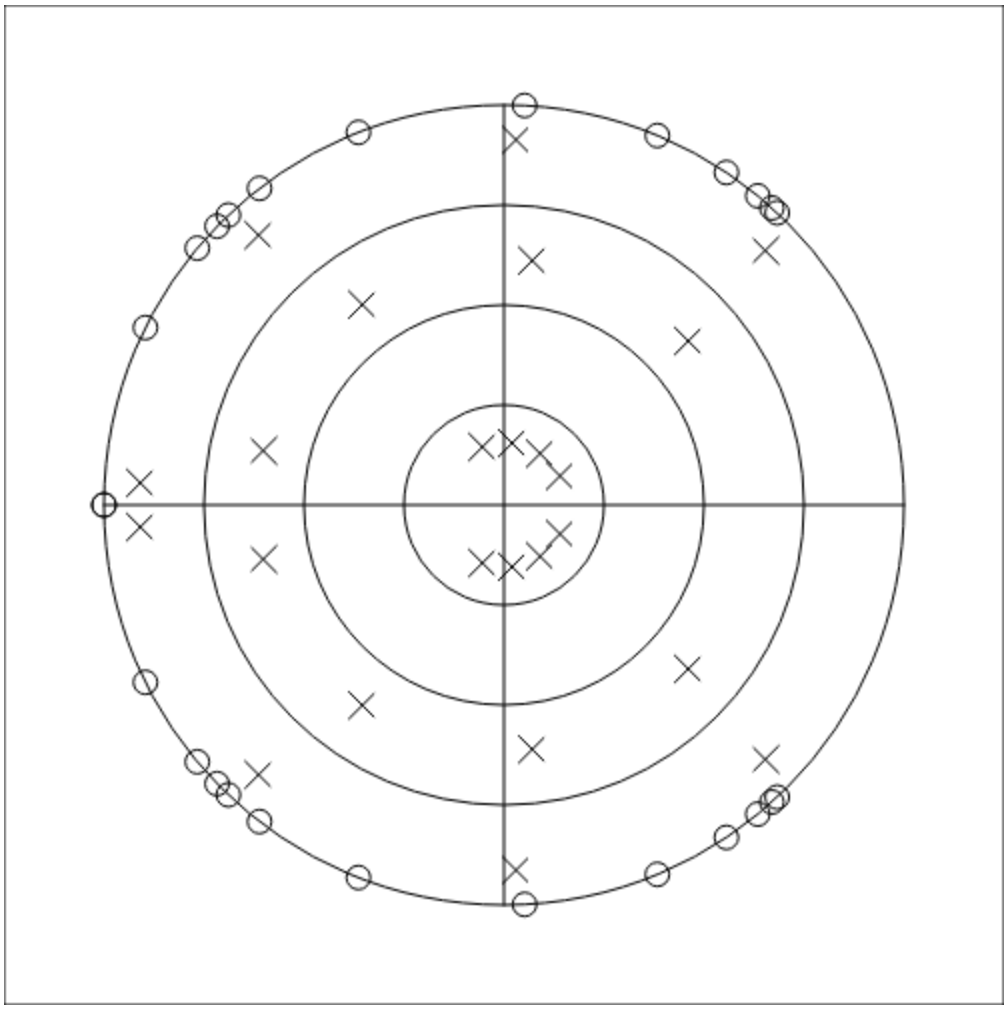

Figure 7. Z-Plane Pole-Zero Graphic for a High-Performance Decimation Filter

5 Technical Details

5.1 General Description of Algorithm

This program models the filter as the ratio of polynomials of cosines of frequency. The numerator polynomial determine the zeros, and the square root of the denominator polynomial determines the passband shape. The passband polynomial is fit to quantities that are large in magnitude because the values of the numerator polynomials, which are fit to values of alternating sign that approximate the ripple peaks in the stopband. Thus the stopband polynomial has some resemblance to a Chebychev polynomial of the first kind, which increases very rapidly outside its region of approximation, in particular the passband. The passband polynomial forces overall filter response by fitting the necessary shape of a polynomial that divides this hugely expanding curve, and itself exhibits value explosion outside its area of approximation, in particular the stopband. A successful filter design forces specified behavior on the passband polynomial, and then given that, on the stopband polynomial, and recurs until a solution is found. The result is that the algorithm is subject to a huge magnitude increase at each

iteration. To avoid a trend in the magnitude of the polynomials, they are re-normalized before finding a solution for either the numerator or denominator polynomial.

5.2 The Coordinate Systems

The user is asked to present frequencies as a fraction of the sampling ratio, which results in frequencies represented by unitless quantities between zero and 0.5, inclusive. Digital filters will be periodic in this quantity with period 1. Functions even about zero frequency that are periodic with period 1 can be expressed as Fourier series in $\cos\left(2\pi \cdot f / f_{SAMPLING}\right)$ which, through trigonometric identities, particularly

$$T_N\left(\cos(\theta)\right) = \cos\left(N \cdot \theta\right)$$

are Fourier series in $\cos\left(2\pi \cdot f / f_{SAMPLING}\right)$. We call this the cosine domain.

We express our filter as a rational z domain transfer function, and we design the numerator and the square of the denominator polynomial in the cosine domain. When the design is done, we find the weights of the numerator and squared denominator with a discrete Fourier transform, which, because the functions involved are even in both the time and frequency domains, is a discrete Cosine transform, and both are linear phase FIR filters. The denominator has complex root-pairs, half of which are in the unit circle and half of which are outside the unit circle. The stable roots are the roots inside the unit circle, and these are found by first radius-searching using Schur's criterion on scaled polynomials to find the radius of each root, then searching around the unit circle to find its imaginary part, and completing the root-finding with the secant method. The roots are then used all together to generate the denominator polynomial, which provides the user with the option to implement the filter by stages of root pairs or with a single denominator multi-stage IIR filter.

The denominator operates at a decimated frequency rate, so the Chebychev polynomial relationship is used to find the cosine space for the denominator from that of the numerator.

Thus the coordinate systems we deal with in this IIR filter design include:

- Normalized frequency, a real, dimensionless number from zero to 0.5,
- Cosine space, where the independent variable is $\cos\left(2\pi \cdot f / f_{SAMPLING}\right)$,
- Denominator polynomial space, where the independent variable is $\cos\left(2\pi \cdot K \cdot f / f_{SAMPLING}\right)$ where K is the decimation ratio, and
- The "time domain" where the filter weights, or z transform coefficients, are produced as output.

5.3 Possible Difficulties

5.3.1 Non-Viable Design Requirements

It is quite easy to specify a filter that cannot exist. For example, if you require a passband peak-to-peak ripple of 0.1 dB over a passband to 0.2 times the center frequency, but a stopband that begins at 0.21 times the center frequency with only one pole-pair and several zeros, this filter cannot exist because no

rational function of reasonable order can pass through the data points as specified by the inputs. You will need more poles to control the passband to achieve the required small transition to stopband while maintaining a flat passband. If the user attempts such a design, the program will produce a pop-up as Figure 8 below.

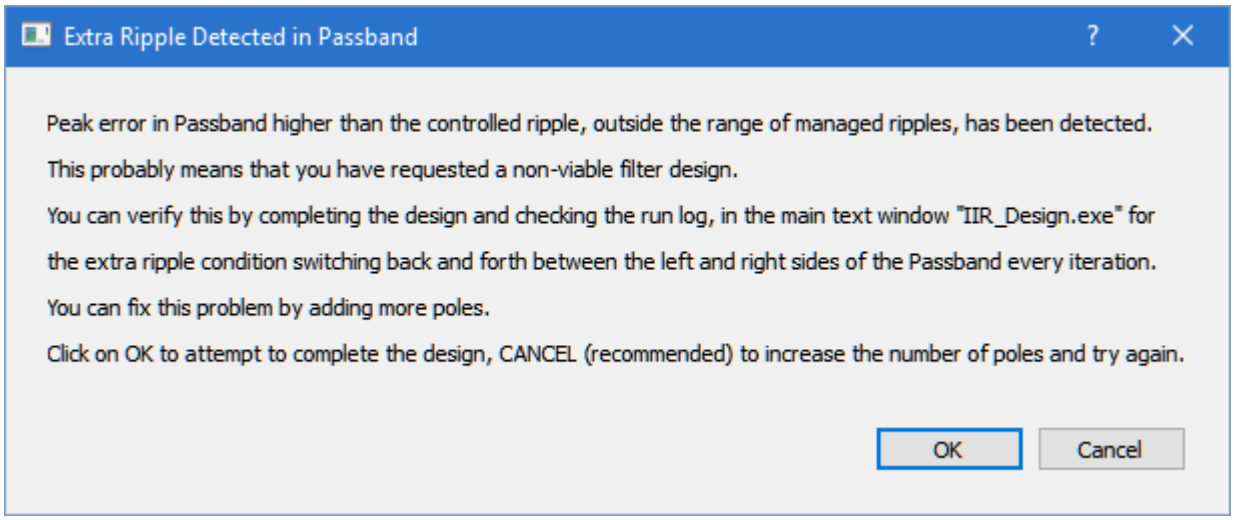

Figure 8. Pop-Up when Too Few Poles are Used

When this pop-up appears, the simplest thing to do is to click Cancel and add a pole-pair to the design and try again.

5.3.2 Tendency to Magnitude Explosion and Overflow

The huge values that each polynomial approximates to force specified behavior in, successively, the numerator and then the denominator, will cause an increase at each iteration that can quickly result in overflow, even with double precision with maximum valid representation of magnitudes on the order of 10^{308}. This is countered in the program by continually re-scaling the magnitudes of the numerator and denominator polynomials.

5.3.3 Possible Magnitude Collapse

Since the algorithm fits the numerator and denominator polynomials to the behavior to the other polynomial, and the Remez exchange algorithm finds the true ripple peaks by placing estimated ripple peaks at a slightly low magnitude and tracks differences from the assigned values to arrive at an equiripple solution, the algorithm tends to request a slightly low value for the stopband attenuation at each iteration. Since equiripple stopband is achieved for a numerator polynomial that is all zeros, this is, numerically, a possible solution; this provides a low magnitude curve for the passband function to force to equiripple behavior, decreasing the magnitude of the denominator polynomial. The result is, that in cases where the specifications are for a non-viable design or for a very difficult viable design, the magnitudes of the numerator and denominator polynomials may “crash” or tend to very low values. This is called magnitude collapse.

5.3.4 Numerical Failure

Due to the huge numerical values in the numerator polynomial seen when cosine space values for the passband are used, overflow sometimes occurs. The denominator fitting portion of the program looks at values that include values of the denominator polynomial divided by the square of values of the numerator polynomial; when that value is in overflow, values divided by it are taken as zero. This expediency prevents the computation of the magnitude of the filter frequency response in cosine space as linear functions of the values of the numerator and denominator polynomials and results in errors in the frequency response curve. If this happens during the course of the filter design, the error will be left behind after the next normalization and filter design iteration. If it is present when the filter program has completed, you will usually get a pop-up that poor convergence has occurred, and the frequency response curves will not agree when computed from cosine space and from the computed filter weights. In general, neither frequency response curve will meet user requirements. When this happens, you can move toward a viable design by changing your inputs. Methods for doing this include

- If the stopband attenuation is greater than 80 dB, try decreasing the transition between the passband and stopband, or reducing the number of zeros.
- If you get a warning pop-up as in Figure 8, click Cancel to start over and try again with one more pole-pair.
- Start over with just a few zeros and increase the number of zeros to get the desired stopband attenuation.

6 Possible Future Enhancements

The source code may or may not be released at a future date. The simplified BSD license is contemplated for such a release.

A multiple precision version of the program may be released. This will eliminate overflow and underflow as practical problems in the filter design.

A true Windows user interface may be used in a future release. This will mean that the program is not source-code-compatible between Windows, Mac OS, and Linux, an advantage for the current program because Absoft Pro Fortran is source-code-compatible between these three platforms.

The current program is compiled as a 32-bit Windows application. A 64-bit version is not required and will result in a slightly larger and slightly slower binary, and thus is not planned at this time.

Rational z transforms can be expanded into continued fractions, which have a corresponding IIR filter data flow and architecture. Because the sensitivity of the filter frequency response to continued fraction coefficients is usually lower than the sensitivity to numerator or denominator polynomial coefficients, more accurate frequency response for a given word length in hardware may be achieved with this modified IIR data flow and architecture. This capability may be released for this program in the future.

7 Reporting Errors

If you have a repeatable error, and you have tried all the procedures listed in Section 5.3.4 above, please send me an e-mail with a reasonably complete description of your problem, and attach the Last_IIRDES_Inputs.txt file that generates the error. Although I cannot commit to providing services on demand without compensation, I appreciate bug reports and will try to respond to user problems to make the program easier to use and understand.

