IIR Filters

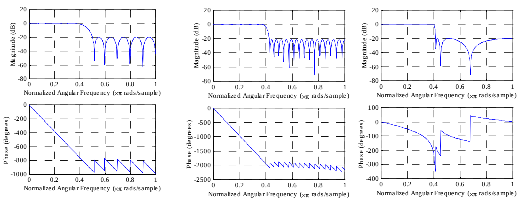
- Advantages of IIR filters
- Standard Filter shapes and z-domain Transformations
- Effect of Coefficient errors and use of Biquads
- Alternative Signal-flow Graphs
- C implementation and the #include directive
- Circular Buffer Implementation
- Summary

IIR Filters

$$H(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_N z^{-M}}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}}$$

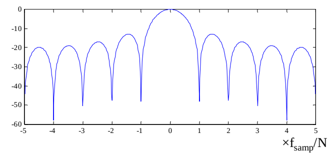
- Advantages
  - Fewer coefficients for sharp cutoff filters
  - Can calculate coefficients for standard filters
  - Can apply transformations to frequency axis
  - Can make all-pass filters (if $b_i = a_{N-i}$)
- Disadvantages
  - Non-linear phase response
  - Can be unstable: adaptive filters difficult, coefficient precision vital

Filter Sharpness



- 21[st] order FIR, 54[th] order FIR and 6[th] order IIR
- IIR filter has steepest cutoff but very non-linear phase
  - Linear phase ($\phi = -\omega\tau$) $\Leftrightarrow$ pure delay of τ $\Leftrightarrow$ Symmetrical FIR
  - The 180° phase jumps arise when the response changes sign

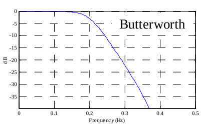
Sharpness & Impulse Response

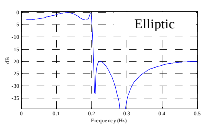


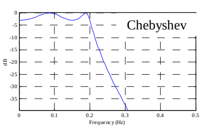
- Truncating the impulse response to N samples:
  - convolves the frequency response with a Dirichlet function (aliased sinc)
  - smears out sharp frequency-domain transitions
  - A step transition will now cover a frequency range of f_{samp} /N

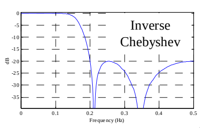
Standard Filters

- Standard filters have specifiable ripple in passband and/or stopband:



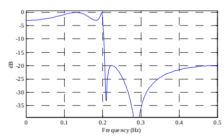


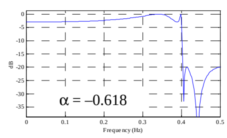




Frequency Transformations

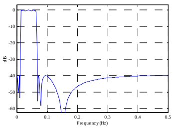
- If we replace z by $\dfrac{z-\alpha}{1-\alpha z}$ we apply a non-linear frequency transformation.
- Choose $\alpha = \dfrac{\sin((f_1 - f_2)T\pi)}{\sin((f_1 + f_2)T\pi)}$ to map f_1 to f_2



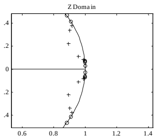


- Similar transformations map lowpass to highpass, bandpass or bandstop

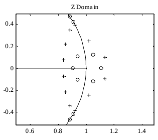
Coefficient Precision



12th order bandpass filter



Pole/Zero positions
52 bit coefficients



Pole/Zero positions
33 bit coefficients

- Reducing coefficient precision to 33 bits (10 significant digits) causes severe instability + wrong filter shape.
- Required precision increases rapidly for high order filters.
- Solution: factorize filter transfer function into 2nd order factors.

Filter Implementation

Transfer Function: $$H(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_N z^{-M}}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}}$$

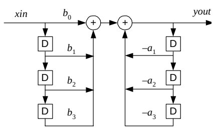
Recurrence Relation: $$y(n) = b_0 x(n) + b_1 x(n-1) + \cdots + b_N x(n-M) \\ -a_1 y(n-1) - \cdots - a_N y(n-N)$$

- Note the sign change of denominator coefficients
- We can factorize H(z) and implement as cascaded filters:

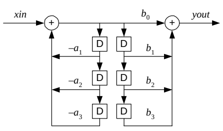
$$H(z) = b_0 + b_1 z^{-1} + \cdots + b_N z^{-M} \times \frac{1}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}}$$

Direct Form I and II

- Direct Form I
  - FIR followed by IIR
  - D represents a one-sample delay (multiplication by z^{-1})
  - We can merge the adders.

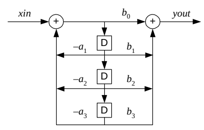


- Direct Form II
  - IIR followed by FIR
  - Only needs a single delay line

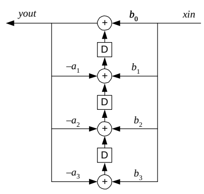


Transpose Networks

- The transfer function is unchanged if you:
  - Reverse the flow in each branch
  - Interchange branch divisions and branch summations
- Coding can be slightly more efficient



Direct Form II

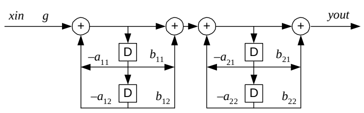


Direct Form II Transposed

Biquad Implementation

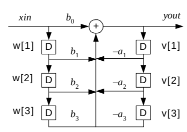
$$H(z) = g \times \frac{1 + b_{11} z^{-1} + b_{12} z^{-2}}{1 + a_{11} z^{-1} + a_{12} z^{-2}} \times \cdots \times \frac{1 + b_{K1} z^{-1} + b_{K2} z^{-2}}{1 + a_{K1} z^{-1} + a_{K2} z^{-2}}$$

- We can always factorise the numerator and denominator of *H(z)* into quadratic factors with real coefficients.
- Each complex pole-pair or zero-pair forms a single factor.
- Group nearby pole and zero pairs to form 2nd-order sections known as biquads.
- 2nd-order filters are much less sensitive to coefficient errors.



C Code for IIR Filters

```
yout = b[0]*xin;
for (k=order; k>0; k--) {
      yout += w[k]*b[k]-v[k]*a[k];
      w[k] = w[k-1];  v[k] = v[k-1];
      }
w[1]=xin; v[1]=yout;
```



- Software implementation of Direct Form I
- The diagram illustrates the case: `order = 3`
- We access the delay lines in reverse order so that we can update their values in the second line of the loop.
- `w[1]` and `v[1]` are updated with junk in the final iteration, then overwritten with their correct values outside the loop.

05/06/2001 IIR Filters

Coefficient File

```
float a[ ] = {     1,     -1.76,    1.1829,    -0.2781,};
float b[ ] = {  0.0181,    0.0543,    0.0543,    0.0181,};
```

- Get MATLAB to create a file: `coef.txt`
  - See the `fopen()`, `fprintf()` and `fclose()` commands
- The comma after the last value in each array is optional
- The length of each array is determined by the number of values specified.
- In C, arrays go from `a[0]` … `a[order]` whereas in MATLAB, they go from `a(1)` … `a(order+1)`.

13

05/06/2001 IIR Filters

Initialisation

```
int order;
float *w, *v;
#include "coef.txt"

order=sizeof(a)/sizeof(a[0]) - 1;
w = (float *) calloc(order+1, sizeof(float));
v = (float *) calloc(order+1, sizeof(float));
```

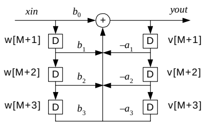
- To get the number of values the `a[ ]` array, we divided the length (in bytes) of the whole array by the length of each element.
- The filter order is one less than the number of `a[ ]` values.
- Array names like `w` and `v` are pointers (i.e. they just store a memory address).
- Use `calloc()` to reserve an area of memory, initialize its contents to `0` and to set `w` to its starting address.
- Our code needs the length of `w` and `v` to be one greater than the order.

14

05/06/2001 IIR Filters

Circular Buffer

```
yout = b[0]*xin;
for (k=1; k<=order; k++) {
        if (++m >= order) m=0;
        yout += w[m]*b[k]-v[m]*a[k];
        }
w[m]=xin; v[m--]=yout;
```



- In the diagram, `M` is the value of `m` as we enter the loop.
- With `++m`, we increment `m` each time through the loop and ensure that it never exceeds `order-1`.
- The final line overwrites the oldest values in the delay line and then decrements `m` so that they will act as `w[M+1]` and `v[M+1]` next time.
- The `w` and `v` arrays only need to contain `order` elements.

15

05/06/2001 IIR Filters

Circular Buffer

M	k=1	k=2	k=2
0	1	2	0
–1	0	1	2
1	2	0	1
0	…	…	…

- The table shows the value of `m` used in each loop iteration.
- `M` is the starting value as we initially enter the loop.
- The value of `m` from the final iteration determines the storage location: `w[m]=xin` and `v[m]=yout`.
- We then decrement `m` to provide `M` for the next sample

16

05/06/2001 IIR Filters

Summary

- IIR filters
  - Need fewer coefficients for sharp cut-off filters
  - Poles and zeros of standard filters can be calculated directly
  - z-domain transformations can change the cutoff-frequency and transform lowpass into highpass or, with a doubling of the order, into pandpass or bandstop.
- Alternative Signal-flow Graphs
  - Important differences in numerical performance when fixed point arithmetic is used - less important with floating point.
  - Factorize into biquads to avoid coefficient precision problems.
- C implementation
  - Use `#include` directive to insert a coefficient file into program.
  - Can use circular buffer to implement an efficient delay line.

17