Gmail

Matthew Klassen <mattjklassen@gmail.com>

Filter response curve

12 messages

Waverly Edwards <wedwards@k2m.com> Mon, Jan 16, 2017 at 12:20 PM
To: "coreaudio-api@lists.apple.com" <coreaudio-api@lists.apple.com>

May I ask what ways are there to get the response curve of a filter? My current method is to pass a white noise source through the filter and write the output to disk, then read the disk output into Audacity and plot the output.

This is the only way that I've come up which makes sense as I believe Audacity is using some form of Fourier Transform on the data. I've been looking at the vDSP functions for FFT. This is the direction that I am headed, using FFT but even though I don't know any other way, I feel this must be using a sledgehammer to kill a mosquito.

Any thoughts on this?

Thank you,

W.

Richard Dobson <richard@rwdobson.com> Tue, Jan 17, 2017 at 8:35 AM
To: coreaudio-api@lists.apple.com

The cleanest way is to feed a single "unit impulse" through the filter and store the output, which is by definition the impulse response (assuming this is a standard LTI filter). Then take the FFT of the impulse response to get the transfer function, which is the response curve you are looking for. The unit impulse is exactly what the name says - a single sample of full amplitude followed by silence.

Richard Dobson

On 16/01/2017 20:20, Waverly Edwards wrote:

> May I ask what ways are there to get the response curve of a filter? My
> current method is to pass a white noise source through the filter and
> write the output to disk, then read the disk output into Audacity and
> plot the output.
>
> This is the only way that I've come up which makes sense as I believe
> Audacity is using some form of Fourier Transform on the data. I've been
> looking at the vDSP functions for FFT. This is the direction that I am
> headed, using FFT but even though I don't know any other way, I feel
> this must be using a sledgehammer to kill a mosquito.

Any thoughts on this?

Thank you,

W.

Evan Balster <evan@imitone.com> Tue, Jan 17, 2017 at 9:30 AM
To: Richard Dobson <richard@rwdobson.com>
Cc: coreaudio-api@lists.apple.com

I must disagree. The cleanest way to compute the transfer function, is to *compute the transfer function*, directly — you'll get an exact answer, with minimal computational expense.

If I have a filter defined as:

```
y[t] = b0*x[t]
  + b1*x[t-1] + b2*x[t-2]
  - a1*y[t-1] - a2*y[t-2];
```

This corresponds to a transfer function:

H(z) = (b0 + b1*z^-1 + b2*z^-2) / (1 + a1*z^-1 + a2*z^-2)

(Note that "a" coefficients become negative.)

If I want to find the response of that filter to a frequency of 1000 hz, where the sample-rate is 10000 hz, I compute this Z-function for the corresponding complex frequency:

z = e^(i*2pi*f/S)
... = e^(i*2pi*1000/10000)
... = cos(2pi*.1) + i*sin(2pi*.1)

Pass that number through the transfer function H(z) above, and you'll get an exact answer for whatever frequency you like.

There's a lot of misinformation floating around about this topic, and I imagine it's because people are put off by complex numbers and Z-domain math. It isn't as spooky as it looks!

– Evan Balster
creator of imitone

On Tue, Jan 17, 2017 at 10:35 AM, Richard Dobson <richard@rwdobson.com> wrote:

> The cleanest way is to feed a single "unit impulse" through the filter and store the output, which is by definition the impulse response (assuming this is a standard LTI filter). Then take the FFT of the impulse response to get the transfer function, which is the response curve you are looking for. The unit impulse is exactly what the name says - a single sample of full amplitude followed by silence.
>
> Richard Dobson
>
> On 16/01/2017 20:20, Waverly Edwards wrote:
>
>> May I ask what ways are there to get the response curve of a filter? My current method is to pass a white noise source through the filter and write the output to disk, then read the disk output into Audacity and plot the output.
>>
>> This is the only way that I've come up which makes sense as I believe Audacity is using some form of Fourier Transform on the data. I've been looking at the vDSP functions for FFT. This is the direction that I am headed, using FFT but even though I don't know any other way, I feel this must be using a sledgehammer to kill a mosquito.
>>
>> Any thoughts on this?
>>
>> Thank you,
>>
>> W.
>>
>>
>
>

https://lists.apple.com/mailman/options/coreaudio-api/mklassen%40digipen.edu

This email sent to mklassen@digipen.edu

John Geelen <jgelectronics@icloud.com> Tue, Jan 17, 2017 at 10:27 AM
To: Waverly Edwards <wedwards@k2m.com>
Cc: coreaudio-api@lists.apple.com

Download the free book:
The Scientist and Engineer's Guide to Digital Signal Processing
In the book you will find everything you need to know about DSP.

regards,

John Geelen

> Op 16 jan. 2017, om 21:20 heeft Waverly Edwards <wedwards@k2m.com> het volgende geschreven:
>
> May I ask what ways are there to get the response curve of a filter? My current method is to pass a white noise source through the filter and write the output to disk, then read the disk output into Audacity and plot the output.
> This is the only way that I've come up which makes sense as I believe Audacity is using some form of Fourier Transform on the data. I've been looking at the vDSP functions for FFT. This is the direction that I am headed, using FFT but even though I don't know any other way, I feel this must be using a sledgehammer to kill a mosquito.
>
> Any thoughts on this?
>
> Thank you,
>
> W.
>
>

Richard Dobson <richard@rwdobson.com> Tue, Jan 17, 2017 at 11:05 AM
To: coreaudio-api@lists.apple.com

Indeed, if you have access to the coefficients, that is the proper way to do it. My suggestion was for the situations where you don't (or where getting a set of coefficients from the source code is something of a task in itself).

Richard Dobson

On 17/01/2017 17:30, Evan Balster wrote:

> I must disagree. The cleanest way to compute the transfer function
> <https://en.wikipedia.org/wiki/Digital_filter#Characterization>, is to
> /compute the transfer function/, directly — you'll get an exact answer,
> with minimal computational expense.

Roman Thilenius <politik@laut8leise.de> Tue, Feb 7, 2017 at 8:10 AM
To: coreaudio-api@lists.apple.com

isnt that exactly the same as sending a stream of one spike into the filter?

i am just not so sure if the transfer function itself really describes the "response curve" best. :)

On Jan 17, 2017, at 6:30 PM, Evan Balster wrote:

> I must disagree. The cleanest way to compute the transfer function, is to compute the transfer function, directly — you'll get an exact answer, with minimal computational expense.

Roman Thilenius <politik@laut8leise.de> Tue, Feb 7, 2017 at 12:46 PM
To: coreaudio-api@lists.apple.com

i guess i am elsewhere.

a "response curve" is normally a graphical plot of frequency vs gain, and its purpose is to give a useful visual feedback to the user.

the transfer function of a biquad or IIR filter tells the human eye almost nothing about the sound, you dont see anything at all (except maybe wether it is symetric or not.)

On Feb 7, 2017, at 6:22 PM, Evan Balster wrote:

> A filter's transfer function simultaneously describes its response to a one-sample impulse and its response to any complex frequency in the z-domain. There's no need to compromise.
>
> The reason for this? Applying a filter to a signal is the same as convolving the signal by the filter's impulse response.

> Convolution in the time domain, as exemplified by filters, is identical to multiplication in the frequency domain. Thus we can look at any point in the transfer function (as evaluated in the z-domain) and derive the effect the described filter will have on that frequency.
>
> On Tue, Feb 7, 2017 at 10:11 AM Roman Thilenius <politik@laut8leise.de> wrote:
>
> isnt that exactly the same as sending a stream of one spike into the
> filter?
>
> i am just not so sure if the transfer function itself really
> describes the "response curve" best. :)
>
> On Jan 17, 2017, at 6:30 PM, Evan Balster wrote:
>
> > I must disagree. The cleanest way to compute the transfer
> > function, is to compute the transfer function, directly — you'll
> > get an exact answer, with minimal computational expense.

Evan Balster <evan@imitone.com> Tue, Feb 7, 2017 at 1:17 PM
To: coreaudio-api@lists.apple.com

Whoop, meant to send this reply to the list:

> A filter's transfer function simultaneously describes its response to a one-sample impulse *and* its response to any complex frequency in the z-domain. There's no need to compromise.
> The reason for this? Applying a filter to a signal is the same as convolving the signal by the filter's impulse response.
> Convolution in the time domain, as exemplified by filters, is identical to multiplication in the frequency domain. Thus we can look at any point in the transfer function (as evaluated in the z-domain) and derive the effect the described filter will have on that frequency.

To expand on that: When we graph the filter response, the transfer function is what's getting graphed. A magnitude chart depicts the logarithm of the transfer function's magnitude for e^(iw) where w is angular frequency; a phase chart depicts the imaginary part of the logarithm.

– Evan Balster
creator of imitone

On Tue, Feb 7, 2017 at 10:10 AM, Roman Thilenius <politik@laut8leise.de> wrote:

> isnt that exactly the same as sending a stream of one spike into the filter?
>
> i am just not so sure if the transfer function itself really describes the "response curve" best. :)
>
> On Jan 17, 2017, at 6:30 PM, Evan Balster wrote:
>
> > I must disagree. The cleanest way to compute the transfer function, is to compute the transfer function, directly — you'll get an exact answer, with minimal computational expense.

>

Aran Mulholland <aranmulholland@gmail.com> Tue, Feb 7, 2017 at 8:00 PM
To: Evan Balster <evan@imitone.com>
Cc: Core Audio Mailing List <coreaudio-api@lists.apple.com>

Questions like these probably belong on the music dsp mailing list.

http://musicdsp.org/

Not saying people here don't know the correct answer, but there are probably more people there that can have the discussion.

By the way if you really want to know how Audacity works under the hood you could just go and look at the source code:

http://www.audacityteam.org/community/developers/#git

On Wed, Feb 8, 2017 at 8:17 AM, Evan Balster <evan@imitone.com> wrote:

> Whoop, meant to send this reply to the list:
>
> > A filter's transfer function simultaneously describes its response to a one-sample impulse *and* its response to any complex frequency in the z-domain. There's no need to compromise.
> > The reason for this? Applying a filter to a signal is the same as convolving the signal by the filter's impulse response. Convolution in the time domain, as exemplified by filters, is identical to multiplication in the frequency domain. Thus we can look at any point in the transfer function (as evaluated in the z-domain) and derive the effect the described filter will have on that frequency.
>
> To expand on that: When we graph the filter response, the transfer function is what's getting graphed. A magnitude chart depicts the logarithm of the transfer function's magnitude for e^(iw) where w is angular frequency; a phase chart depicts the imaginary part of the logarithm.
>
> – Evan Balster
> creator of imitone
>
> On Tue, Feb 7, 2017 at 10:10 AM, Roman Thilenius <politik@laut8leise.de> wrote:
>
> > isnt that exactly the same as sending a stream of one spike into the filter?
> >
> > i am just not so sure if the transfer function itself really describes the "response curve" best. :)
> >
> > On Jan 17, 2017, at 6:30 PM, Evan Balster wrote:
> >
> > > I must disagree. The cleanest way to compute the transfer function, is to compute the transfer function, directly — you'll get an exact answer, with minimal computational expense.

Brian Willoughby <brianw@audiobanshee.com>
To: Evan Balster <evan@imitone.com>
Cc: coreaudio-api@lists.apple.com

Tue, Feb 7, 2017 at 9:46 PM

Hello all,

I apologize for being pedantic, but I believe that even the smallest misrepresentation can cause a lot of grief for a newcomer.

To that end, I'd like to point out that the phase chart does not correspond to the imaginary part (of the "logarithm" or anything else).

Fourier Transforms produce paired results in real and imaginary parts. These real & imaginary pairs are also known as the rectangular form. In order to obtain the magnitude and phase, the rectangular coordinates must be converted to a polar form. In the polar form, the "radius" corresponds to the magnitude and the "angle" corresponds to the phase.

It's true that some software API reuse the imaginary buffer to hold the phase values after a conversion from rectangular to polar coordinates, but my point is that the data is never both imaginary and phase. When the data represents imaginary values, there is no real world analog of what those values mean. However, when converted to polar form, the newly calculated values do correspond to the phase response for each frequency bin.

A secondary and much more minor point is that the magnitude is not necessarily a logarithm. The only guarantee is that the magnitude is the square root of the sum of the squares of the real and imaginary parts. The primary result in a linear amplitude, but we humans aren't generally able to make as much sense of linear amplitudes. Therefore, it's nearly universal that the magnitudes will be converted to a decibel scale, which is a logarithm by definition.

Apart from these corrections, Evan is correct - especially about the fact that mathematics has proven that the impulse response of a linear, time-invariant system like a filter corresponds precisely to the frequency response.

p.s. When converting from rectangular to polar coordinates, the magnitude is calculated using the root of two squares, as mentioned above, while the phase is calculated using trigonometric math to determine the angle based on the x and y vectors from the real and imaginary parts. Since trigonometric math is usually rather expensive in terms of CPU cycles, many FFT-based spectrum algorithms will only calculate the magnitude and not bother calculating the phase. For most applications, the phase is not necessary, so leaving it out makes the program run faster. In other words, it's too bad that the phase is not the imaginary part, because that would mean we didn't have to do any further calculations!

Brian Willoughby
Sound Consulting

On Feb 7, 2017, at 1:17 PM, Evan Balster <evan@imitone.com> wrote:
> A filter's transfer function simultaneously describes its response to a one-sample impulse and its response to any complex frequency in the z-domain. There's no need to compromise.
> The reason for this? Applying a filter to a signal is the same as convolving the signal by the filter's impulse response. Convolution in the time domain, as exemplified by filters, is identical to multiplication in the frequency domain. Thus we can look at any point in the transfer function (as evaluated in the z-domain) and derive the effect the described filter will have on that frequency.
>
> To expand on that: When we graph the filter response, the transfer function is what's getting graphed. A magnitude chart depicts the logarithm of the transfer function's magnitude for e^(iw) where w is angular frequency; a phase chart depicts the imaginary part of the logarithm.

Evan Balster <evan@imitone.com> Tue, Feb 7, 2017 at 10:00 PM
Cc: coreaudio-api@lists.apple.com

Hey, Brian —

Apologies for the confusing language. I tend to prefer logarithmic representation because of its elegant similarity to polar representation; the real part of the complex logarithm is the logarithm of the input number's magnitude, while the imaginary part of the logarithm is the input number's phase.

For example, the natural logarithm of -1 is i*pi — a phase of 180 degrees and a log-magnitude of zero. The natural logarithm of e*i is 1+i*pi/2 — 1, because of the scalar e term, and pi/2 because of the quarter-circle orientation imposed by i.

That said, these concepts are a bit heady for the discussion here and I've probably created more confusion than I've resolved! We can discuss further on music-dsp if you like.

– Evan Balster
creator of imitone

On Tue, Feb 7, 2017 at 11:46 PM, Brian Willoughby <brianw@audiobanshee.com> wrote:

> Hello all,
>
> I apologize for being pedantic, but I believe that even the smallest misrepresentation can cause a lot of grief for a newcomer.
>
> To that end, I'd like to point out that the phase chart does not correspond to the imaginary part (of the "logarithm" or anything else).
>
> Fourier Transforms produce paired results in real and imaginary parts. These real & imaginary pairs are also known as the rectangular form. In order to obtain the magnitude and phase, the rectangular coordinates must be converted to a polar form. In the polar form, the "radius" corresponds to the magnitude and the "angle" corresponds to the phase.

It's true that some software API reuse the imaginary buffer to hold the phase values after a conversion from rectangular to polar coordinates, but my point is that the data is never both imaginary and phase. When the data represents imaginary values, there is no real world analog of what those values mean. However, when converted to polar form, the newly calculated values do correspond to the phase response for each frequency bin.

A secondary and much more minor point is that the magnitude is not necessarily a logarithm. The only guarantee is that the magnitude is the square root of the sum of the squares of the real and imaginary parts. The primary result in a linear amplitude, but we humans aren't generally able to make as much sense of linear amplitudes. Therefore, it's nearly universal that the magnitudes will be converted to a decibel scale, which is a logarithm by definition.

Apart from these corrections, Evan is correct - especially about the fact that mathematics has proven that the impulse response of a linear, time-invariant system like a filter corresponds precisely to the frequency response.

p.s. When converting from rectangular to polar coordinates, the magnitude is calculated using the root of two squares, as mentioned above, while the phase is calculated using trigonometric math to determine the angle based on the x and y vectors from the real and imaginary parts. Since trigonometric math is usually rather expensive in terms of CPU cycles, many FFT-based spectrum algorithms will only calculate the magnitude and not bother calculating the phase. For most applications, the phase is not necessary, so leaving it out makes the program run faster. In other words, it's too bad that the phase is not the imaginary part, because that would mean we didn't have to do any further calculations!

Brian Willoughby
Sound Consulting

On Feb 7, 2017, at 1:17 PM, Evan Balster <evan@imitone.com> wrote:
> A filter's transfer function simultaneously describes its response to a one-sample impulse and its response to any complex frequency in the z-domain. There's no need to compromise.
> The reason for this? Applying a filter to a signal is the same as convolving the signal by the filter's impulse response. Convolution in the time domain, as exemplified by filters, is identical to multiplication in the frequency domain. Thus we can look at any point in the transfer function (as evaluated in the z-domain) and derive the effect the described filter will have on that frequency.
>
> To expand on that: When we graph the filter response, the transfer function is what's getting graphed. A magnitude chart depicts the logarithm of the transfer function's magnitude for e^(iw) where w is angular frequency; a phase chart depicts the imaginary part of the logarithm.

Brian Willoughby <brianw@audiobanshee.com> Tue, Feb 7, 2017 at 10:13 PM
To: Evan Balster <evan@imitone.com>
Cc: coreaudio-api@lists.apple.com

Ok. I see what you mean now.

I'm referring to the format and representation of the numbers that we actually have in the memory buffers when running the software that we've written. All of the forms that can be expressed on paper have been put out of my mind so I can concentrate on the actual values sent to FFT subroutines and the values returned from those subroutines. All of my discussion of rectangular and polar coordinates directly corresponds to the format of the data parameters for common libraries like Accelerate and vecLib.

Folks are welcome to migrate to music-dsp for theory, if they like. However, if you want to discuss methods for writing code on macOS using CoreAudio and the Accelerate framework, then I recommend keeping the latter questions on this mailing list.

Brian Willoughby
Sound Consulting

p.s. I recently completed an application which graphs the Signal Transfer Function of selected AudioUnits. I have previously written an application which generates an impulse and graphs the frequency response of selected AudioUnits. These techniques are quite valuable when developing an AudioUnit (or even when building an AUGraph).

On Feb 7, 2017, at 10:00 PM, Evan Balster <evan@imitone.com> wrote:
> Hey, Brian —
>
> Apologies for the confusing language. I tend to prefer logarithmic representation because of its elegant similarity to polar representation; the real part of the complex logarithm is the logarithm of the input number's magnitude, while the imaginary part of the logarithm is the input number's phase.
>
> For example, the natural logarithm of -1 is i*pi — a phase of 180 degrees and a log-magnitude of zero. The natural logarithm of e*i is 1+i*pi/2 — 1, because of the scalar e term, and pi/2 because of the quarter-circle orientation imposed by i.
>
> That said, these concepts are a bit heady for the discussion here and I've probably created more confusion than I've resolved! We can discuss further on music-dsp if you like.
>
> – Evan Balster
> creator of imitone
>
> On Tue, Feb 7, 2017 at 11:46 PM, Brian Willoughby <brianw@audiobanshee.com> wrote:
>> Hello all,
>>
>> I apologize for being pedantic, but I believe that even the smallest misrepresentation can cause a lot of grief for a newcomer.
>>
>>
>> To that end, I'd like to point out that the phase chart does not correspond to the imaginary part (of the "logarithm" or anything else).
>>
>> Fourier Transforms produce paired results in real and imaginary parts. These real & imaginary pairs are also known as the rectangular form. In order to obtain the magnitude and phase, the rectangular coordinates must be converted to a polar form. In the polar form, the "radius" corresponds to the magnitude and the "angle" corresponds to the phase.
>>
>> It's true that some software API reuse the imaginary buffer to hold the phase values after a conversion from rectangular to polar coordinates, but my point is that the data is never both imaginary and phase. When the data represents imaginary values, there is no real world analog of what those values mean. However, when converted to polar form, the newly calculated values do correspond to the phase response for each frequency bin.
>>
>>
>> A secondary and much more minor point is that the magnitude is not necessarily a logarithm. The only guarantee is that the magnitude is the square root of the sum of the squares of the real and imaginary parts. The primary result in a linear amplitude, but we humans aren't generally able to make as much sense of linear amplitudes. Therefore, it's nearly universal that the magnitudes will be converted to a decibel scale, which is a logarithm by definition.
>>
>>
>> Apart from these corrections, Evan is correct - especially about the fact that mathematics has proven that the impulse response of a linear, time-invariant system like a filter corresponds precisely to the frequency response.
>>
>>
>> p.s. When converting from rectangular to polar coordinates, the magnitude is calculated using the root of two squares, as mentioned above, while the phase is calculated using trigonometric math to determine the angle based on the x and y vectors from the real and imaginary parts. Since trigonometric math is usually rather expensive in terms of CPU cycles, many FFT-based spectrum algorithms will only calculate the magnitude and not bother calculating the phase. For most applications, the phase is not necessary, so leaving it out makes the program run faster. In other words, it's too bad that the phase is not the imaginary part, because that would mean we didn't have to do any further calculations!
>>
>> Brian Willoughby

>> Sound Consulting
>>
>>
>> On Feb 7, 2017, at 1:17 PM, Evan Balster <evan@imitone.com> wrote:
>> > A filter's transfer function simultaneously describes its response to a one-sample impulse and its response to any complex frequency in the z-domain. There's no need to compromise.
>> > The reason for this? Applying a filter to a signal is the same as convolving the signal by the filter's impulse response. Convolution in the time domain, as exemplified by filters, is identical to multiplication in the frequency domain. Thus we can look at any point in the transfer function (as evaluated in the z-domain) and derive the effect the described filter will have on that frequency.
>> >
>> > To expand on that: When we graph the filter response, the transfer function is what's getting graphed. A magnitude chart depicts the logarithm of the transfer function's magnitude for e^(iw) where w is angular frequency; a phase chart depicts the imaginary part of the logarithm.
>