



Module IN3031 / INM378

Digital Signal Processing

and Audio Programming

Tillman Weyde
t.e.veyde@city.ac.uk



Digital Filtering (moving from theory to applications)



RECAP: Convolution

- **Convolution** combines two signals, **similarly to cross-correlation**
 - it's the correlation with a reversed signal

$$\text{conv}(s1, s2)[t1] = \sum_{t=0}^{N2-1} s1[t1-t]s2[t]$$

$N2$ is the length of $s2$, $s1[i] = 0$ assumed where $i < 0$ or $i \geq N$

- Often **written** as **$s1 * s2$**



RECAP: Convolution Theorem

- The **most important property** of the convolution is given by the convolution theorem:
A convolution in the time domain is equivalent to a multiplication in the frequency domain:

$$x * y \leftrightarrow X \cdot Y$$

$$\text{meaning: } FT(\text{conv}(x, y)) = FT(x) \cdot FT(y)$$



Digital Filters

- Sound **spectra** are **changed** by **filters**
- **SFFT** manipulation and resynthesis - a form of **filtering in the frequency domain**
- **Most filtering** happens in the **time domain** by **convolution**



Linear Filters

- Linear filters **sum scaled and delayed copies** of the signal to itself (**convolution** with the **scaling factors**)
- **2 types**, depending on where they take the signal from
 - **Finite Impulse Response (FIR)** filters
(use input signal)
 - **Infinite Impulse Response (IIR)** filters
(use input & output signal)



The Order of Filters

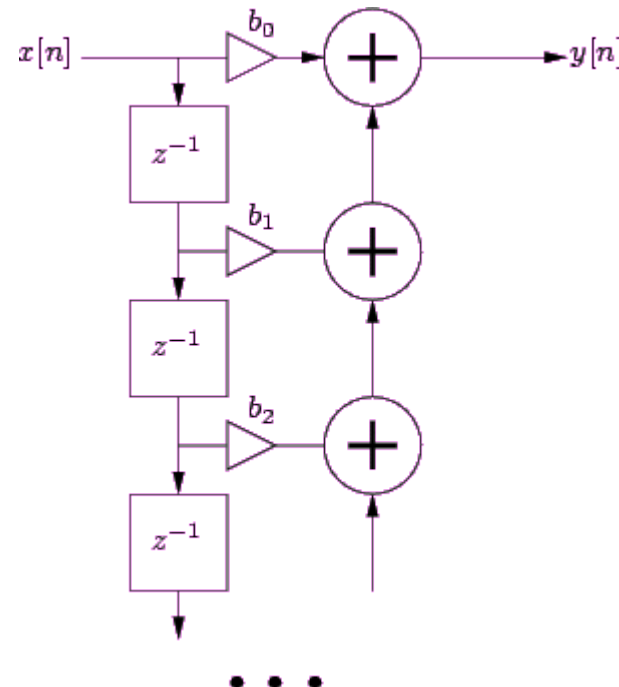
- An **FIR** filter ***f*** of **order *k*** has this **structure**
$$f(x[n]) = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] + \dots + b_k x[n-k]$$
- An **IIR** filter ***g*** of **order *k*** has this **recursive** structure
$$g(x[n]) = + b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] + \dots + b_k x[n-k] \\ - a_1 g(x[n-1]) - a_2 g(x[n-2]) - \dots - a_k g(x[n-k])$$
- or as a **difference equation**
$$y[n] = - \sum_{i=1}^k a_i y[n-i] + \sum_{i=0}^k b_i x[n-i]$$
- **a_n** and **b_n** are called **filter coefficients**

An FIR Filter

- An **FIR** filter f of order k has this **structure**

$$f(x[n]) = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] + \dots + b_k x[n-k]$$
 with **coefficients** $b = [b_0, b_1, b_2, \dots, b_k]$

- **Graphically:**



▷ : multiplication with
a scalar

z^{-1} : delay by 1 sample



Uses of Digital Filters

- Digital filters are **used for**
 - **anti-aliasing** (before downsampling)
 - **equalisation** (removing frequency imbalances of microphones, room acoustics etc)
 - user **sound modification** (adjust to personal taste)
 - sound **analysis** (select the frequency range to analyse)
 - sound **synthesis** (shape the timbre of a synthetic sound)



Types of Digital Filters

- There are four **common types** of filters:
 - **low pass** (anti-aliasing, synthesis, HF noise removal)
 - **high pass** (remove rumbling, protect speakers)
 - **band pass** (sound analysis)
 - **band stop** (removing unwanted signal, e.g. from power supply)
- **Other** types of filters:
 - **comb** filters (usually the result of short delays)
 - **all pass** filters (modify only the phase)



Properties of Digital Filters

- Filter **architecture** (FIR or IIR)
- Filter **order** ($\text{\#sample delays} = \text{\#coefficients} - 1$)
- Filter **coefficients** (the values defining the filter operation)

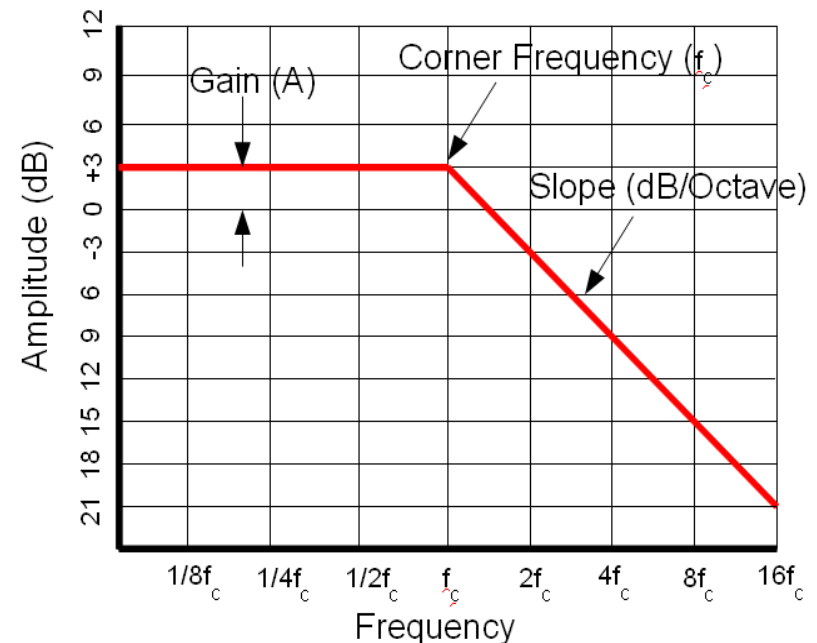
resulting from these

- **Frequency response** (mainly magnitude)
- **Impulse response** (sometimes step response)
- **Time behaviour** (phase response, group delay)



Filter Parameters

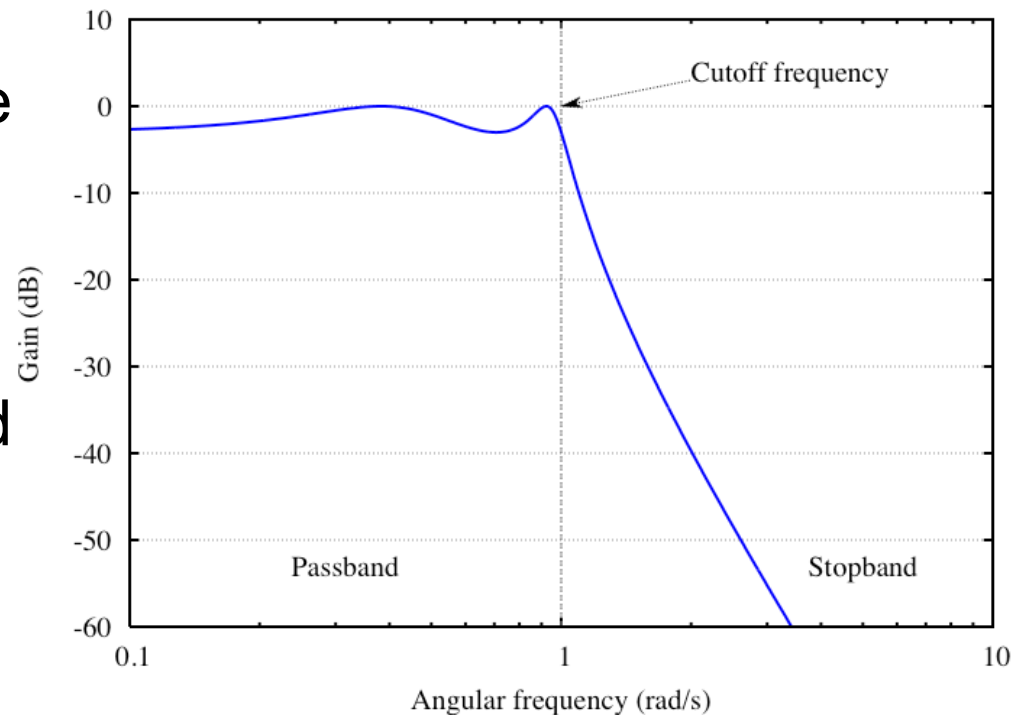
- The **pass band** is a range of frequencies, that should pass the filter unattenuated.
- The **stop band** is a range of frequencies, that should not pass the filter.
- The pass band ends at the **corner frequency** (usually at -3dB).
- The **slope** of the freq. resp. is measured in dB/octave (sometimes decade)





More Filter Parameters

- **Ripple** is the unevenness of the frequency response
- **Resonance** is a peak in frequency response near cut-off frequency
- **Stability**: The filter should (usually) not oscillate by itself





FIR Filter Design

FIR:

- Approach: coefficients as **iFFT** of frequency response
- **Pro:**
 - **FIR** filters are **always stable**
 - Good phase behaviour
- **Cons:**
 - for steep slope in the transition band, we need high **number of coefficients** (and thus computation time)



FIR Filter Design in Matlab

Filter Design & Analysis Tool - [untitled.fda *]

File Edit Analysis Targets View Window Help

Current Filter Information:

- Structure: Direct-Form FIR
- Order: 50
- Sections: 1
- Stable: Yes
- Source: Designed

Filter Specifications:

Response Type:

- ☒ Lowpass
- ☐ Highpass
- ☐ Bandpass
- ☐ Bandstop
- ☐ Differentiator

Design Method:

- ☐ IIR Butterworth
- ☒ FIR Equiripple

Filter Order:

- ☒ Specify order: 64
- ☐ Minimum order

Options:

Density Factor: 16

Frequency Specifications:

Units: Hz

Fs: 44100

Fpass: 4000

Fstop: 5000

Magnitude Specifications:

Enter a weight value for each band below.

Wpass: 1

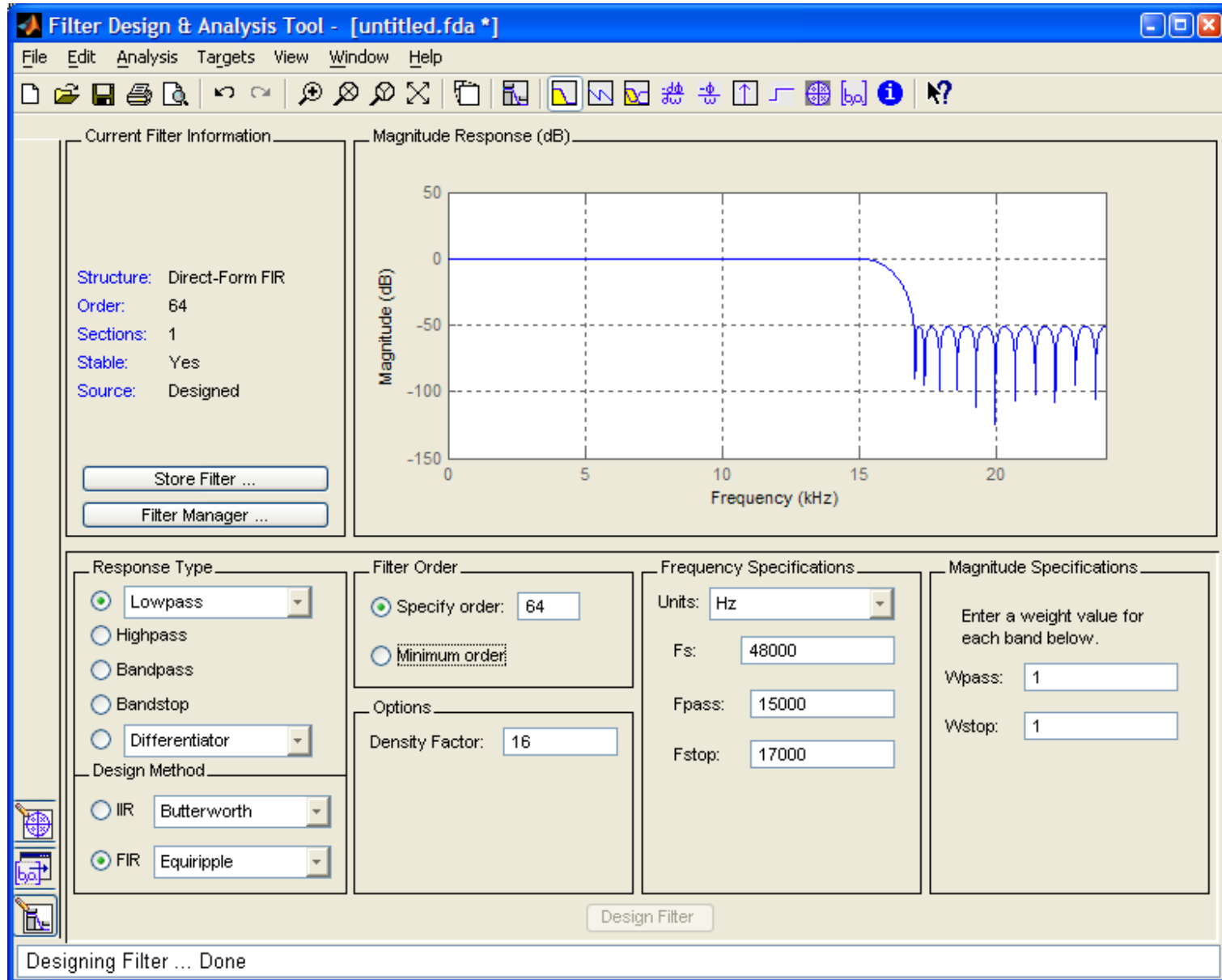
Wstop: 1

Design Filter

Ready

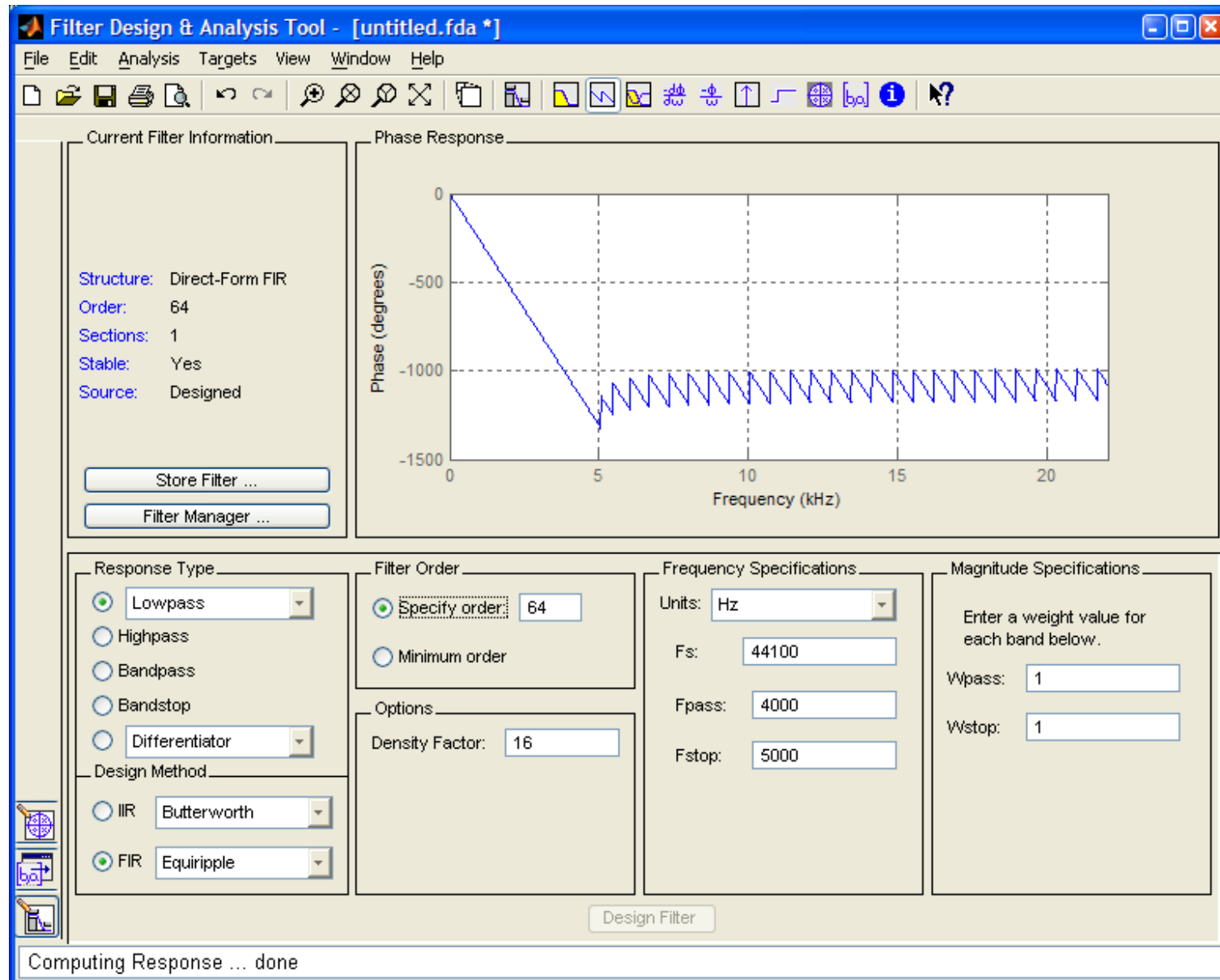


Frequency Response





Phase Response





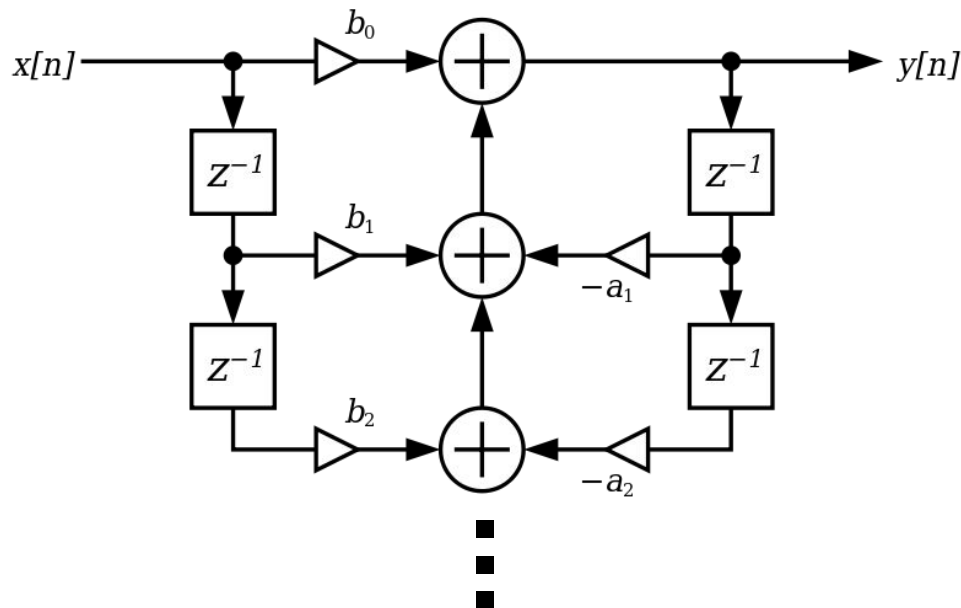
IIR Filter Design

IIR:

- Approach: Define **numerical methods** for finding appropriate filter coefficients (mathematically demanding).
- Pro:
 - IIR filters can be **very efficient**.
- Cons:
 - Uncontrolled **phase** behaviour
 - **May be unstable**
 - Quantisation noise may multiply through recursion

IIR Filter Diagram

$$y[n] = -\sum_{i=1}^k a_i y[n-i] + \sum_{i=0}^k b_i x[n-i]$$



z^{-1} represents a delay by one sample

This structure is called *Direct Form 1*

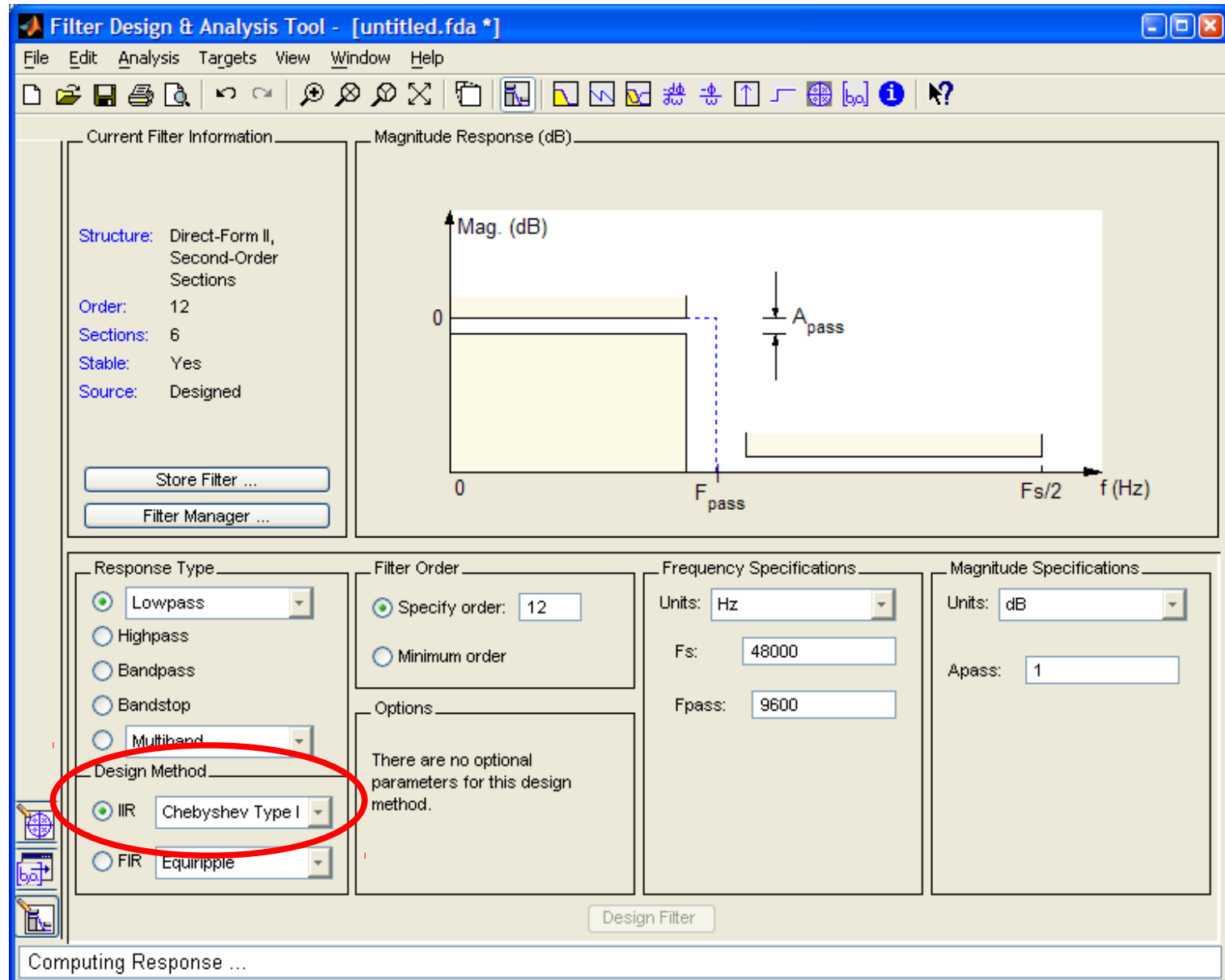


The Impulse Response of an IIR Filter

- An IIR filter g
$$g(x[n]) = -a_1 g(x[n-1]) - a_2 g(x[n-2]) - \dots - a_k g(x[n-k])$$
$$+ b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] + \dots + b_k x[n-k]$$
- The impulse response of g has to be computed recursively and may be infinitely long
- IIR filters
 - allow very effective filtering with few coefficients
 - may oscillate by themselves
 - frequency response is hard to compute

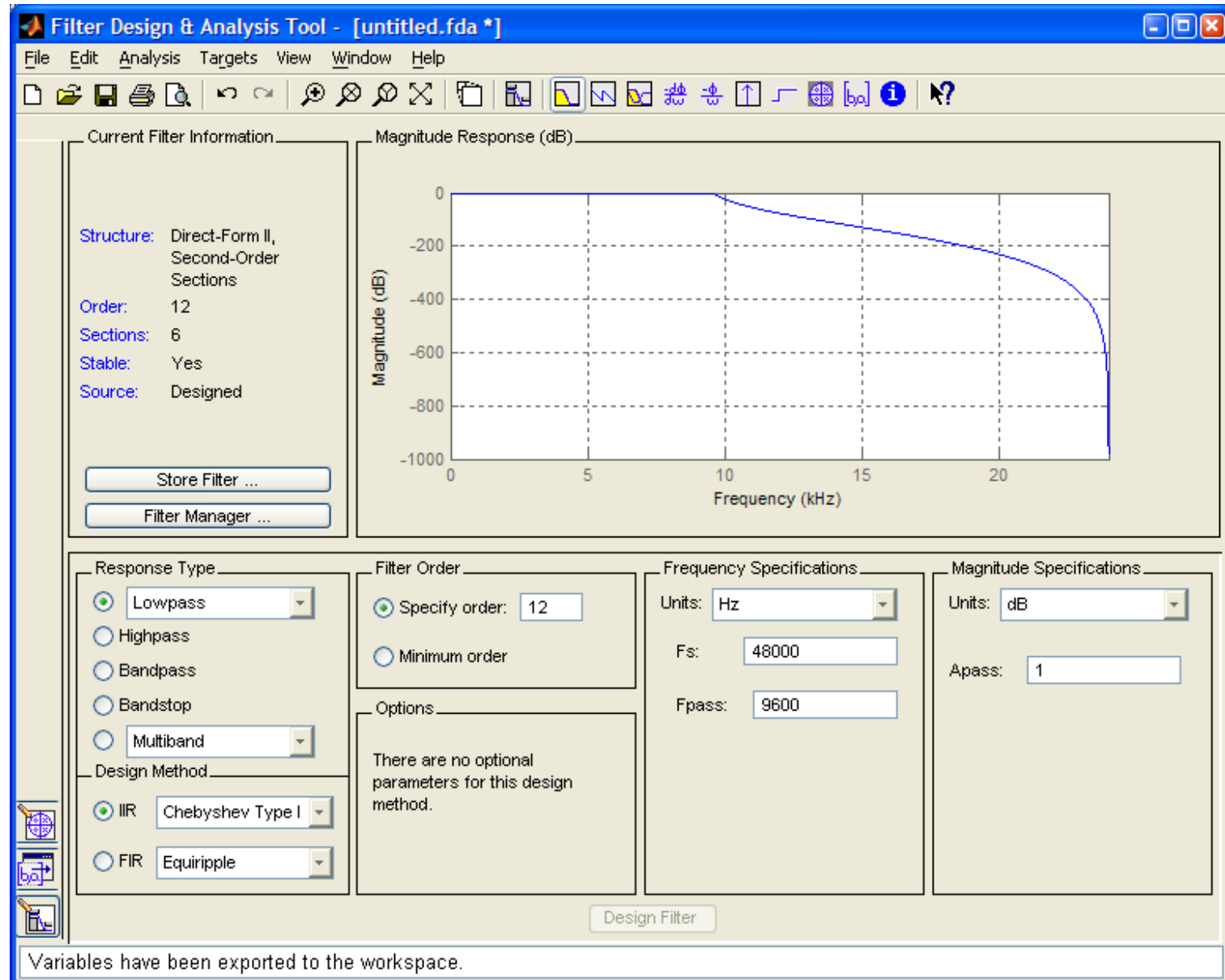


IIR Filter Design in Matlab



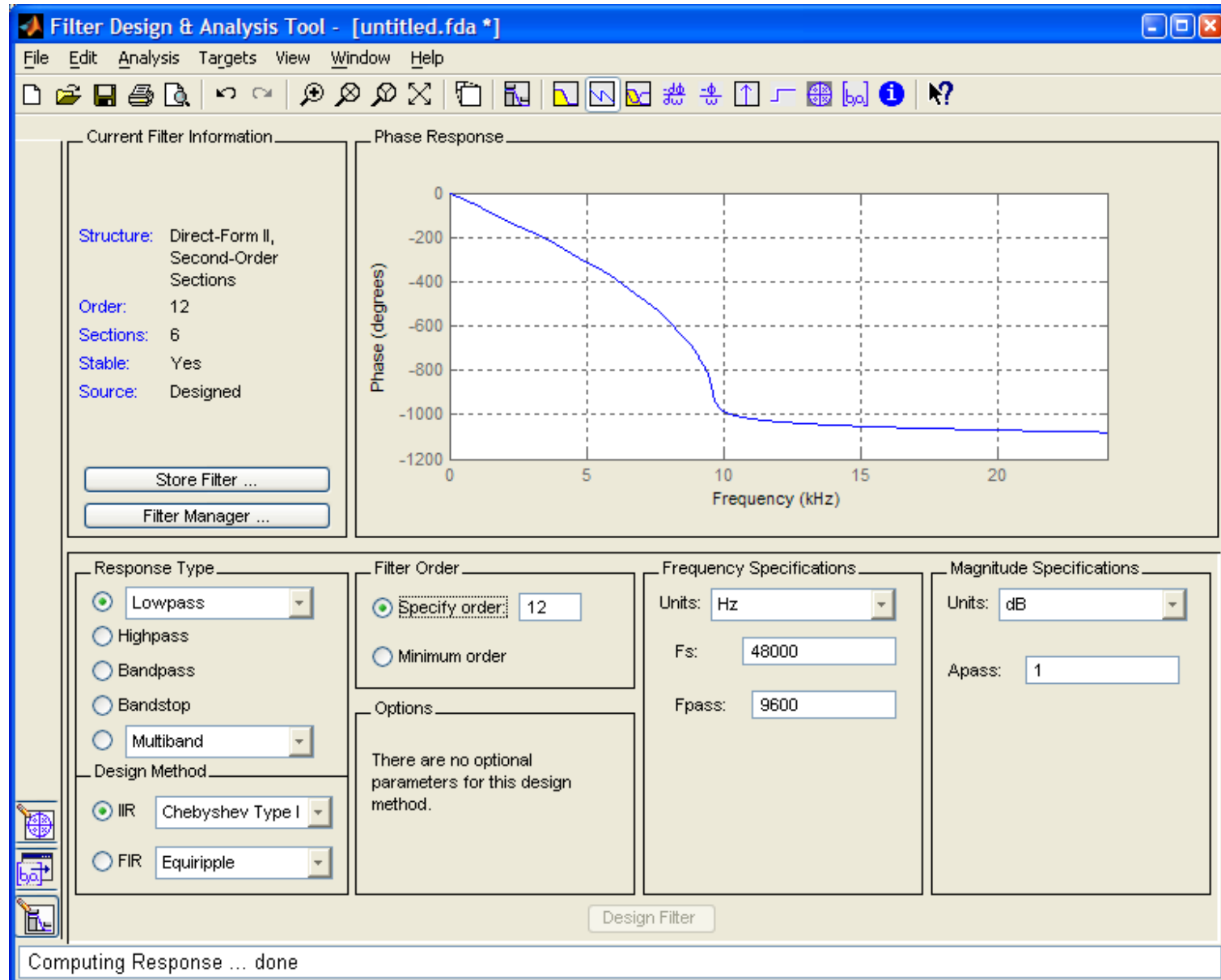


Frequency Response



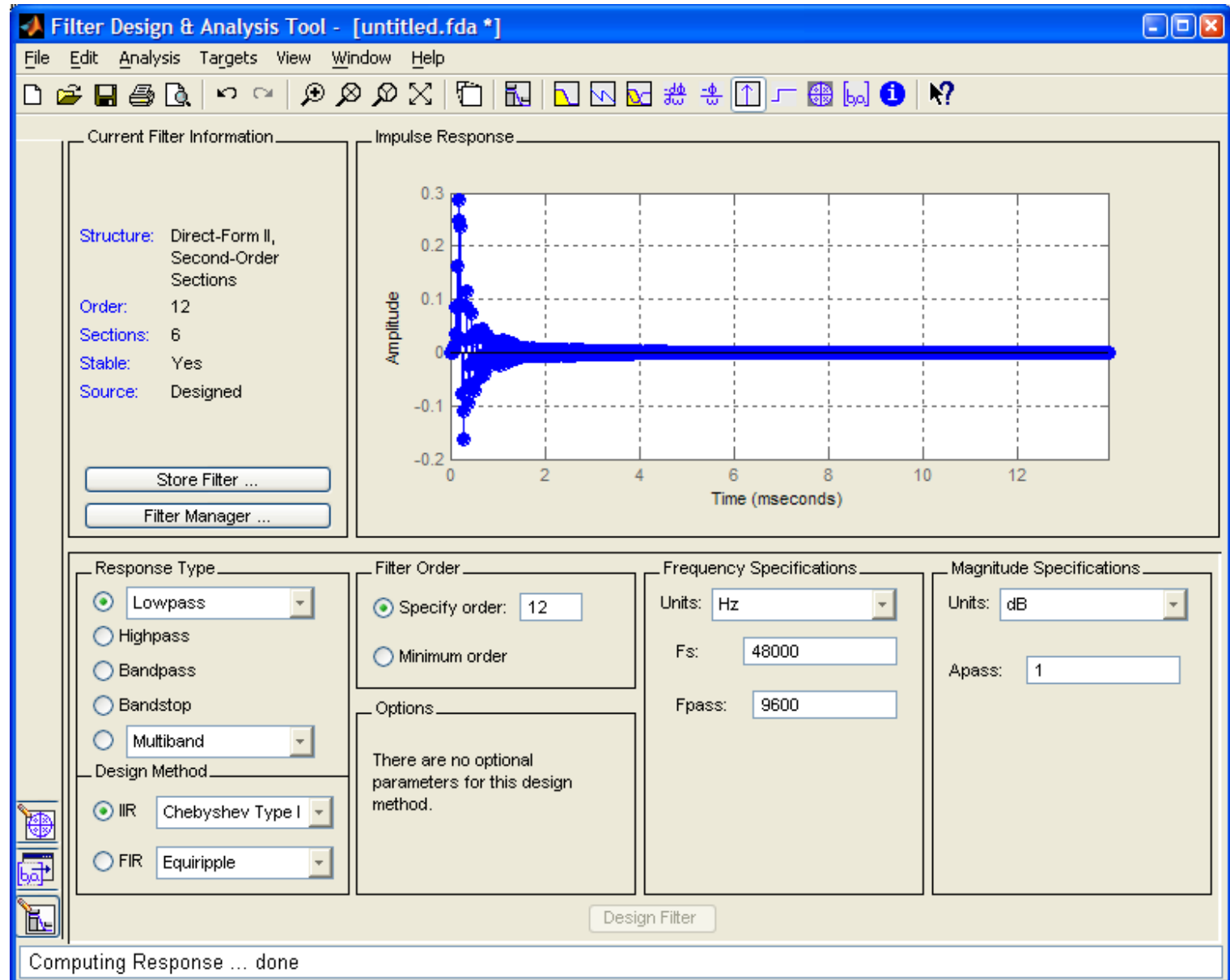


Phase Response



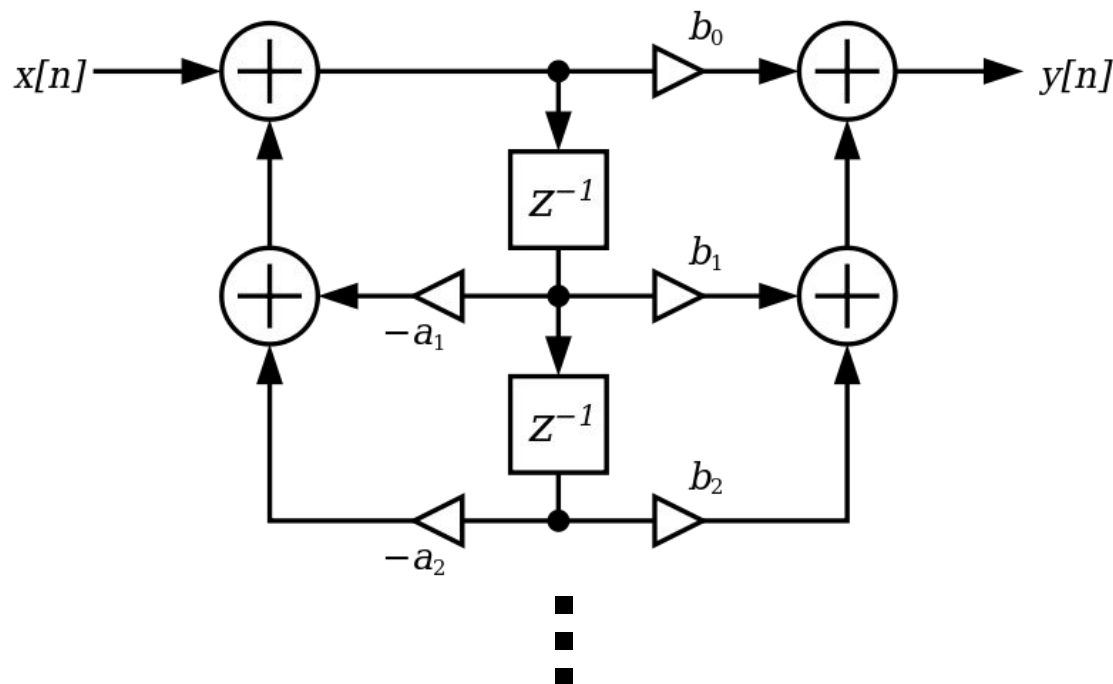


Impulse Response



IRR Filters in Practice

- IRR filters are used in *Direct Form 2*
- Equivalent to Direct Form 1, but more efficient





IRR Filters in Practice (2)

- Many architectures exist
 - equivalent possibilities (linear systems ...)
 - numerical and computational trade-offs
- Standard DF2 uses SOS (internal) and G (scaling) coefficients
- Can be transformed to A and B coefficients in
 $[B, A] = \text{sos2tf}(SOS, G)$
- Apply A and B coefficients with
`filter(A, B, sig)`



Exporting an IIR filter to the Matlab workspace

The screenshot shows the 'Filter Designer' application window with the title 'Filter Designer - [untitled.fda *]'. The 'Export' dialog box is open, showing the following settings:

- Export To:** Workspace
- Export As:** Coefficients
- Variable Names:**
 - SOS Matrix: SOS
 - Scale Values: G
 - ☐ Overwrite Variables

The background window displays the 'Current Filter Information' panel with the following details:

- Structure:** Direct-Form II, Second-Order Sections
- Order:** 4
- Sections:** 2
- Stable:** Yes
- Source:** Designed

Buttons for 'Store Filter ...' and 'Filter Manager ...' are visible. The 'Response Type' section shows 'Lowpass' selected. The 'Design Method' section shows 'IIR' selected with 'Butterworth' as the filter type. The 'Filter Order' section shows 'Minimum order' selected. The 'Frequency Specifications' section shows 'Units: Hz', 'Fs: 100', 'Fpass: 4', and 'Fstop: 20'. The 'Magnitude Specifications' section shows 'Units: dB', 'Apass: 1', and 'Astop: 50'. A 'Design Filter' button is at the bottom right. A plot of the filter's magnitude response is visible on the right side of the window.

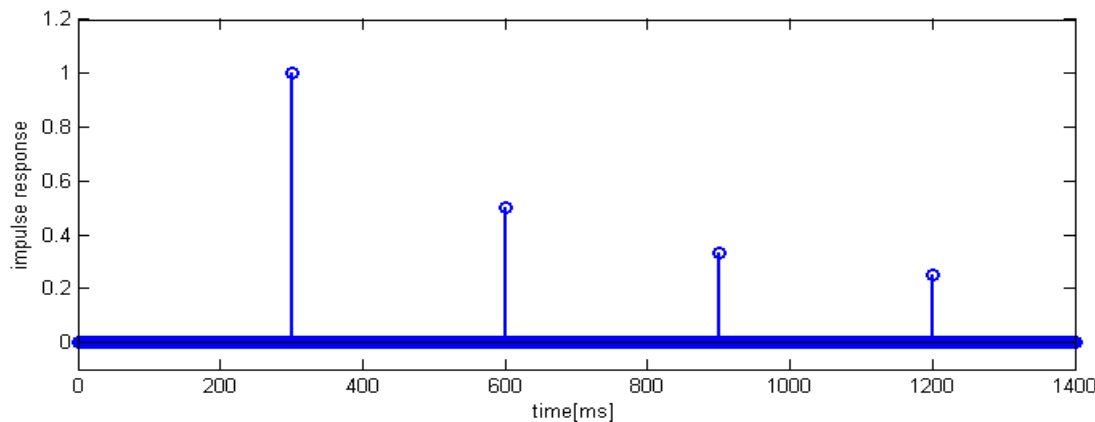


Impulse Responses and Audio Effects

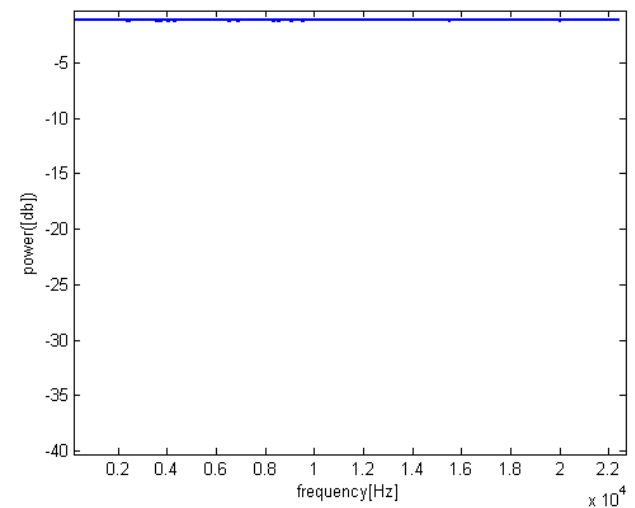


Echo

- Impulse response: Few filter coefficients **span over seconds**
- Frequency response is **flat**



Impulse Response

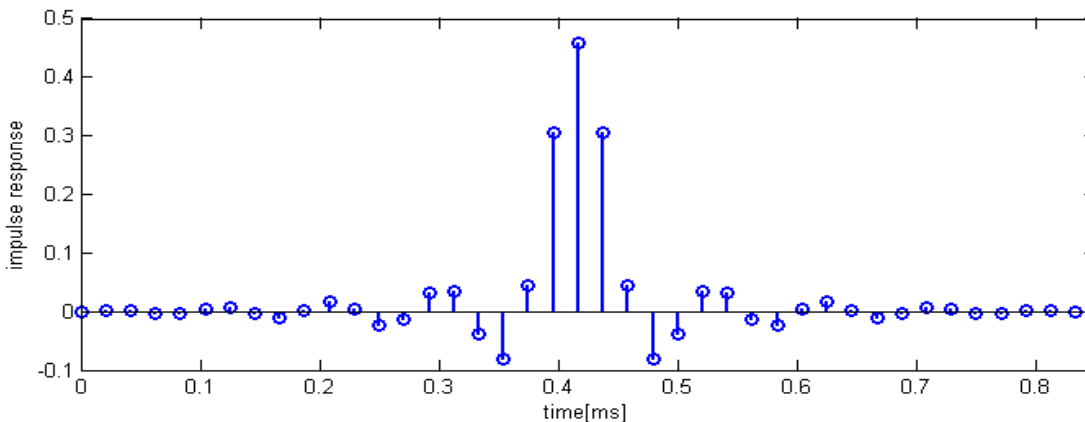


Frequency Domain

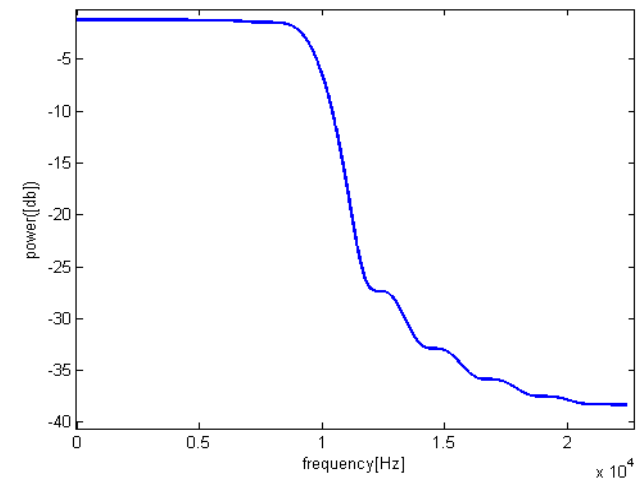


Low-Pass

- Impulse response: Many filter coefficients in the **first few milliseconds**
- **Approximates a rectangular window** in frequency domain



Impulse Response

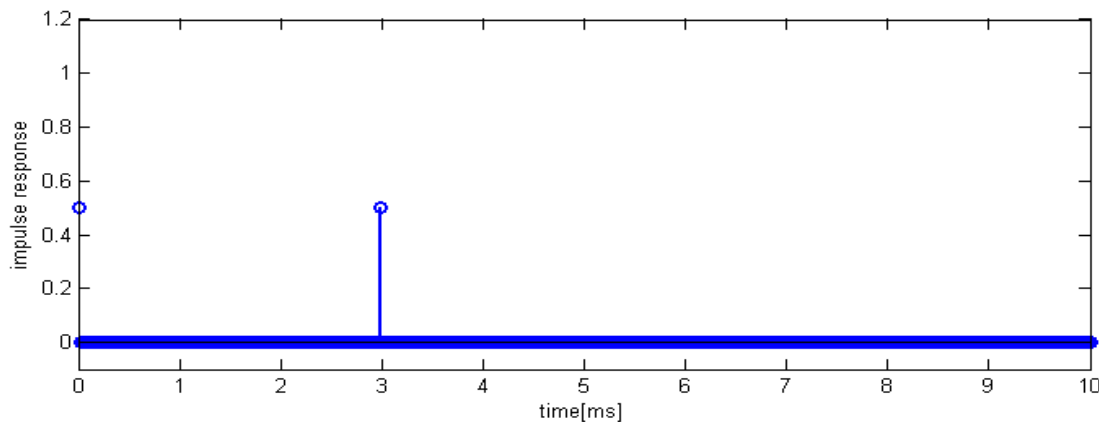


Frequency Domain

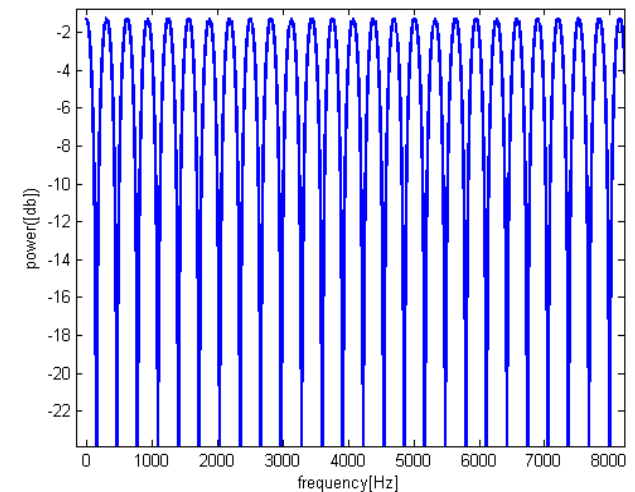


Flanger as Filter

- Impulse response: 2 filter coefficients in the **first few milliseconds (identity + delay)**
- **Comb filter shape** in frequency domain
- IR changes over time



Impulse Response



Frequency Domain

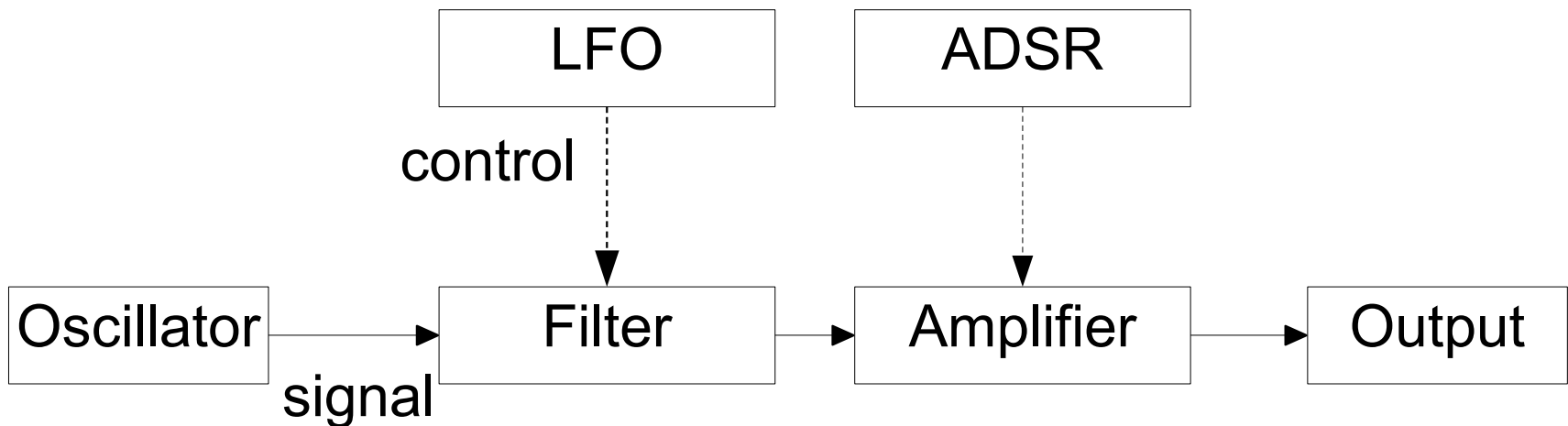


Using Filters for Subtractive Synthesis



Subtractive Sound Synthesis

- Most common form of analogue synthesis
- Generates a sound, filters and amplifies (attenuates) it
- Exemplary set-up:





“Virtual” Synthesizer





Take-Home Messages

- FIR and IIR filters are often used to remove frequency bands (Hi-Pass, Lo-Pass, ...)
- Filters need delay-lines, i.e. memory buffers
- Filters can be time-variant (flanger)
- Can be used in subtractive synthesis
- Games need real-time programming
- Complexity often hidden by building blocks (FMOD)



Reading:
FMOD Studio API Docs
Smith, DSP Guide, chpt 15