Sample Code

# Applying biquadratic filters to a music loop

Change the frequency response of an audio signal using a cascaded biquadratic filter.

Download

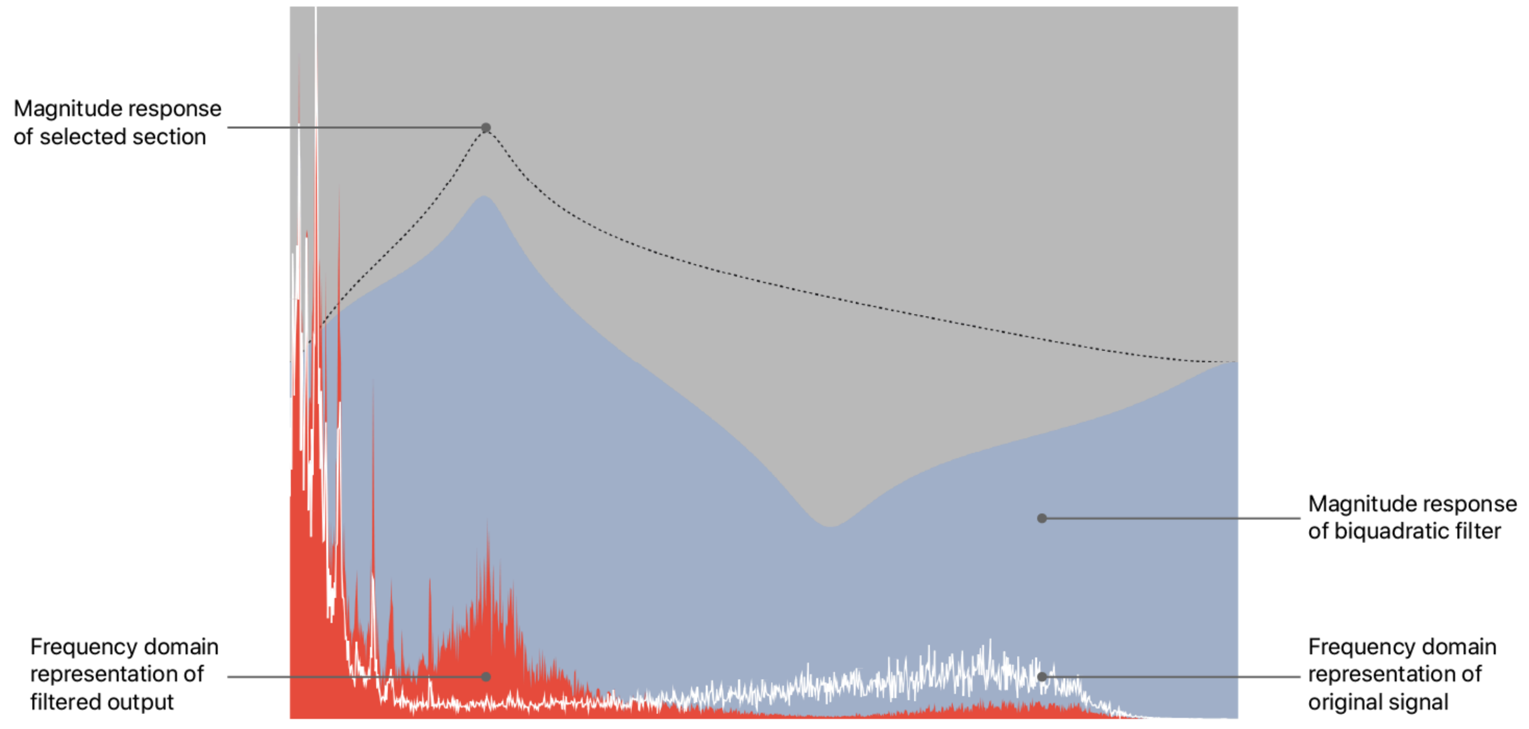
macOS 13.0+ | Xcode 14.3+

## Overview

You can shape the output of an audio signal, such as by boosting or cutting the bass or treble of a music track, with the single-channel and multichannel biquadratic filters that the vDSP library provides.

The vDSP library defines a biquadratic filter from a set of five coefficients for each section. This sample code app calculates those coefficients from a set of properties that it displays as controls in the user interface. The user interface provides controls to select the filter type (such as low-pass or high-pass), the center frequency of the filter, and the Q value that controls the width of the filter's frequency band.

The sample code app displays the magnitude response of the selected section, the magnitude response of the entire filter, the frequency-domain representation of the input signal, and the frequency-domain representation of the filtered, output signal.



Before exploring the code, try building and running the app to familiarize yourself with the effect of the different parameters on the music loop.

## Initialize the biquadratic filter

The `biquadSectionCount` constant defines the number of sections that the biquadratic filter implements. The sample code app sets this to 3 by default.

```
public let biquadSectionCount = vDSP_Length(3)
```

The `vDSP_biquad_CreateSetup` function returns a new biquadratic filter structure that contains `biquadSectionCount` sections. The sample code app defines the coefficients to produce a filter that returns an output that's identical to the input.

```
let coefficients = [[Float]](
    repeating: BiquadCoefficientCalculator.passthroughCoefficients.array,
    count: Int(biquadSectionCount)).flatMap {
        $0
}

biquadSetup = vDSP_biquad_CreateSetup(vDSP.floatToDouble(coefficients),
                                      biquadSectionCount)!
```

## Define the biquadratic coefficients

Five coefficients define each section of a biquadratic filter. The following formula describes the underlying math of the biquadratic filter, with *z* referring to the complex frequency-domain representation of the signal:

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

The sample code app includes the `BiquadCoefficientCalculator` structure that provides the `static BiquadCoefficientCalculator.coefficients(for:sampleRate:)` function. This function returns the five coefficients for a filter type, center frequency, Q, and sample rate.

Each filter type uses the same shared values.

```
let omega = 2.0 * .pi * frequency / Float(sampleRate)
let sinOmega = sin(omega)
let alpha = sinOmega / (2 * Q)
let cosOmega = cos(omega)
```

A `switch` statement calculates the coefficients for each filter type case. For example, the following code calculates the coefficients for a low-pass filter (that reduces high frequencies):

```
case .lowpass:
    b0 = (1 - cosOmega) / 2
    b1 = 1 - cosOmega
    b2 = (1 - cosOmega) / 2
    a0 = 1 + alpha
    a1 = -2 * cosOmega
    a2 = 1 - alpha
```

## Set the biquadratic coefficients

The `vDSP_biquad_SetCoefficientsSingle` function sets the coefficients for a section of the biquadratic filter. The sample code app defines `selectedSectionIndex` as the index of the currently selected section, and the `BiquadCoefficientCalculator.SectionCoefficients` structure provides an `array` variable that returns `[b0, b1, b2, a1, a2]`.

```
vDSP_biquad_SetCoefficientsSingle(biquadSetup,
                                  selectedSectionCoefficients.array,
                                  vDSP_Length(selectedSectionIndex),
                                  1)
```

## Apply the biquadratic filter to the audio sample

The vDSP_biquad function applies the biquadratic filter to a page of input samples and writes the result to the `outputSignal` array. The `delay` array contains the past state for each section of the biquadratic filter.

```
vDSP_biquad(biquadSetup,
            &delay,
            page, 1,
            &outputSignal, 1,
            vDSP_Length(Self.sampleCount))
```

On return, `outputSignal` contains the filtered version of `page`, and `delay` contains the final state data of the filter. The sample code app passes `delay` to the next call of vDSP_biquad.

## See Also

### Audio Processing

{} Visualizing sound as an audio spectrogram

Share image data between vDSP and vImage to visualize audio that a device microphone captures.

{} Equalizing audio with discrete cosine transforms (DCTs)

Change the frequency response of an audio signal by manipulating frequency-domain data.

Biquadratic IIR filters

Apply biquadratic filters to single-channel and multichannel data.

Discrete Cosine transforms

Transform vectors of temporal and spatial domain real values to the frequency domain, and vice versa.