EECS 452 – Lecture 9

Today:	Finite precision FIR/IIR filter design Overflow and roundoff errors
Announcements:	Hw4 due on thursday. Oct 9 (Thu) lecture added to schedule: Real time embedded DSP Oct 10 (Fri): deadline for parts orders
References:	Please see last slide.

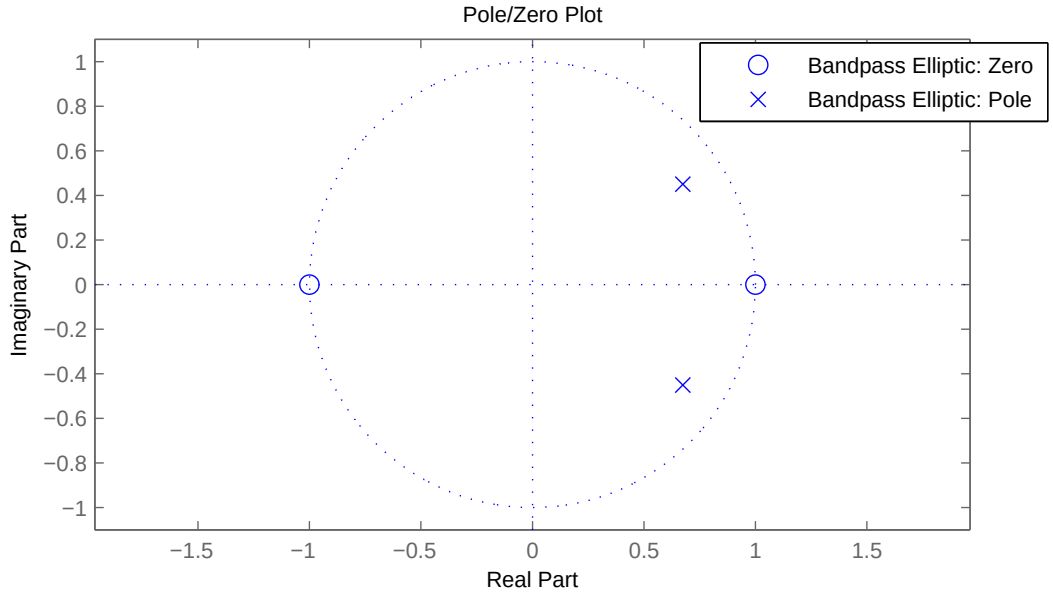
Last one out should close the lab door!!!!

Please keep the lab clean and organized.

You see things; and you say, "Why?" But I dream things that never were; and I say, "Why not?"

George Barnard Shaw

Effect of IIR coefficient quantization

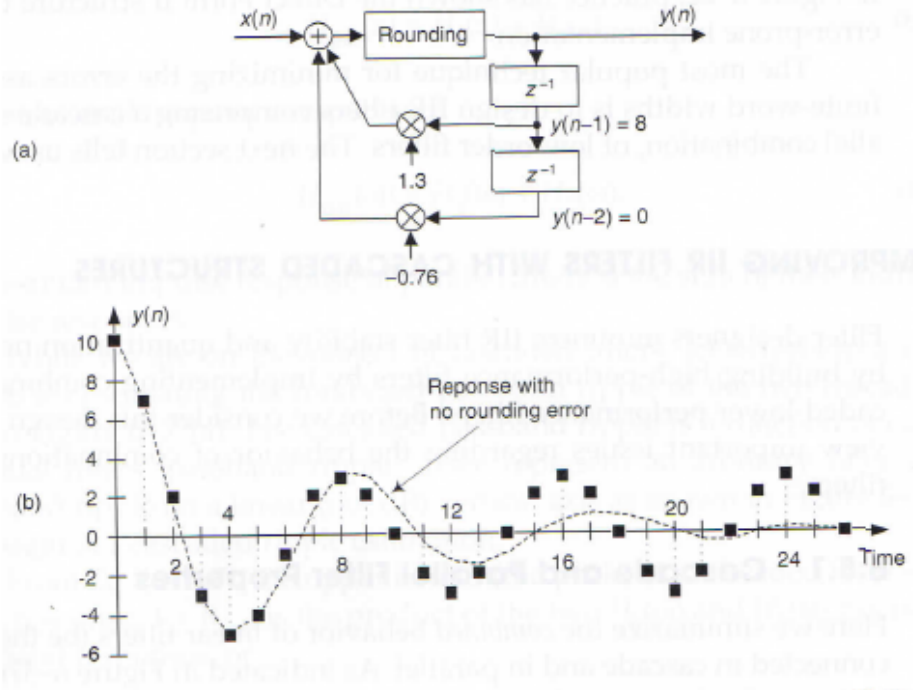


$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}}$$

- If quantize denominator coefficients poles might move closer to the unit circle.
- This can cause the filter to become unstable or quasi-stable.
  - Unstable: quantized pole outside of unit circle.
  - Quasi-stable: quantized pole on the unit circle.

IIR coefficient quantization and limit cycles (Lyons)

$$H(z) = \frac{1}{1 - 1.3z^{-1} + 0.76z^{-2}}$$

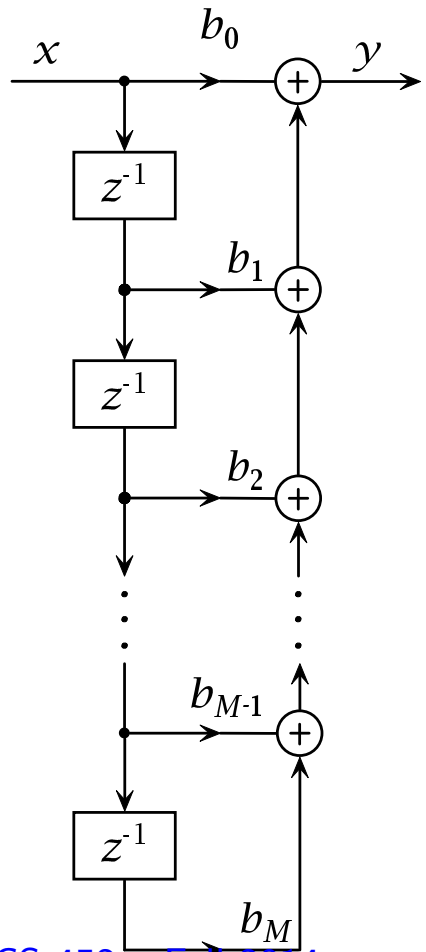


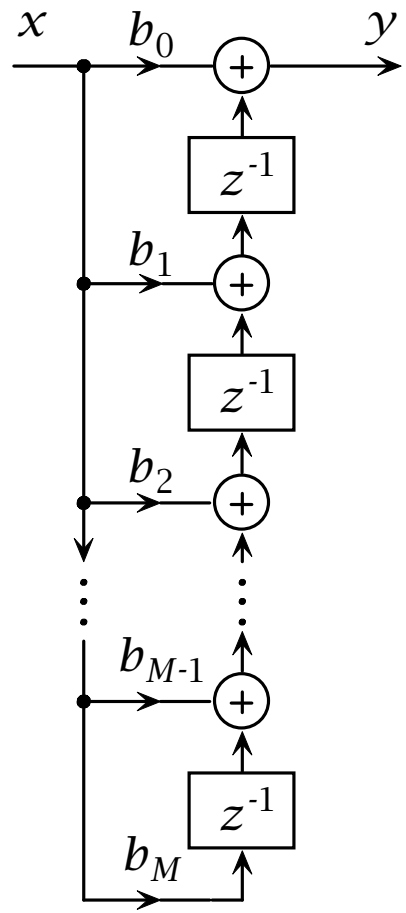
- Adder rounds its output to nearest integer.
- Eternal oscillation occurs when $y(-2) = 0$, $y(-1) = 8$ and $x[0] = x[1] = \cdots 0$.

Roundoff/overflow: FIR Direct and Transpose Forms

$$Y = b_0X + b_1(z^{-1}X) + \ldots + b_M(z^{-M}X)$$

$$Y = (b_0X) + (b_1X)z^{-1} + \ldots + (b_MX)z^{-M}$$





Controlling roundoff/overflow in FIR filters

$$y[n] = b_0 x[n] + b_1 x[n-1] + \cdots + b_M x[n-M]$$

- Roundoff errors and overflows may occur as the result of any multiply, add and accumulate (MAC) operations.
- To minimize effect of roundoff error use 16 bit Q15 binary and choose a design that minimizes dynamic range of coefficients.
- To limit overflow we will do the following:
  - Scale the input values $x[n]$ so that $|x[n]| \leq 1$. Given N input samples $x[0], x[1], \ldots, x[N-1]$:

$$x[n] \rightarrow \frac{x[n]}{\max\{|x[k]|\}_{k=1}^{N}}, \quad n = 0, \ldots, N-1$$

    .
  - Scale the FIR coefficients $b_0, \ldots, b_M$ so that $|b_k| \leq 1$ **and** $|y[n]| \leq 1$. Assume FIR coefficients determined by `fdatool`

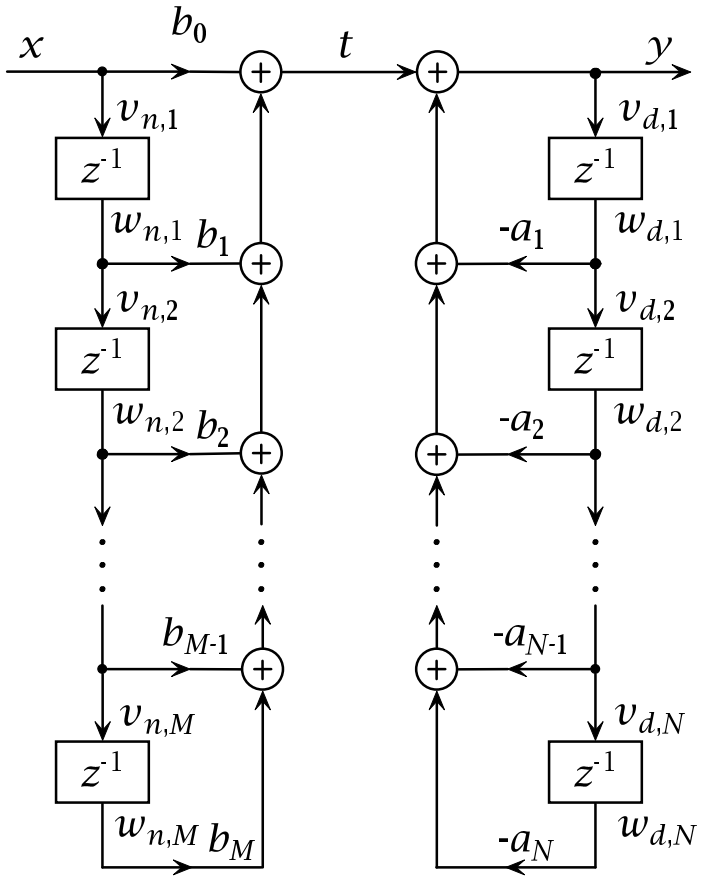
$$b_i \rightarrow \frac{b_i}{\sum_{k=0}^{M} |b_k|}, \quad i = 0, \ldots, M$$

- This will ensure no overflow in FIR filter MACs.

Roundoff/overflow: IIR Direct forms 1 and 2

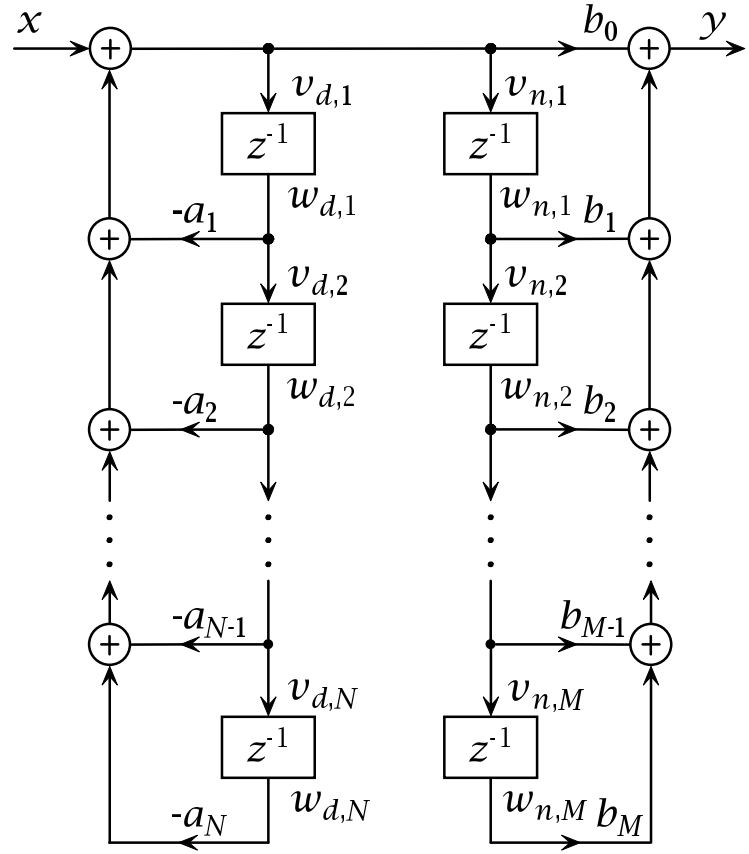
Direct Form 1 (DF1)

$$H(z) = B(z) \times \frac{1}{A(z)}$$

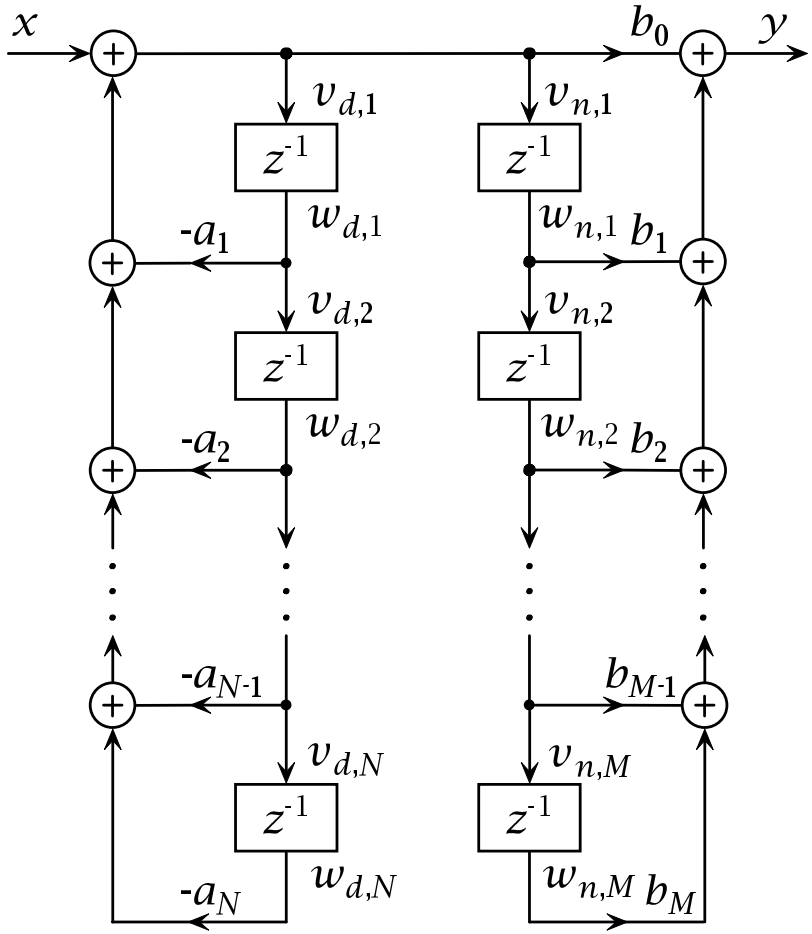


Direct Form 2 (DF2)

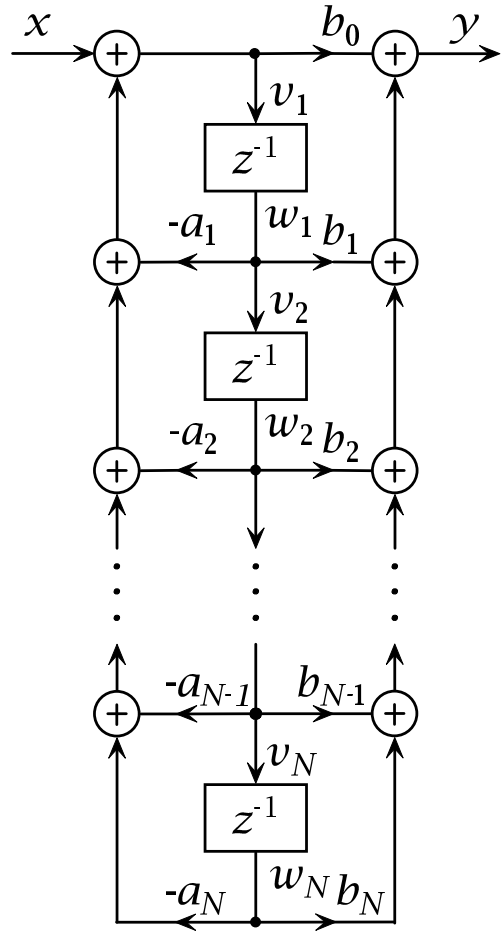
$$H(z) = \frac{1}{A(z)} \times B(z)$$



IIR Canonical direct form 2



(a)



(b)

a) Non-canonical Direct Form 2.

b) DF2 in canonical form.

Overflow sensitivity of IIR filters.

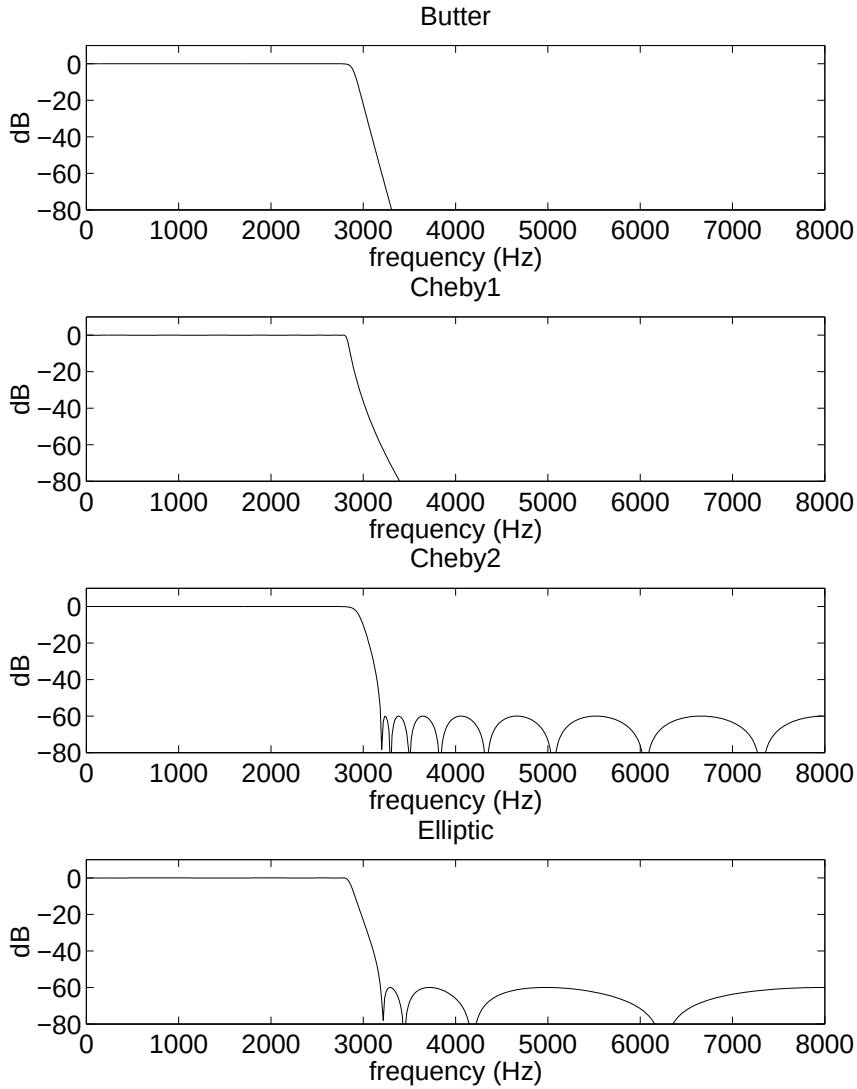
Controlling roundoff and overflow in IIR filters is more complicated than in FIR

Some IIR design approaches are more sensitive than others.

There are many ways to design IIR filters:

- Butterworth — maximally flat
- Chebyshev
  - Type 1: equiripple in the passband, monotone in the stopband.
  - Type 2: monotone in the pass band, equiripple in the stopband.
- elliptic — equiripple in the passband and in the stopband. Optimal in the sense that for a given order it passes through the smallest possible transition band.

Properties of these filters



Butterworth. Maximally flat in the pass band. Monotonic roll off in the stop band.

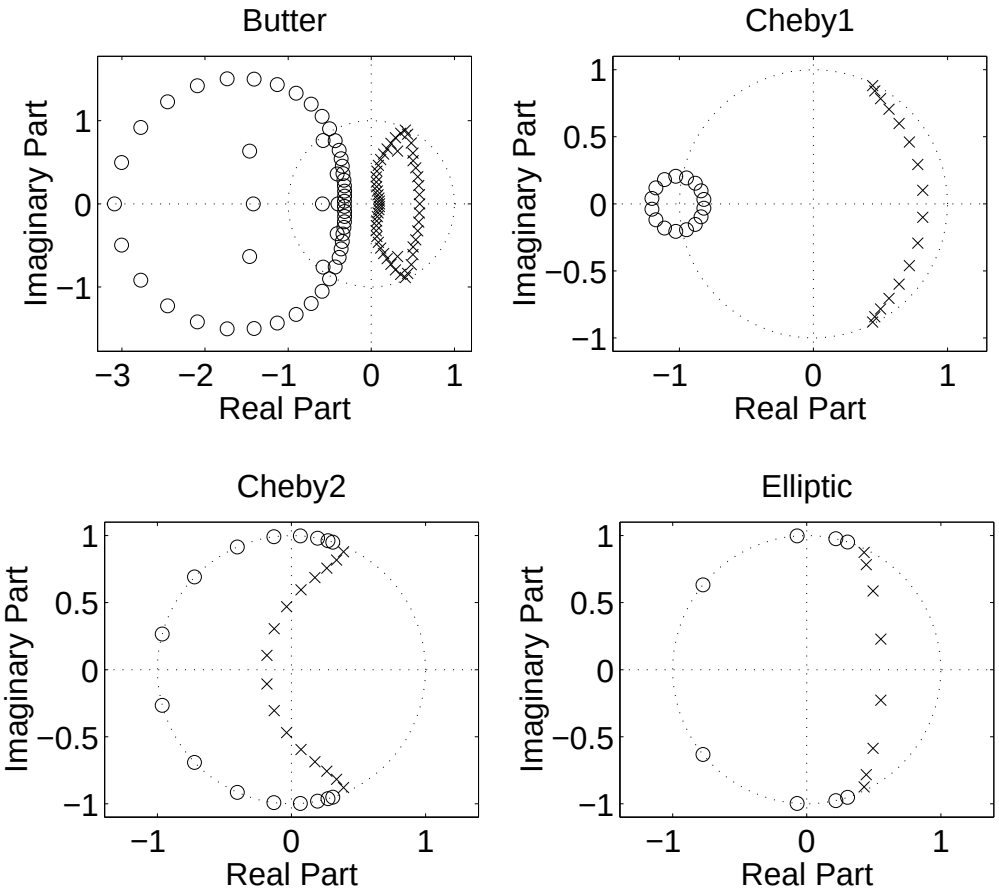
Chebyshev type 1. Equiripple in the pass band. Monotonic roll off in the stop band.

Chebyshev type 2. Monotonic in the pass band. Equiripple in the stop band.

Elliptic. Equiripple both in the pass and stop bands. Optimal transition between pass and stop bands.

Pole and Zero Locations

Butterworth:	52 poles
Chebyshev 1:	16 poles
Chebyshev 2:	16 poles
Elliptic:	8 poles



Butterworth filter coefficient values

```
Butter
Numerator coefficients (53)
  0.0000000000   0.0000000000   0.0000000000   0.0000000000   0.0000000000
  0.0000000000   0.0000000000   0.0000000000   0.0000000001   0.0000000004
  0.0000000016   0.0000000060   0.0000000206   0.0000000635   0.0000001768
  0.0000004479   0.0000010359   0.0000021936   0.0000042653   0.0000076327
  0.0000125939   0.0000191907   0.0000270414   0.0000352714   0.0000426196
  0.0000477340   0.0000495699   0.0000477340   0.0000426196   0.0000352714
  0.0000270414   0.0000191907   0.0000125939   0.0000076327   0.0000042653
  0.0000021936   0.0000010359   0.0000004479   0.0000001768   0.0000000635
  0.0000000206   0.0000000060   0.0000000016   0.0000000004   0.0000000001
  0.0000000000   0.0000000000   0.0000000000   0.0000000000   0.0000000000
  0.0000000000   0.0000000000   0.0000000000
Denominator coefficients (53)
      1.000000     -14.485235     108.892629    -560.590282    2207.916845
  -7059.412172   19007.953411  -44181.160327   90234.032916 -164088.539461
 268398.869177 -398070.141447  538774.788242 -668949.194632  765208.835443
-809294.988642  793688.087323 -723548.120952  614376.363748 -486708.393544
 360206.142051 -249310.489583  161505.343889  -97980.476106   55687.060582
 -29655.011284   14796.263025   -6915.328167    3026.163865   -1239.130105
    474.376094    -169.610425      56.564993     -17.568725       5.072718
     -1.358702       0.336753      -0.077009       0.016194      -0.003119
      0.000548      -0.000087       0.000013      -0.000002       0.000000
     -0.000000       0.000000      -0.000000       0.000000      -0.000000
      0.000000      -0.000000       0.000000
```

Chebyshev 1 and 2 filter coefficient values

```
Cheby1
Numerator coefficients (17)
  0.0000000085   0.0000001356   0.0000010173   0.0000047474   0.0000154290
  0.0000370296   0.0000678875   0.0000969822   0.0001091049   0.0000969822
  0.0000678875   0.0000370296   0.0000154290   0.0000047474   0.0000010173
  0.0000001356   0.0000000085
Denominator coefficients
      1.000000      -9.817865      48.302917    -156.772970     373.680906
   -691.001892    1022.672308   -1233.283016    1223.105455   -1000.026974
    671.545993    -366.484942     159.400953     -53.461355      13.056916
     -2.078911       0.163038

Cheby2
Numerator coefficients (17)
  0.0094956544   0.0261944594   0.0770790872   0.1525062902   0.2717122008
  0.4072371571   0.5434552757   0.6418315575   0.6787499757   0.6418315575
  0.5434552757   0.4072371571   0.2717122008   0.1525062902   0.0770790872
  0.0261944594   0.0094956544
Denominator coefficients
      1.000000      -1.777756       4.322853      -4.364343       5.779933
     -3.561359       3.418699      -1.144368       1.056520      -0.077544
      0.205087       0.036841       0.030444       0.009021       0.003002
      0.000653       0.000090
```

Elliptic filter coefficient values

```
Elliptic
Numerator coefficients (9)
  0.0090414378   0.0058832667   0.0246209031   0.0195809558   0.0316918245
  0.0195809558   0.0246209031   0.0058832667   0.0090414378
Denominator coefficients (9)
      1.000000      -3.841248       8.219775     -11.387778      11.100736
     -7.650511       3.645340      -1.096464       0.161832
```

Overflow IIR filter comparisons

The Butterworth realization has the most coefficients and they have large dynamic range. This isn't going to implement well on a 16-bit processor or even on a 32 bit processor.

The Chebyshev coefficients probably are less a problem but the range in values is still large.

The Elliptic filter has the fewest coefficient values.

All of the filter realizations have some coefficient values significantly larger than 1. Some sort of scaling procedure must be implemented.

Overflow and saturation in IIR filters

- Adder and MAC overflows cause very significant I/O distortion and errors.
- Must protect against overflows at all IIR signal nodes: internal overflows.
- Two ways to correct for overflow:
  - Scaling the input down to an acceptable level.
  - Reorganizing IIR filter operations to minimize overflow at every node.
- Design approaches
  - Scaling: need find the right scale factor for no overflow.
  - Reorganizing filter realization: use most robust implementations of transfer function (DF 1 vs DF 2).
- You will explore these approaches in Lab 5.

Scaling the input samples to avoid overflow

Consider a sequence of input values $\{x[n]\}$ such that $-1 < x[n] < 1$ for all n. The input values are to be divided by a scale factor S. Three values commonly used for S are:

$$S = \sum_{n=0}^{\infty} |h[n]| \qquad = \|h\|_1 = \mathrm{L}_1 \text{ norm } of \ h$$

$$S = \left(\sum_{n=0}^{\infty} h^2[n] \right)^{1/2} \qquad = \|h\|_2 = \mathrm{L}_2 \text{ norm } of \ h$$

$$S = \max_f |H(f)| \qquad = \|H\|_\infty = \mathrm{L}_\infty \text{ norm } of \ H$$

$h = \{h[n]\}$ is the filter impulse response

$H(f) = H_z(e^{j2\pi f})$ is transfer function over (digital) frequency $f \in [0, 1]$.

Which norm to use?

- The output of filter is $y[k] = \sum_n h[n]x[k-n]$.
- If $\|y\|_\infty = \max_k |y[k]| \leq 1$ there will be no overflow.

1. Normalization of $x[n]$ by $\|h\|_1$ guarantees there will be no overflow.[a]

2. Normalization by $\|h\|_2$ does not give overflow guarantees.[b]

3. Normalization by $\|H\|_\infty$ guarantees that no sinewave input will cause overflow. However, other types of inputs may still cause overflow.

Nonetheless, due to its relative simplicity, we will only consider $\|H\|_\infty$ normalization here.

---

[a] The well known modulus bound of functional analysis implies $\left|\sum_n h[n]x[k-n]\right| \leq \sum_n |h[n]|\,|x[k-n]|$

[b] Normalization by $\sqrt{M}\|h\|_2$ does provide guarantees for M-th order FIR filters due to another result of functional analysis: $\|h\|_2 \leq \|h\|_1 \leq \sqrt{M}\|h\|_2$.

Implementing TFs in factored form

Unfactored transfer functions, in a sense, implement the poles and zeros all at once. A small quantization error in a coefficient value can affect all aspects of the filter.

Implementing the poles and zeros individually may result in a more robust implementation.

As we are only considering filters having real valued coefficients, any complex valued poles or zeros must appear in conjugate pairs.

This leads to idea of factoring transfer functions into the ratio of the product of quadratic factors.

Using biquad sections to implement a TF

To keep things simple we will assume that $H(z)$ has an equal even number of poles and zeros. An IIR transfer function can be written in factored form as

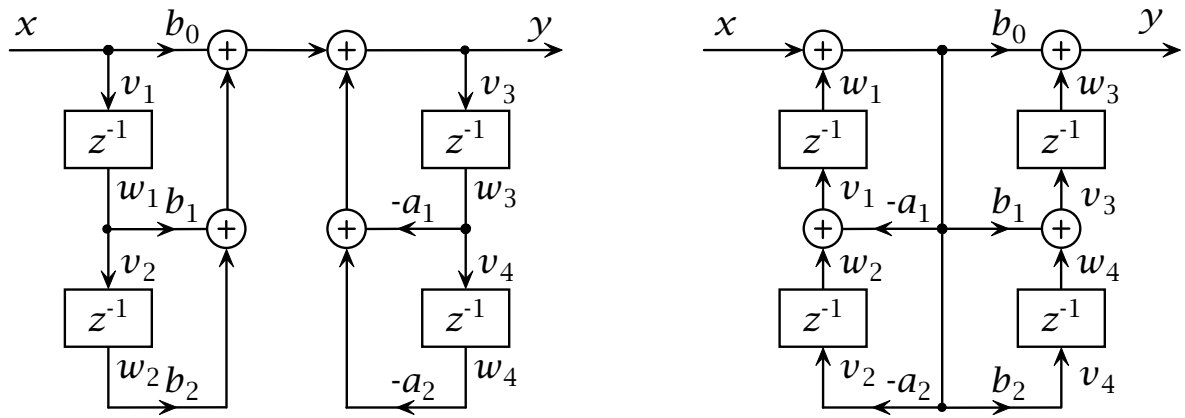
$$H(z) = \frac{\prod_{s=0}^{N/2-1} b_{s,0} + b_{s,1}z^{-1} + b_{s,2}z^{-2}}{\prod_{r=0}^{N/2-1} 1 + a_{r,1}z^{-1} + a_{r,2}z^{-2}} .$$

A biquadratic filter section can be used to implement a set of zeros and a set of poles.

$$H(z) = \prod_{r=0}^{N/2-1} \frac{b_{r,0} + b_{r,1}z^{-1} + b_{r,2}z^{-2}}{1 + a_{r,1}z^{-1} + a_{r,2}z^{-2}} .$$

Design principle: pair up zeros and poles and order the resulting biquad sections to obtain the best performance.

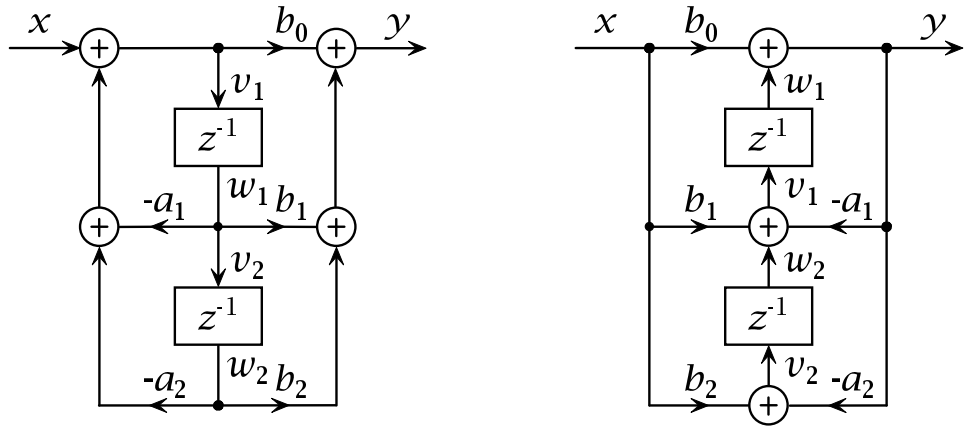
The DF1 and TDF1 biquad sections



(a) Direct form type 1 biquad section. (b) Transposed direct form 1 biquad section. These are not canonical.

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

The DF2 and TDF2 biquad sections

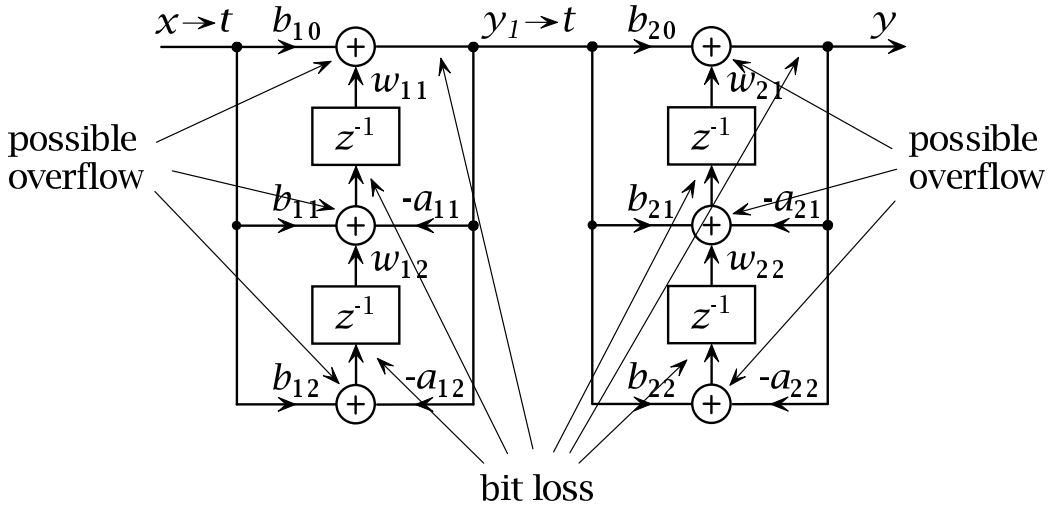


(a) Direct form type 2 biquad section. (b) Transposed direct form 2 biquad section.

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

The most commonly used biquad is direct form 2. We need to analyze the transfer function magnitudes between input and internal states in addition to between input and output.

TDF2 and DF2 biquad overflow concerns



The filters to be designed for use in the lab will nominally have input to output passband gain of 1. This sets the overall transfer function gain level. Of concern is whether or not this level will give rise to overflows internal to the filter.

In order to check for this possibility in the above (TDF2) block diagram we are interested in the transfer functions from the input x to the values y_i, w_{i1} and w_{i2} for each section i. These are the locations at which overflows can occur.

Input-to-delay stage transfer functions: DF2

Input-to-v_1 transfer function (in z-domain) is easily determined

$$V_1 = X - a_1 z^{-1} V_1 - a_2 z^{-2} V_1 \, .$$

The transfer function between X and V_1 is

$$\frac{V_1}{X} = \frac{1}{1 + a_1 z^{-1} + a_2 z^{-2}} \, .$$

The transfer function between X and W_1 is then

$$\frac{W_1}{X} = \frac{z^{-1}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

and between X and W_2

$$\frac{W_2}{X} = \frac{z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \, .$$

Input-to-delay stage transfer functions: DF2

The section input-to-delay stage transfer functions:

$$H_{X\to W_1}(z) = \frac{W_1(z)}{X(z)}, \qquad H_{X\to W_2}(z) = \frac{W_2(z)}{X(z)}, \qquad H_{X\to V_1}(z) = \frac{V_1(z)}{X(z)}$$

These are related by:

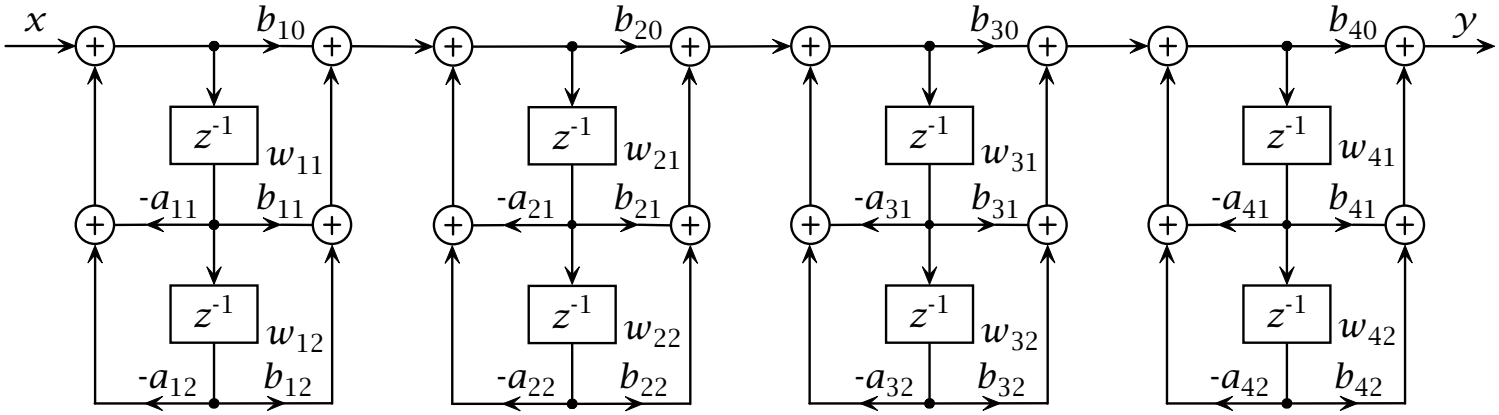
$$\frac{W_2}{X} = z^{-1}\frac{W_1}{X} = z^{-2}\frac{V_1}{X}$$

The magnitudes **are the same** on the unit circle $z = e^{j2\pi f}$:

$$\left|\frac{W_2}{X}\right| = |e^{-j2\pi f}|\left|\frac{W_1}{X}\right| = |z = e^{-j4\pi f}|\left|\frac{V_1}{X}\right|$$

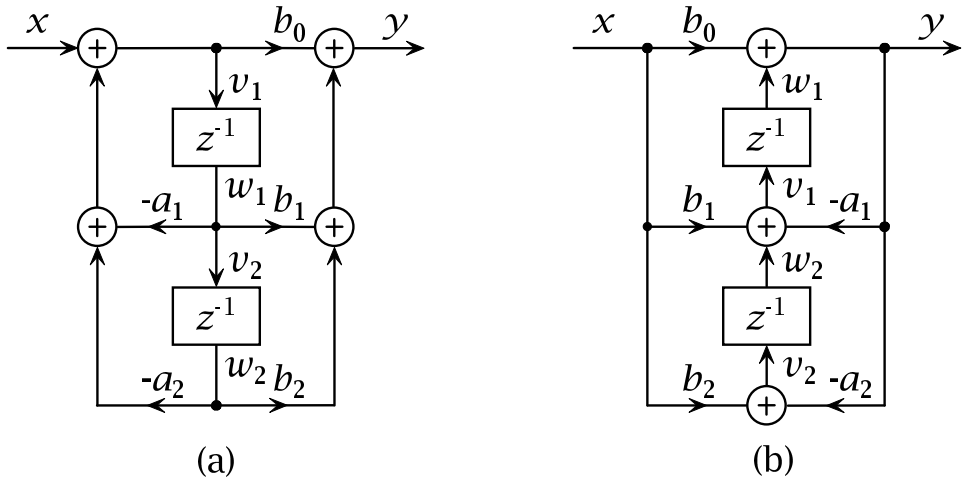
since $|e^{-j2\pi f}| = |e^{-j4\pi f}| = 1$.

DF2 biquad cascade



The above block diagram shows a cascade of four DF2 second order biquad sections. This can be used to implement an eighth order lowpass filter.

The DF2 and TDF2 biquad sections (again)



(a) Direct form type 2 biquad section. (b) Transposed direct form 2 biquad section.

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

The most commonly used biquad is direct form 2. We need to analyze the transfer function magnitudes between input and internal states in addition to between input and output.

Input-to-delay stage transfer functions: TDF2

For the TDF2 section the z-transformed equations are

$$\begin{aligned} Y &= b_0 X + z^{-1} V_1, \\ V_1 &= b_1 X - a_1 Y + z^{-1} V_2, \\ V_2 &= b_2 X - a_2 Y. \end{aligned}$$

Substituting the equation for Y into the equations for V_1 and V_2 gives

$$\begin{aligned} V_1 &= b_1 X - a_1 b_0 X - a_1 z^{-1} V_1 + z^{-1} V_2, \\ V_2 &= b_2 X - a_2 b_0 X - a_2 z^{-1} V_1. \end{aligned}$$

$$\begin{bmatrix} 1 + a_1 z^{-1} & -z^{-1} \\ a_2 z^{-1} & 1 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} (b_1 - a_1 b_0) \\ (b_2 - a_2 b_0) \end{bmatrix} X.$$

Solving for the TDF2 biquad TFs

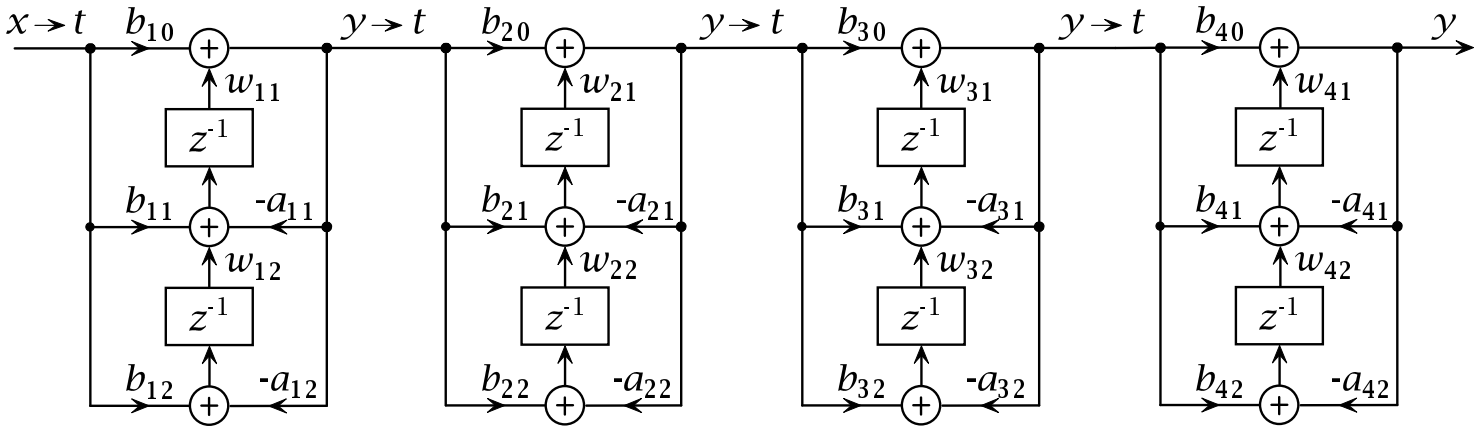
The two-by-two matrix is easily inverted giving transfer functions, in non-matrix form,

$$\frac{V_1}{X} = \frac{b_1 - a_1 b_0 + (b_2 - a_2 b_0) z^{-1}}{1 + a_1 z^{-1} + a_2 z^{-2}},$$

$$\frac{V_2}{X} = \frac{b_2 - a_2 b_0 + (a_1 b_2 - a_2 b_1) z^{-1}}{1 + a_1 z^{-1} + a_2 z^{-2}}.$$

$$\frac{W_1}{X} = z^{-1} \frac{V_1}{X} \quad \text{and} \quad \frac{W_2}{X} = z^{-1} \frac{V_2}{X}.$$

TDF2 biquad cascade

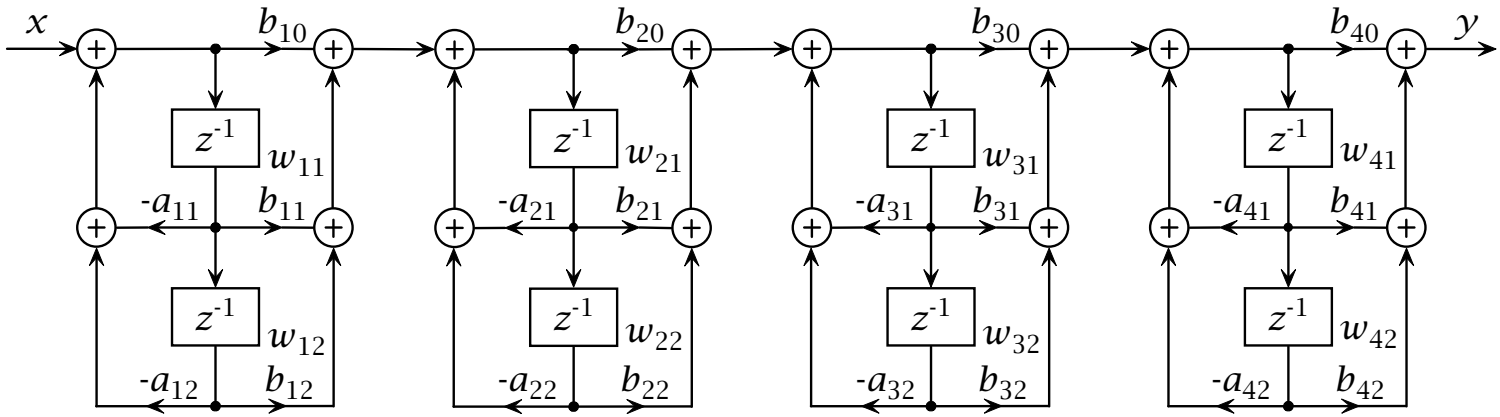


The above block diagram shows a cascade of four TDF2 second order biquad sections. This can be used to implement an eighth order lowpass filter.

Multistage analysis of overflow proclivity

- We can now illustrate the analysis of overflow of DF2 and TDF2.
- Given: a quad factorization of the IIR filter
- Given: the input-to-stage transfer functions H found above
- The procedure is to evaluate $\max_f |H(f)| = \| H \|_\infty$ for each stage and scale the input by the maximum over all the stages.
- This will result in:
  - Guarantee of no-overflow for **sinusoidal inputs**.
  - For other inputs will need to experimentally validate.
- Note: the following matter a lot
  - The matching of pole pairs and zero pairs for each biquad section.
  - The cascade order of the biquad sections.
- Pairing and cascade ordering are non trivial
- $M!$ possible ways of permuting the cascade order.

DF2 biquad cascade transfer functions



Write $H_i = \dfrac{Y_i}{X_i}$, $H_{i1} = \dfrac{W_{i1}}{X_i}$ and $H_{i2} = \dfrac{W_{i2}}{X_i}$. Because of our choice of the L_∞ norm we are interested in the magnitudes of the input to delay stage filter functions:

section 1	H_{11}	H_1
section 2	$H_1 H_{21}$	$H_1 H_2$
section 3	$H_1 H_2 H_{31}$	$H_1 H_2 H_3$
section 4	$H_1 H_2 H_3 H_{41}$	$H_1 H_2 H_3 H_4$

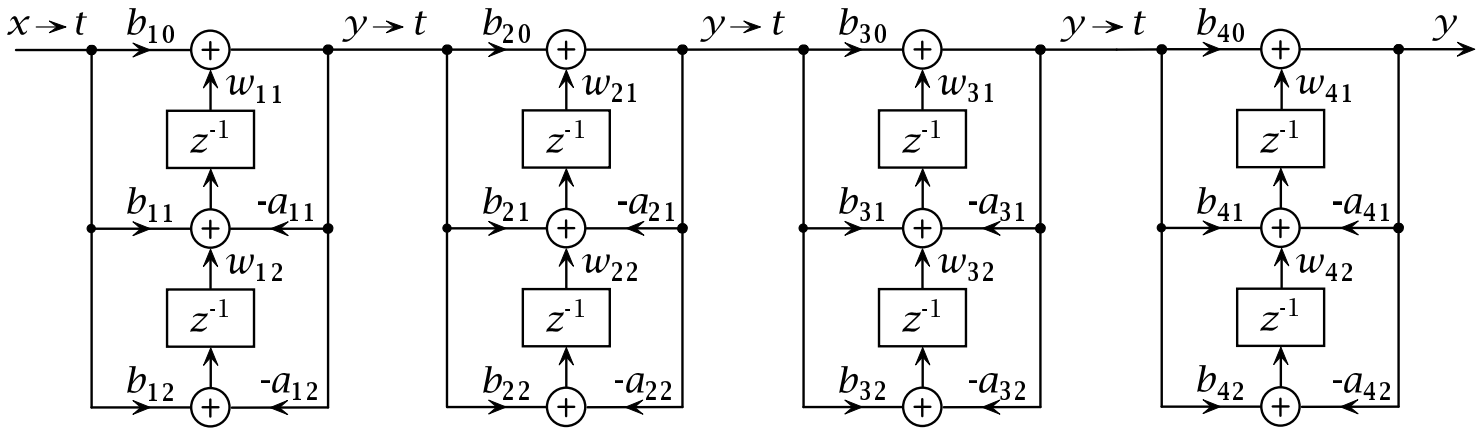
DF2 section delay output magnitudes are equal

Within a section the *magnitudes* of the section input to the delay stage outputs are equal.

If we evaluate the transfer function magnitude for one delay stage we have the same result for the other delay stage output.

We don't have to determine the input to delay stage output transfer functions for both delay stages in a DF2 biquad. A work savings.

TDF2 biquad cascade transfer functions



Write $H_i = \dfrac{Y_i}{X_i}$, $H_{i1} = \dfrac{W_{i1}}{X_i}$ and $H_{i2} = \dfrac{W_{i2}}{X_i}$. Because of our choice of the L_∞ norm we are interested in the magnitudes of the input to delay stage filter functions:

section 1	H_{11}	H_{12}	H_1
section 2	$H_1 H_{21}$	$H_1 H_{22}$	$H_1 H_2$
section 3	$H_1 H_2 H_{31}$	$H_1 H_2 H_{32}$	$H_1 H_2 H_3$
section 4	$H_1 H_2 H_3 H_{41}$	$H_1 H_2 H_3 H_{42}$	$H_1 H_2 H_3 H_4$

Overflow resistant IIR filter implementation

Goal: implement IIR LPF filter[a] to meet given specifications.

Procedure: Use MATLAB for design and TI supplied IIR filter functions (DSPlib) for implementation.

What could possibly go wrong?

- TI supplied functions are mostly DF2. The DF2 has internal resonance peaks leading to gain exceeding one and resulting in overflow for Q15 inputs and outputs.
- MATLAB design may give coefficient values that exceed one, leading to Q15 overflow.

First we deal with **gain-induced overflow** of internal states:

- Distributing the gain over biquad sections.
- Scaling the input $x[n]$ to avoid internal overflow.

[a] While we will not discuss it here, overflow resistant implementation of other types of IIR filters (BPF, BSF, etc) is similar.

Controlling gain-induced overflow

1. Design IIR LPF filter using MATLAB and obtain poles and zeros of TF $H_z(z)$.

2. **Construct biquad factorization**: $H_z(z) = H_z^{(1)}(z) \dots H_z^{(M)}(z)$
   - 2.1 List the poles of H_z in order of decreasing distance to unit circle (pole closest to unit circle is at bottom of list).
   - 2.2 For last pole on list match it with the zero closest to it.
   - 2.3 Group this matched pair with the conjugate pair ($h[k]$ is real valued).
   - 2.4 Progress up the list and construct successive biquad sections.

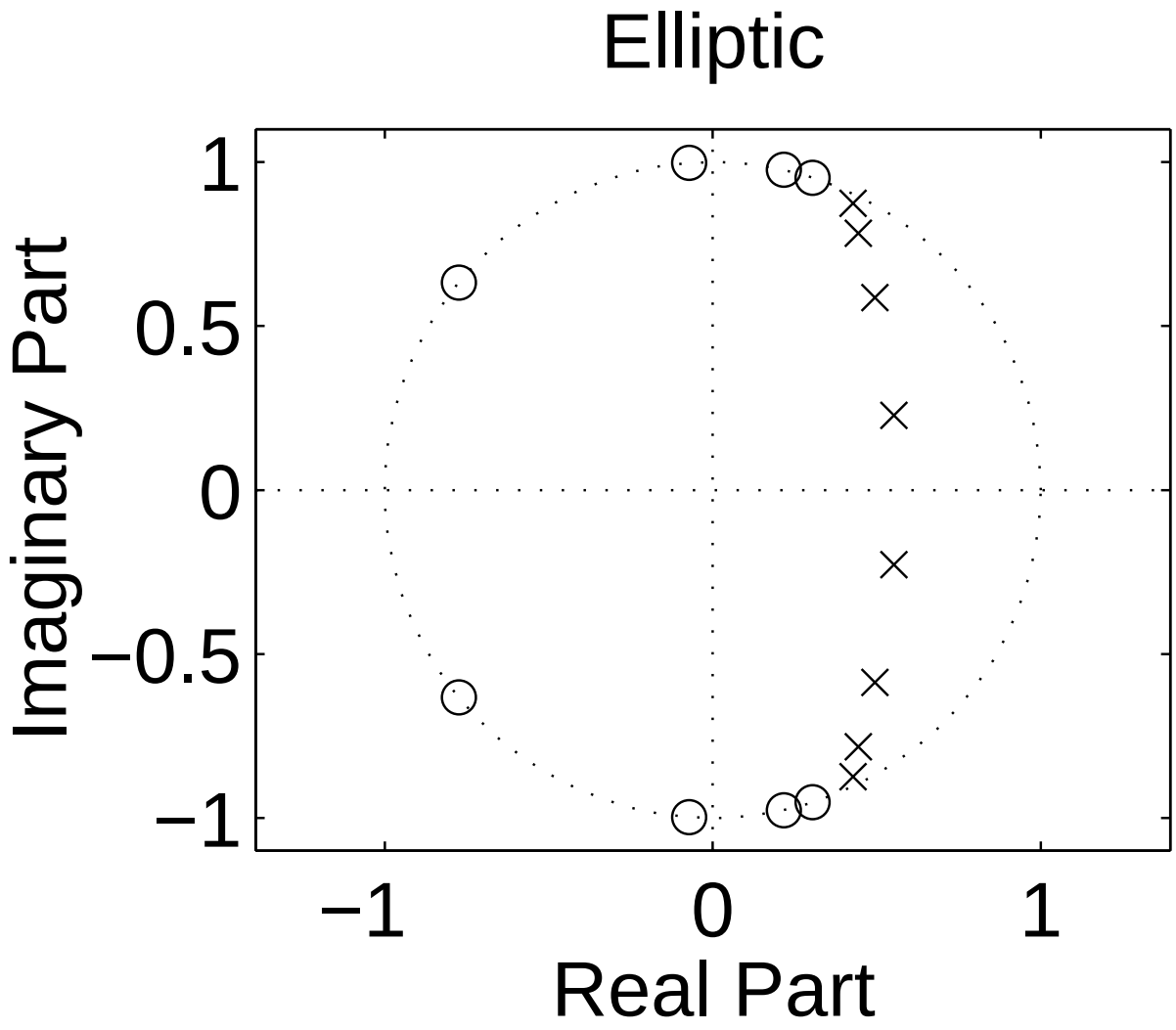
3. **Scale numerator coefficients**: force each biquad to have unit gain at DC $\left(\frac{b_0+b_1+b_2}{1+a_1+a_2} = 1\right)$. (Note: This does **not** guarantee internal gains less than one).

4. Compute input-to-delay stage TF $H_{x\to i}(f)$ for all internal states i.

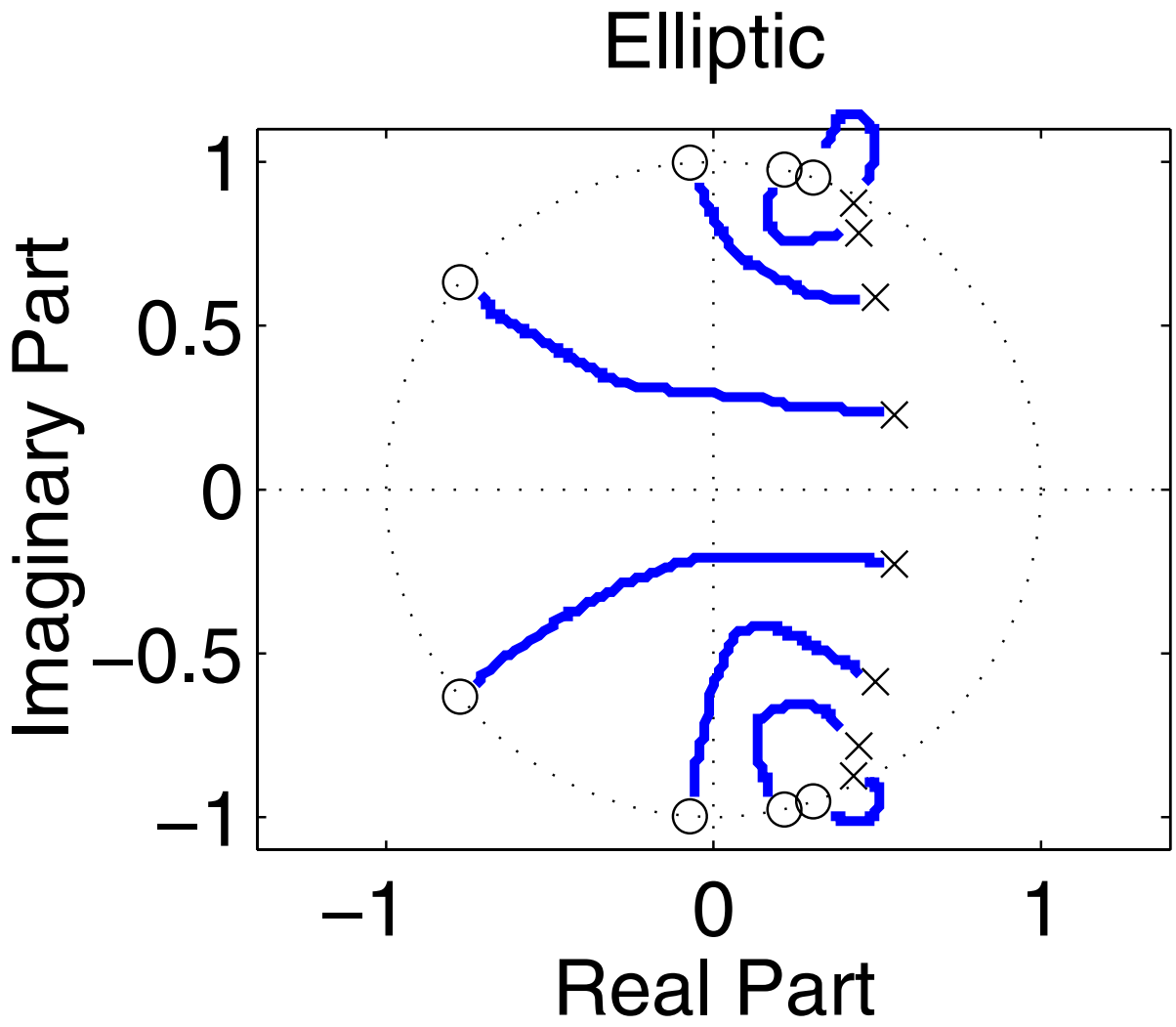
5. **Scale input** $x[n]$ by replacing by $x[n]/S$ where $S = \max_i \max_f |H_{x\to i}(f)|$.

6. Can we use Q15 for numerator and denominator coefficients of biquads?. (Below)

Pole - zero pairing example



Pole - zero pairing example



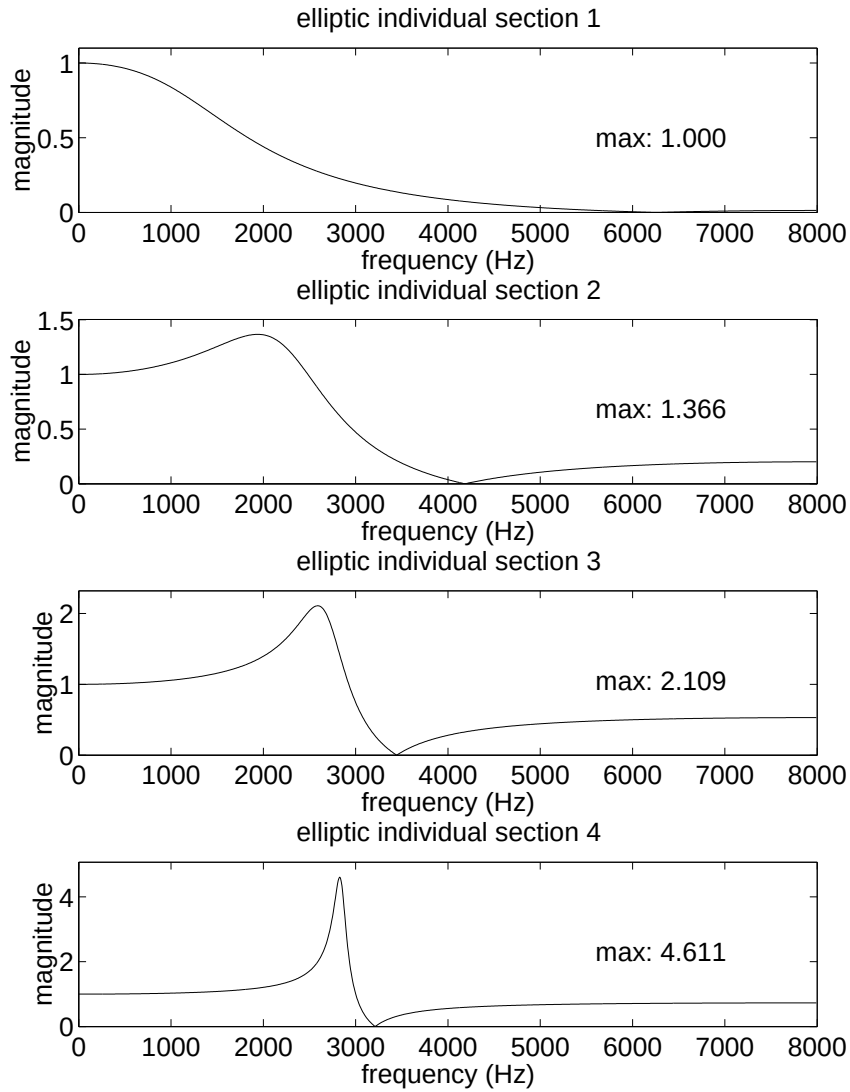
Example of individual biquad TFs

The four plots show the transfer functions of the biquad sections for an eighth order elliptic lowpass filter.

Going from the top down we have

$$H_1(f)$$
$$H_2(f)$$
$$H_3(f)$$
$$H_4(f)$$

The sections are ordered lowest Q to highest Q.

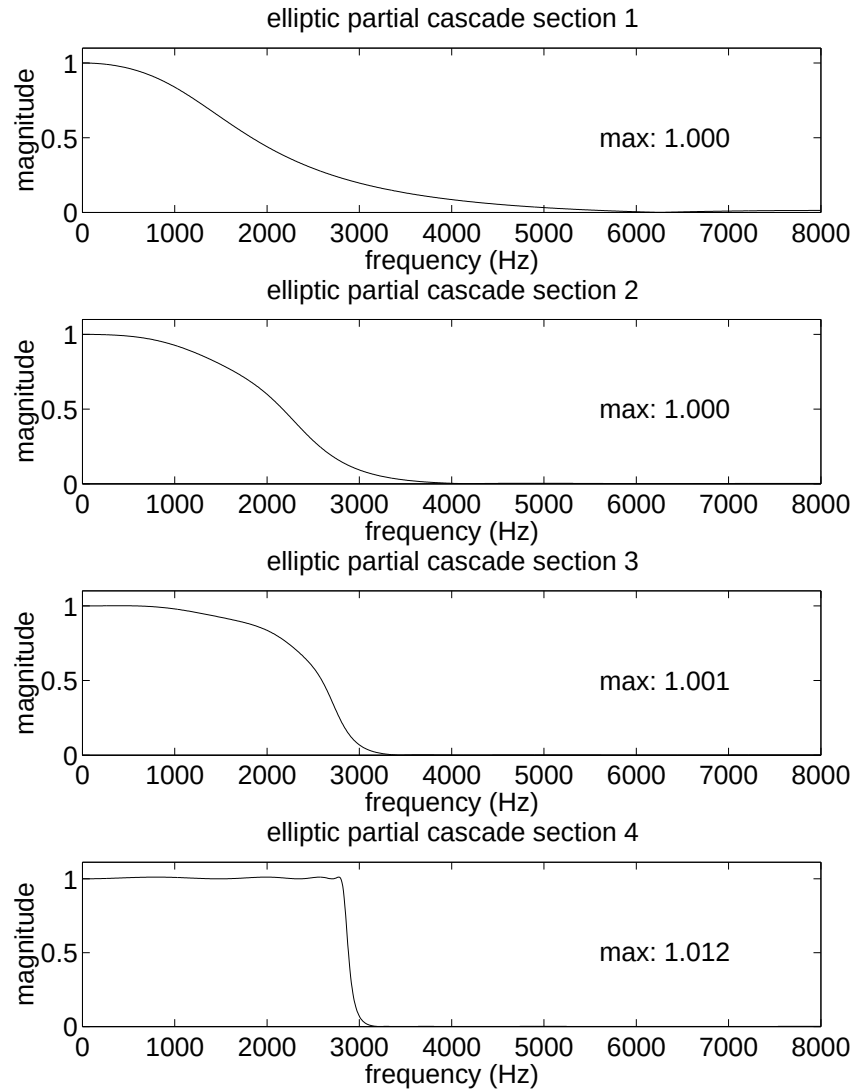


Example of moving through a cascade

The four plots show the development of the transfer function of an eighth order elliptic lowpass filter as one moves *between* sections going from input to output.

Going from the top down we have

$H_1(f)$
$H_1(f)H_2(f)$
$H_1(f)H_2(f)H_3(f)$
$H_1(f)H_2(f)H_3(f)H_4(f) = H(f)$

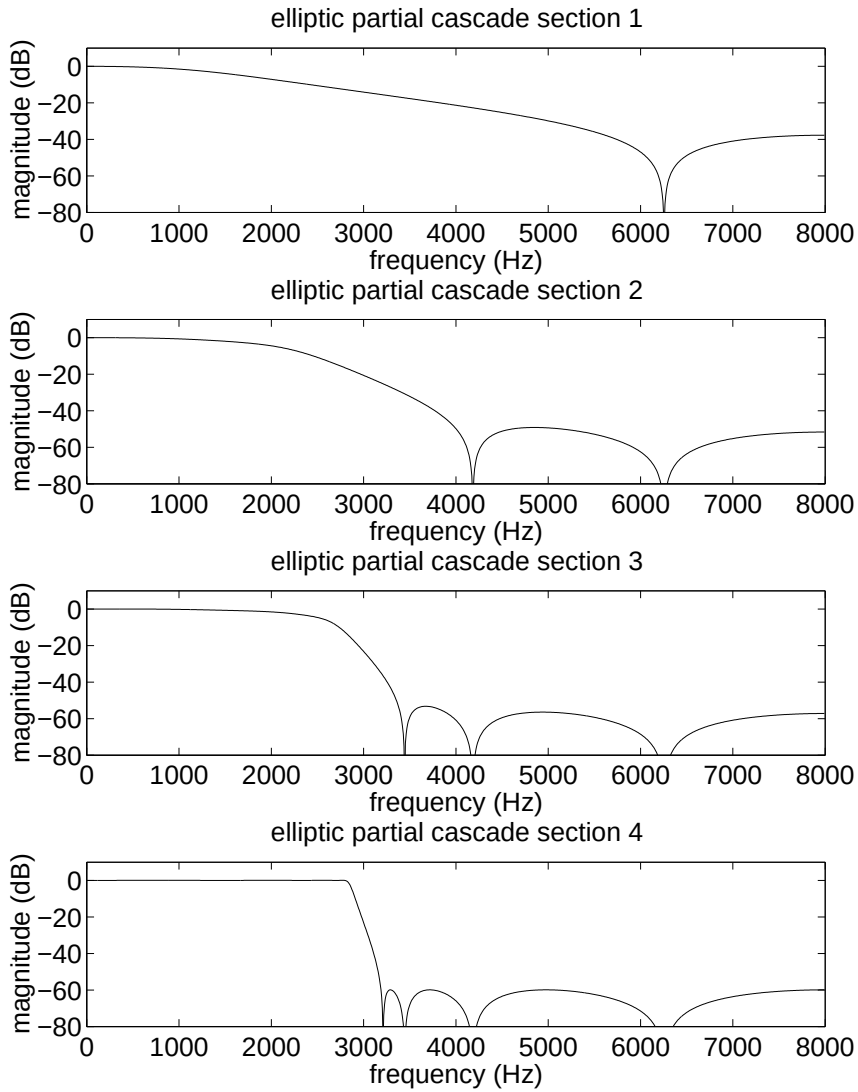


Example of moving through a cascade (dB)

The four plots show the development of the transfer function of an eighth order elliptic lowpass filter as one moves between sections going from input to output.

Going from the top down we have

$H_1(f)$
$H_1(f)H_2(f)$
$H_1(f)H_2(f)H_3(f)$
$H_1(f)H_2(f)H_3(f)H_4(f) = H(f)$

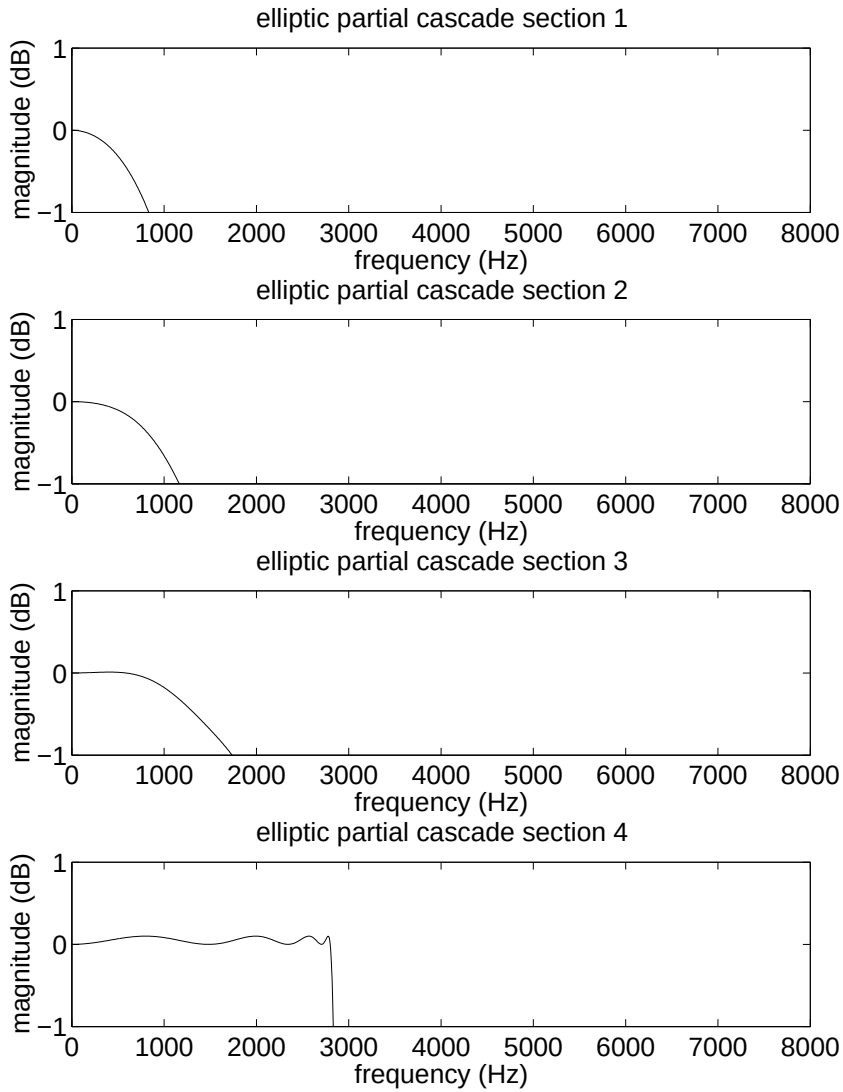


Expanded Passband — Moving Through Cascade (dB)

The four plots show the development of the transfer function of an eighth order elliptic low pass filter as one moves between sections going from input to output.

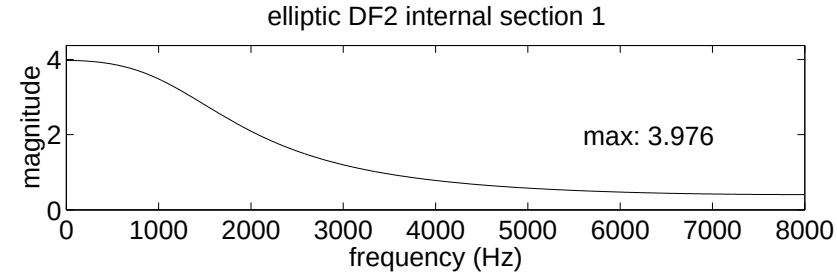
Going from the top down we have

$H_1(f)$
$H_1(f)H_2(f)$
$H_1(f)H_2(f)H_3(f)$
$H_1(f)H_2(f)H_3(f)H_4(f) = H(f)$

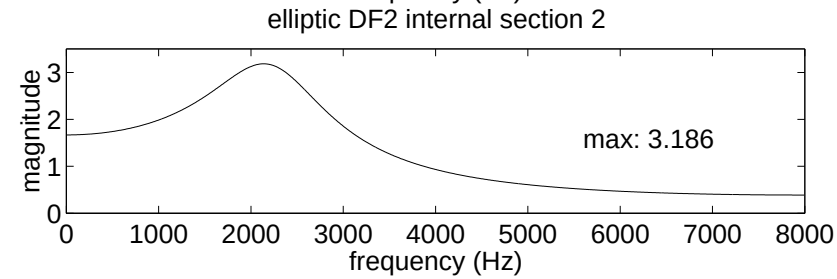


Biquad input-to-delay stage TFs: DF2

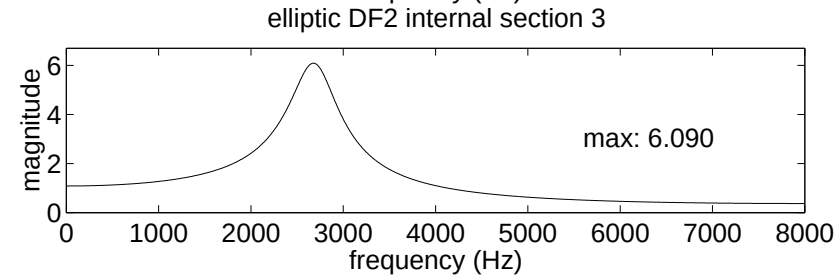
$|H_{11}(f)|$



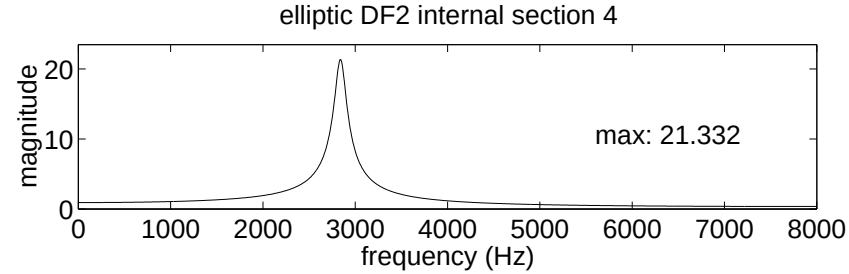
$|H_{21}(f)|$



$|H_{31}(f)|$

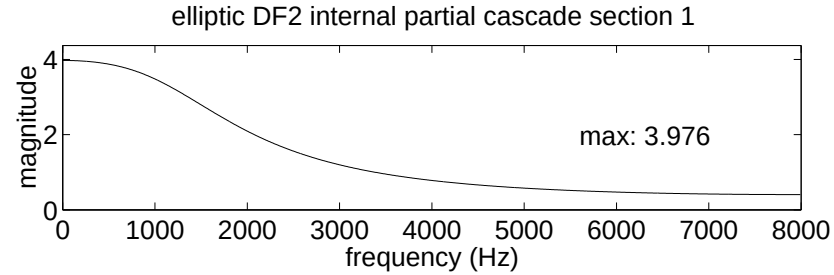


$|H_{41}(f)|$

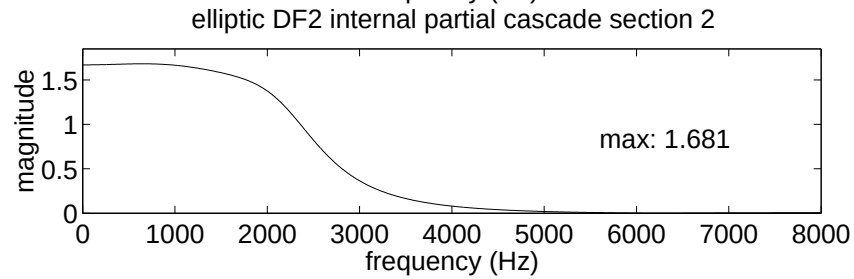


Filter input-to-delay stage TFs: DF2

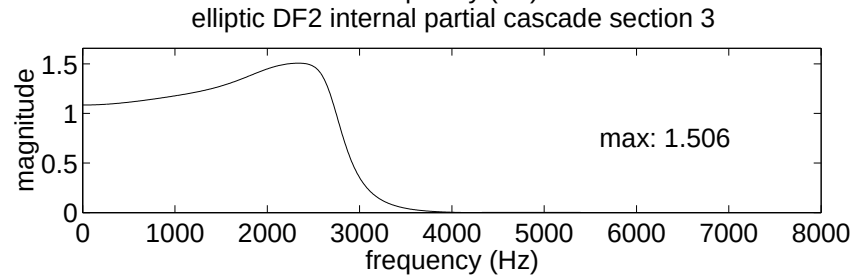
$|H_{11}(f)|$



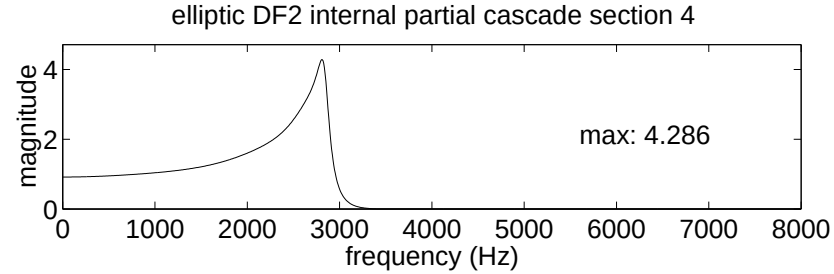
$|H_1(f)H_{21}(f)|$



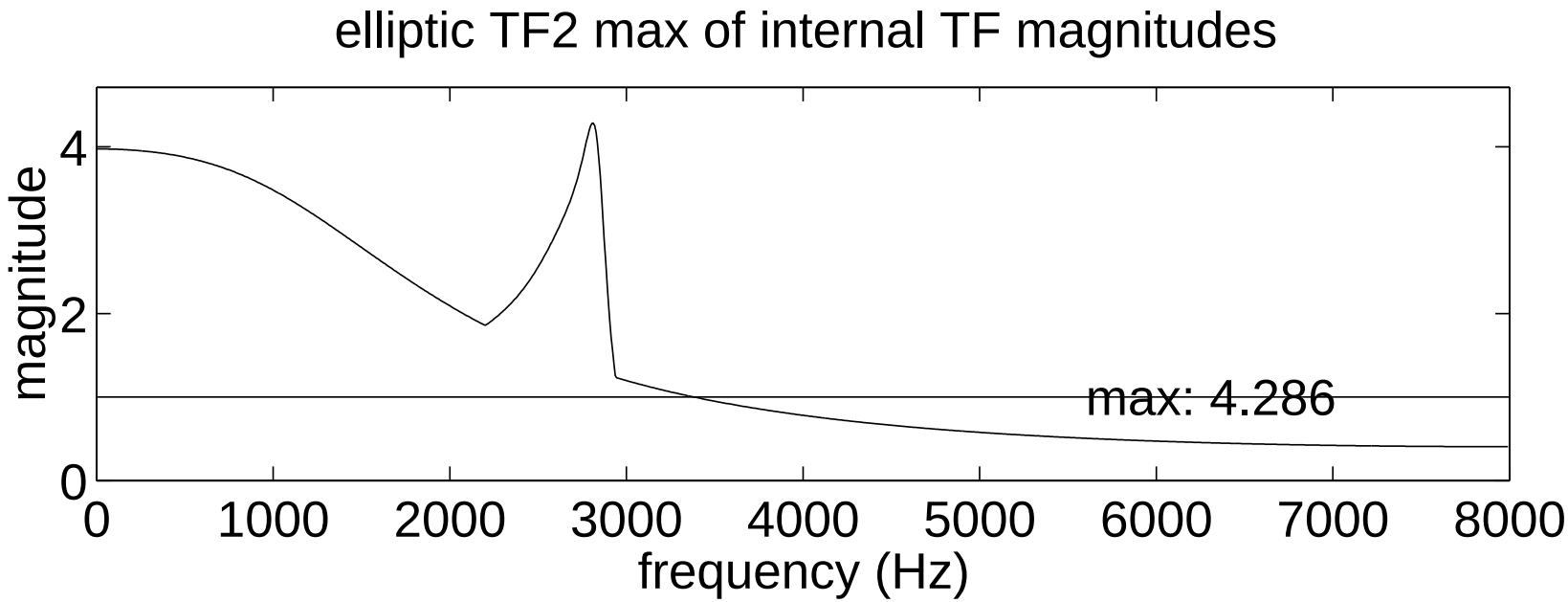
$|H_1(f)H_2(f)H_{31}(f)|$



$|H_1(f)H_2(f)H_3(f)H_{41}(f)|$



Filter input-to-delay stage TFs: DF2 - max gain

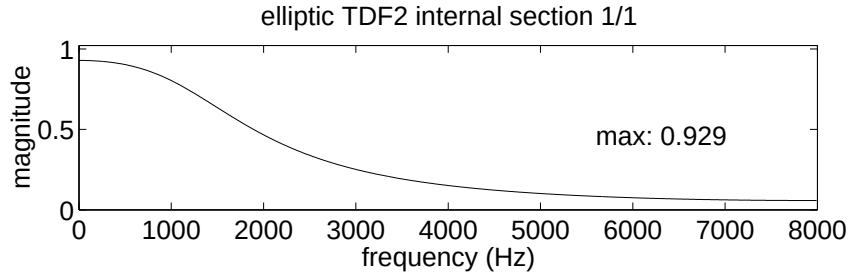


By nominally scaling the input by 4 we can avoid overflow in this realization. If a 12-bit converter is being used and a 16-bit word size, this is no great loss.

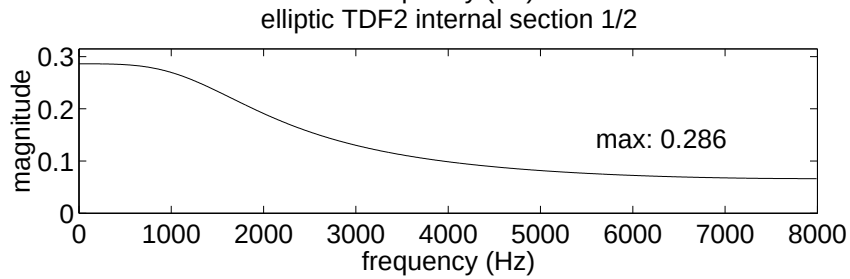
NB: this transfer function isn't the one used in lab.

Biquad input-to-delay stage TFs: TDF2

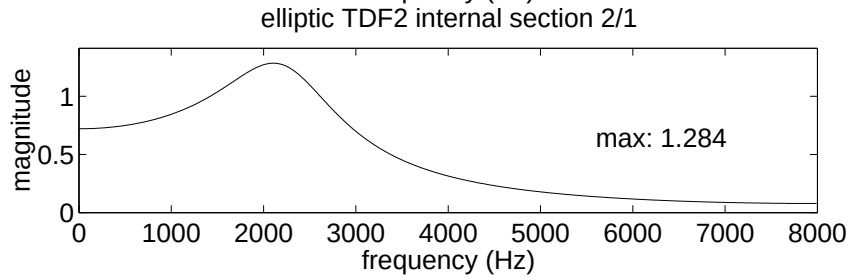
$|H_{11}(f)|$



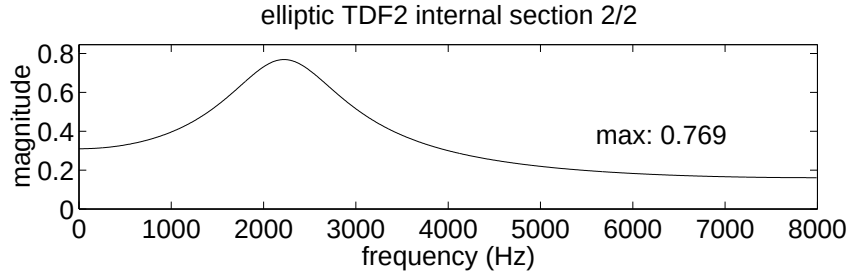
$|H_{12}(f)|$



$|H_{21}(f)|$

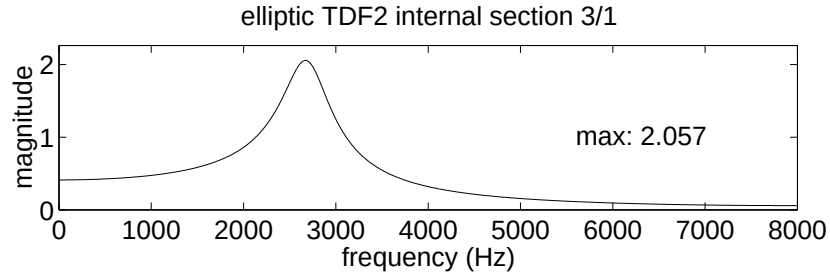


$|H_{22}(f)|$

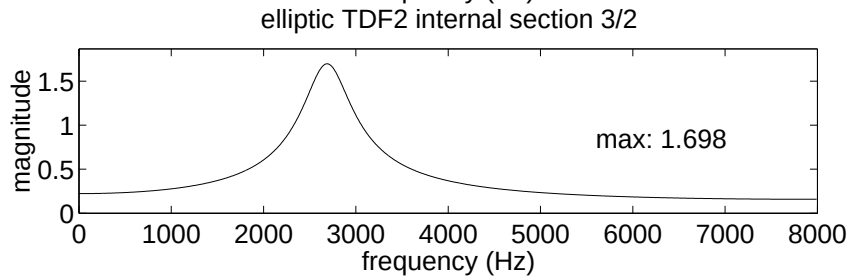


Biquad input-to-delay stage TFs: TDF2 (ctd)

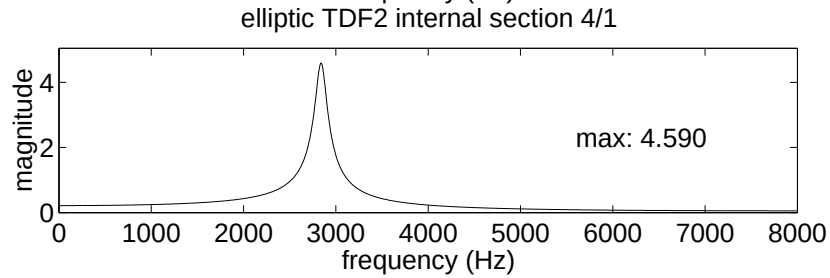
$|H_{31}(f)|$



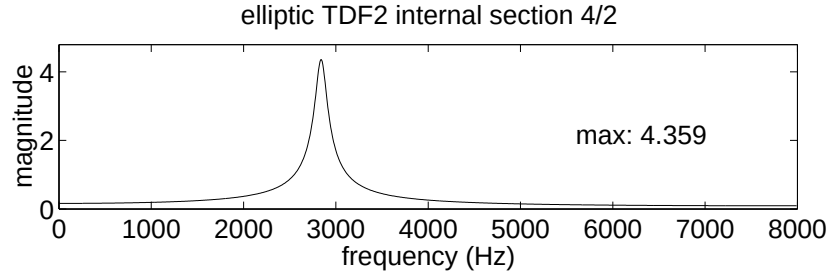
$|H_{32}(f)|$



$|H_{41}(f)|$

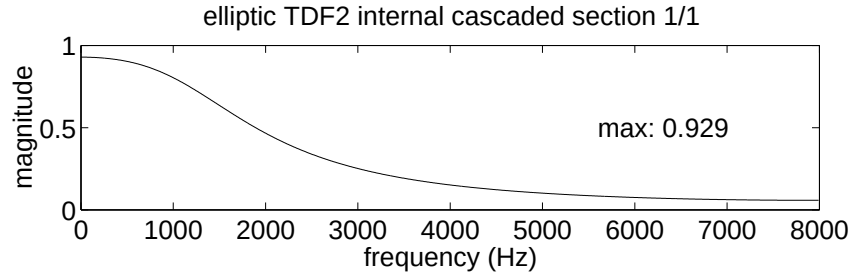


$|H_{42}(f)|$

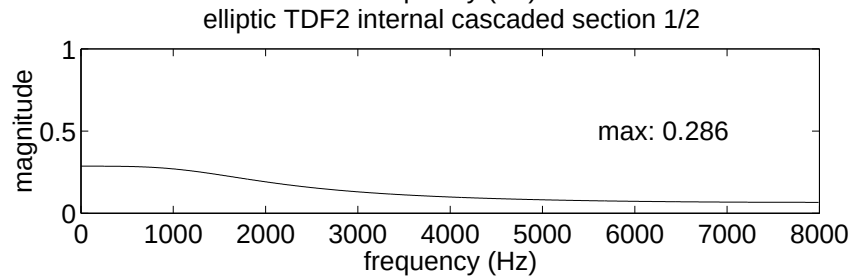


Filter input-to-delay stage TFs: TDF2

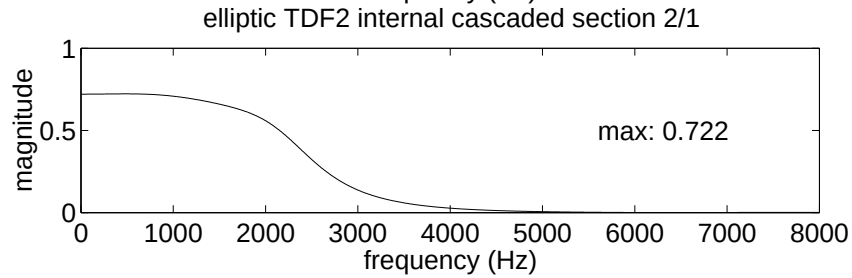
$|H_{11}(f)|$



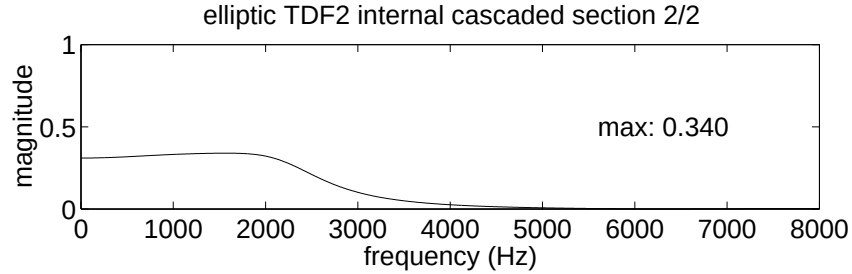
$|H_{12}(f)|$



$|H_1(f)H_{21}(f)|$

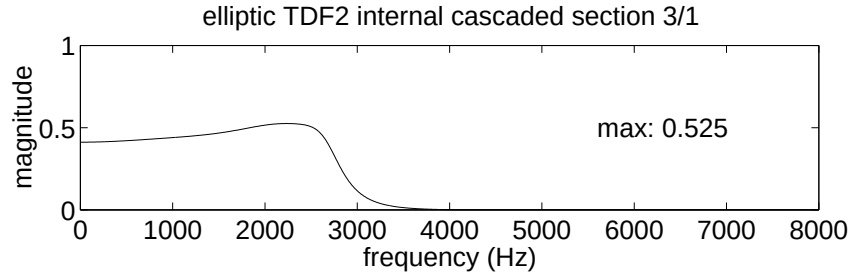


$|H_1(f)H_{22}(f)|$

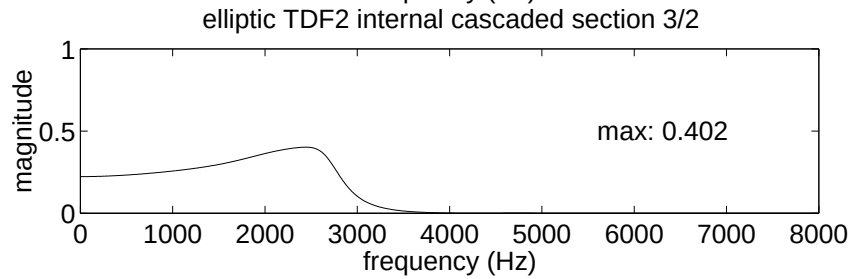


Filter input-to-delay stage TFs: TDF2 (ctd)

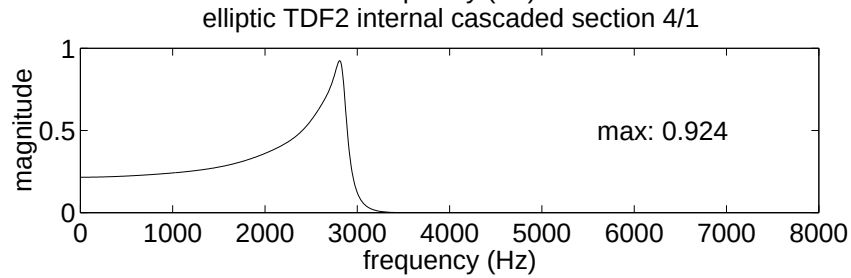
$$|H_1(f)H_2(f)H_{31}(f)|$$



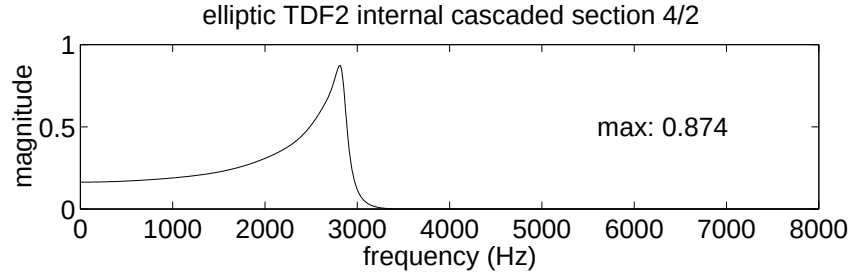
$$|H_1(f)H_2(f)H_{32}(f)|$$



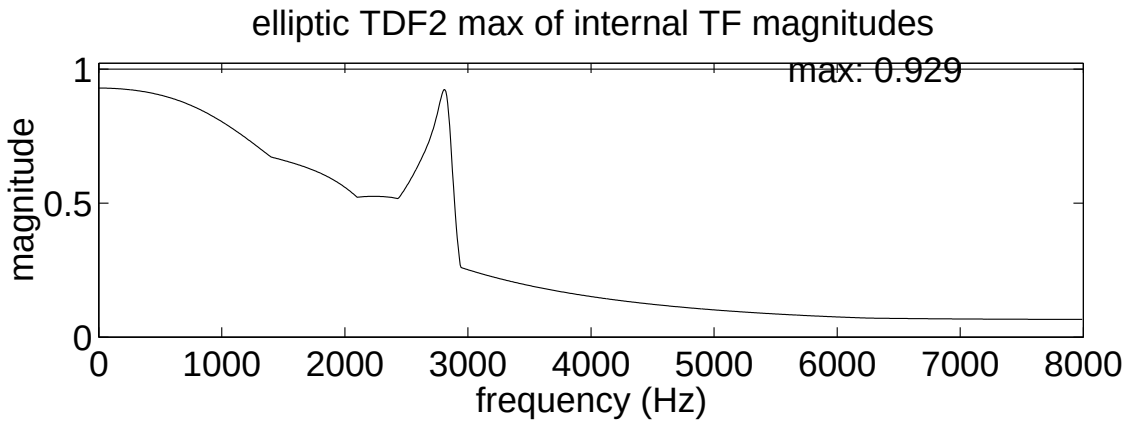
$$|H_1(f)H_2(f)H_3(f)H_{41}(f)|$$



$$|H_1(f)H_2(f)H_3(f)H_{42}(f)|$$



Filter input-to-delay stage TFs: TDF2 - max gain



Implementing a biquad cascade

The implementation steps are:

- Factor the transfer function into pole and zero pairs.
- Choose a biquad architecture.
- Relate the biquad coefficients to the chosen architecture coefficients.
- Order the poles and the zeros to control internal resonance levels.
- Distribute the gain between the biquad sections.
- Program and get to work.
- Test.

Overflow issues for biquad coefficients

Consider the biquad

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} = \frac{b_0 z^2 + b_1 z + b_2}{z^2 + a_1 z + a_2}.$$

The poles of $H(z)$ are determined by $z^2 + a_1 z + a_2$. Assume a complex valued pole pair, $p_1 = re^{j\theta}$ and $p_2 = re^{-j\theta}$.

$$(z - p_1)(z - p_2) = z^2 - 2r\cos(\theta)z + r^2 = z^2 + a_1 z + a_2.$$

In order for the filter to be (conditionally) stable the biquad poles have to be (on or) within the unit circle.

Because $0 \leq r \leq 1$ we have that $0 \leq a_2 \leq 1$ and $-2 < a_1 \leq 2$.
In addition, $a_2 \geq a_1^2/4$.

Overflow issues for biquad coefficients

We will be using Q15 numeric format values in the C5515 and DE2-70. The magnitude of the a_1 value can be greater than 1 (but less than 2). We need to worry about this.

There also may be scaling concerns with the b coefficient values as well. One needs to stay alert.

The b values have generally been well behaved. Occasionally there are b_1 values with magnitude greater than 1. Large b values can be handled by scaling all of the b coefficients. This affects only the gain through the system.

Scaling cannot be applied to the a values without changing the shape of the transfer function. (Recall $a_0 = 1$ requirement). Alternatives:

1. Implement each multiply $a_i x[n]$ as $((a_i/2)x[n])2$.
2. Use $Q(14)$ representation.

Coefficient scaling possibilities

If we divide the a's by k we need to multiply the sum by k.

Use Q14 data and Q14 coefficients?

Q14×Q14 gives Q28. To make Q28 into Q14 shift left 2 then truncate. To make Q29 into Q14 need to left shift 1 then truncate.

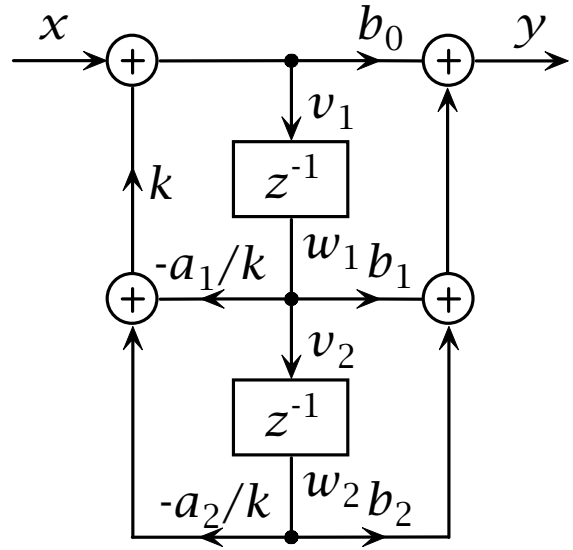
Use Q15 data and Q14 coefficients?

Q15×Q14 gives Q29. To make Q29 into Q15 shift left 2 and truncate.

Use Q15 data and Q15 coefficients?

Q15×Q15 gives Q30.To make Q30 into Q15 shift left 1 and truncate.

Need to remember to round before truncating.



Summary of what we covered today

- Roundoff and overflow for FIR filters
- Roundoff and overflow for IIR filters
- Finite precision IIR FIR filter design strategies

References

"Understanding digital signal processing," R. Lyons, 2006.

"Digital signal processing," Proakis and Manolakis, 3rd Edition.