Thoughts on the analysis of the violin

This document has an in-depth explanation of all the math and file formats I use in the ObieApp.

1 Frequency Response Functions

1.1 What is a FRF?

We want to express the violin mathematically, we can view it as a transfer function (also called the frequency response function)- it converts some input (in this case a force from the hammer) to some output (in this case sound or a pressure fluctuation):

$$f(t) -> p(t) \tag{1}$$

or taking the Fourier Transform, we can find the transfer function for the violin as:

$$H(f) = \frac{P(f)}{F(f)} \tag{2}$$

where $P(f)$ and $F(f)$ are the Fourier transforms of pressure and force and have real and imaginary parts. Remember that the Fourier Transform is defined as:

$$G(f) = \int_{-\infty}^{\infty} g(t) e^{-i2\pi ft} dt \tag{3}$$

I can simplify - get the denominator to be real, by multiplying by F^*:

$$H(f) = \frac{P(f)}{F(f)} = \frac{PF^*}{FF^*} = H1 = \frac{PP^*}{FP^*} = H2 \tag{4}$$

1.2 Least Squares Fit

In reality, we take 5-10 hits per position so that we can get a "best fit" reading. We know that:

$$P(f) = H(f)F(f) \tag{5}$$

So the error of the best fit would be looking at the measured values (F_i and P_i):

$$\epsilon_i(f) = P_i - H(f)F_i \tag{6}$$

Now realizing that $H(f)$ is a complex number $(a + ib)$ and that F, P are complex as well, gives (remember that the magnitude of the product equals the product of the magnitudes):

$$\begin{aligned} \epsilon_i^2(f) &= (P_i - H(f)F_i)(P_i - H(f)F_i)^* \\ &= |P_i|^2 - H(f)F_iP_i^* - P_iH^*(f)F_i^* + |H(f)F_i|^2 \\ &= |P_i|^2 - (a+ib)F_iP_i^* - P_i(a-ib)F_i^* + |(a+ib)F_i|^2 \\ &= |P_i|^2 - (a+ib)F_iP_i^* - P_i(a-ib)F_i^* + (a^2+b^2)F_iF_i^* \end{aligned} \tag{7}$$

Next we sum over all the readings and then find the minimum by differentiating with respect to a and b and setting them equal to 0 to get:

$$\begin{aligned}\frac{\partial \sum \epsilon_i^2(f)}{\partial a} = 0 = 0 - \sum F_i P_i^* - \sum P_i F_i^* + 2a \sum F_i F_i^* \\ \frac{\partial \sum \epsilon_i^2(f)}{\partial b} = 0 = 0 - i \sum F_i P_i^* + i \sum P_i F_i^* + 2b \sum F_i F_i^*\end{aligned} \tag{8}$$

multiplying the second one by i and then adding you get:

$$\begin{aligned}0 = - \sum F_i P_i^* - \sum P_i F_i^* + 2a \sum F_i F_i^* \\ 0 = \sum F_i P_i^* - \sum P_i F_i^* + 2bi \sum F_i F_i^*\end{aligned} \tag{9}$$

I can add them together and get:

$$\begin{aligned}0 = -2 \sum P_i F_i^* + 2a \sum F_i F_i^* + 2bi \sum F_i F_i^* \\ \sum P_i F_i^* = H(f) \sum F_i F_i^*\end{aligned} \tag{10}$$

or

$$H_1(f) = \frac{\sum P_i F_i^*}{\sum F_i F_i^*} \tag{11}$$

Likewise we could look for reducing the effect of noise on the input – or using, as the error,

$$\epsilon_i(f) = F_i - \frac{P_i}{H(f)} \tag{12}$$

And do the analysis again and you will get:

$$\begin{aligned}\epsilon_i^2(f) &= (F_i - \frac{P_i}{H(f)})(F_i - \frac{P_i}{H(f)})^* \\ &= F_i F_i^* - F_i \frac{P_i^*}{H^*} - F_i^* \frac{P_i}{H} + \frac{P_i^*}{H^*} \frac{P_i}{H} \\ &= F_i F_i^* - F_i P_i^* (a - ib)^{-1} - F_i^* P_i (a + ib)^{-1} + P_i P_i^* (a^2 + b^2)^{-1}\end{aligned} \tag{13}$$

Again summing and differentiating:

$$\begin{aligned}\frac{\partial \sum \epsilon_i^2(f)}{\partial a} = 0 = 0 + (a - ib)^{-2} \sum F_i P_i^* + (a + ib)^{-2} \sum P_i F_i^* - (a^2 + b^2)^{-2} 2a \sum P_i P_i^* \\ \frac{\partial \sum \epsilon_i^2(f)}{\partial b} = 0 = 0 - i(a - ib)^{-2} \sum F_i P_i^* + i(a + ib)^{-2} \sum P_i F_i^* - (a^2 + b^2)^{-2} 2b \sum P_i P_i^*\end{aligned} \tag{14}$$

multiplying the second equation through by i gives you:

$$\begin{aligned} 0 &= (a-ib)^{-2}\sum F_i P_i^* + (a+ib)^{-2}\sum P_i F_i^* - (a^2+b^2)^{-2}2a\sum P_i P_i^* \\ 0 &= (a-ib)^{-2}\sum F_i P_i^* - (a+ib)^{-2}\sum P_i F_i^* - (a^2+b^2)^{-2}2ib\sum P_i P_i^* \end{aligned} \tag{15}$$

Adding them together and reshuffling:

$$\begin{aligned} H(a^2+b^2)^{-2}\sum P_i P_i^* &= (a-ib)^{-2}\sum F_i P_i^* \\ H\sum P_i P_i^* &= \frac{(a+ib)^2(a-ib)^2}{(a-ib)(a-ib)}\sum F_i P_i^* \\ \sum P_i P_i^* &= H\sum F_i P_i^* \\ H_2(f) &= \frac{\sum P_i P_i^*}{\sum F_i P_i^*} \end{aligned} \tag{16}$$

gives us the best fit in the other direction (assuming that the input is noisy). If the data is clean and the noise is truly random, these two values for H should be very close and therefore their ratio, the coherence, should be 1.

$$C = \frac{H_1}{H_2} \tag{17}$$

If it is not 1, then one (or both) of your signals is noisy (not a great least squares fit). George then used the geometric mean.

$$H \sim \sqrt{H_1 H_2} \tag{18}$$

1.3 Claude's explanation: Coherence

Coherence is essentially a measure of how consistently a system behaves across repeated measurements.

Think of it this way: you tap a violin with a hammer and measure the input force and the output sound. If you tap it again with a slightly different force, does the ratio of output to input stay the same? If the system is well-behaved (linear, no noise), it should — and the coherence will be close to 1. If something is messy (nonlinearity, background noise, a loose part rattling), the ratio will vary between hits, and coherence drops toward 0. More concretely, coherence answers: "how much of the output can be explained by the input?" The formula from the document makes this intuitive:

$$C = \frac{H1}{H2} = \frac{|\sum P_i F_i^*|^2}{\sum F_i F_i^* \sum P_i P_i^*} \tag{19}$$

The numerator is the cross-spectrum squared — how much F and P move together. The denominator is the product of the individual power spectra — how much each signal varies on its own. It's essentially a frequency-domain analog of R^2 from linear regression. A few practical rules of thumb:

- C $\sim$ 1: the output is well explained by the input — clean measurement, linear system
- C $\sim$ 0: the output has little to do with the input at that frequency — probably noise, or the system simply doesn't respond there
- C between 0 and 1: partial explanation — could be noise, nonlinearity, or multiple inputs contributing to the output

One important subtlety: coherence is frequency-dependent. A violin might have C $\sim$ 1 near a resonance (where it responds strongly and predictably) but low coherence in a frequency range where it barely responds and the output is dominated by noise. So you'd typically plot it across the whole frequency range rather than summarize it as a single number.

2 Band Averages

One can look at an average distribution of the energy by breaking the FRF up into bands and averaging the FRF magnitudes over that area.

$$\bar{b} = \frac{1}{N}\sum_{f_{lo}}^{f_{hi}} \sqrt{H(f)H^*(f)} \qquad \bar{b}_{dB} = 20log(\frac{\bar{b}}{1}) \tag{20}$$

We also calculate the centroid of the band - that is the first moment:

$$c(f) = \frac{\sum_{f_{lo}}^{f_{hi}} f\sqrt{HH^*}}{\sum_{f_{lo}}^{f_{hi}} \sqrt{HH^*}} \tag{21}$$

which tells you something about the variation of the energy within the band. or in Python:

```
f_lo  = float(band['start'])
f_hi  = float(band['end'])
mask = (freq >= f_lo) & (freq <= f_hi)
H_band   = H_mag[mask]  #NOT in dB
f_band   = freq[mask]
avg_db   = 20.0 * math.log10(max(float(np.mean(H_band)), 1e-12))
centroid = float(np.sum(f_band * H_band) / np.sum(H_band))
```

3 Convolutions

One can estimate what the instrument will sound like by convolving the FRF with a recording of a song - but using a force sensor at the bridge rather than a microphone. So

if you want to convolve two functions $f(t)$ and $g(t)$

$$y(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{22}$$

3.1 What they are - from Claude

What convolution does: A violin has resonances — certain frequencies ring longer and louder than others. If you record the violin's frequency response (the FRF), you have a fingerprint of exactly how it shapes sound. Convolution applies that fingerprint to any audio signal. Feed in a Tchaikovsky recording; get out what it would sound like played through or colored by that violin's resonance structure.

How it works: Sound is made of many frequencies added together. The FRF tells you how much the violin boosts or cuts each frequency — and by how much it time-shifts it (phase part).

Note that you can speed up the calculation by multiplying the two waveforms in frequency space and then inverse Fourier transforming them into time space:

$$\begin{aligned} y(t) &= iFFT(FFT(g(t)) \times FFT(f(t)) \\ &= \int_{-\infty}^{\infty} [\int_{-\infty}^{\infty} g(\tau)e^{-i2\pi f\tau}d\tau][\int_{-\infty}^{\infty} f(\tau)e^{-i2\pi f\tau}d\tau]e^{i2\pi ft}df \end{aligned} \tag{23}$$

or in Python:

```
y = np.fft.irfft(
    np.fft.rfft(wav, n_fft) * np.fft.rfft(ir.astype(np.float64), n_fft),
    n_fft,)[:N].real
```

What you hear: The output is the input signal with the violin's character stamped onto it — its resonant peaks amplified, its dead spots suppressed, its natural decay times imposed on every note. If the FRF came from a particularly resonant instrument, the result sounds richer; from a dead one, duller.

3.2 minimum phase reconstruction from Claude

In the case that you do not have the complex FRF, you can still estimate the convolution by realizing for a minimum phase system, the log-magnitude and phase are a Hilbert transform pair. So if you know one, you can compute the other. We do not need to make use of this since everything is stored as complex FRFs.

4 File Formats

4.1 TRF files

Parse and build binary `.trf` files (Rational Acoustics / MtxVec format).

Header Format

Little-endian, 110 bytes total:

Type	Fields
`I`	`index`
`4d`	`Xr, Yr, Xactual, YActual`
`2s`	`char_str`
`3d`	`Hz_Resolution, Start_Freq, End_Freq`
`11f`	`fComplex, fLength, ...` (9 unused floats)
`4s`	`caption`

Data (from byte 110)

- **Complex:** alternating `float64` real/imag pairs
- **Real:** plain `float64` values

4.2 File Formats

4.2.1 AvR / AvC File Format Specification

AvR files store a *real-valued* averaged spectrum; AvC files store a *complex-valued* averaged spectrum. The file layout is identical except for the data section, which is $N \times 8$ bytes for AvR and $N \times 16$ bytes (interleaved Re/Im pairs) for AvC, where N is given by `fLength` in the TSingleArray descriptor. Type conventions: fields prefixed `f` are Single-precision floats (4 bytes); all other 4-byte TSingleArray fields are U32.

Field	Bytes	Type	Description / Possible Values
Header			
DataType	1	TDataType	Response quantity (see enum below).
HzRes	8	Double	Frequency resolution Δf (Hz / bin). > 0.
StartFrequency	8	Double	Start frequency (Hz). ≥ 0.

Field	Bytes	Type	Description / Possible Values
EndFrequency	8	Double	End frequency (Hz). > StartFrequency.
ScaleFactor	8	Double	Scale factor applied to the averaged data.
nAverages	4	U32	Number of averages used. Typically 1–1000.
AverageType	1	Enum	Averaging method (see enum below).
AverageType enum values			
RMS	—	value 0	Root-mean-square averaging.
Mean	—	value 1	Linear (arithmetic mean) averaging.
Complex	—	value 2	Complex (vector) averaging.
Geometric	—	value 3	Geometric mean averaging.
None	—	value 4	No averaging (single measurement).
TSingleArray (MtxVec array descriptor — all fields 4 bytes)			
fComplex	4	Single	`0.0` = real (AvR), `1.0` = complex (AvC).
fLength	4	Single	Number of data points N (stored as float). Determines data section size.
a[2]	4	U32	Reserved / alignment padding.
fConditionCheck	4	Single	Internal integrity / validation value.
Precision	4	U32	`0` = Single (32-bit), `1` = Double (64-bit).
MtxVecVersion	4	U32	Version of MtxVec library that wrote the file.
SizeOf(TSample)	4	U32	Byte size of each data sample.
Tag	4	U32	User-defined tag or identifier.
MtxVecFileCode	4	U32	Magic number identifying this as a MtxVec file.
a[9]	4	U32	Reserved / alignment padding.
a[10]	4	U32	Reserved / alignment padding.
caption	4	String	Short label, 4 bytes, null-padded (`\00\00\00\00` if empty).
Data (length determined by `fLength` and `fComplex`)			

Field	Bytes	Type	Description / Possible Values
AvR data	$N \times 8$	Double[]	N real-valued doubles, little-endian. Present when `fComplex = 0`.
AvC data	$N \times 16$	Double[]	N complex samples as interleaved Re / Im double pairs ($2N$ doubles total), little-endian. Present when `fComplex = 1`.
Notes / Comments (appended after data section)			
notes	var	String	Free-text annotation. Starts at byte offset $N \times 8$ (AvR) or $N \times 16$ (AvC) from the beginning of the data section, i.e. after `fLength` or $2\times$`fLength` doubles respectively.

Data section size summary:

AvR: $N \times 8$ bytes AvC: $N \times 16$ bytes where $N =$ `int(fLength)`

Notes: (1) `f`-prefixed TSingleArray fields are Single-precision floats (4 bytes). (2) `MtxVecFileCode` and `SizeOf(TSample)` are U32. (3) `caption` is 4 bytes, null-padded. (4) All multi-byte values are little-endian. (5) `TDataType` is stored as a single byte (U8). Source: `george.pas` (Dew Research / DSP Master) and LabVIEW writer.

4.2.2 FRF File Format Specifications

Fields verified against the Delphi source (`george.pas`) and LabVIEW writer code. Type conventions: fields prefixed `f` are Single-precision floats (4 bytes); `MtxVecFileCode` and `SizeOf(TSample)` are U32; `caption` is a null-padded string.

Field	Bytes	Type	Description / Possible Values
TFRFHeader (Delphi packed record — no padding between fields)			
index	4	Integer	Channel or measurement index. ≥ 0.
Xr	8	Double	Reference (input) channel number or physical X location.
Yr	8	Double	Response (output) channel number or physical Y location.
XActual	8	Double	Physical X-axis value (e.g. excitation location).

Field	Bytes	Type	Description / Possible Values
YActual	8	Double	Physical Y-axis value (e.g. response location).
FRFType	1	TFRFType	`0` = `ftTransfer` (`.trf`), `1` = `ftPoint` (`.pnt`).
DataType	1	TDataType	Response quantity (see enum below).
HzRes	8	Double	Frequency resolution Δf (Hz / bin). > 0.
StartFrequency	8	Double	Start frequency (Hz). ≥ 0.
EndFrequency	8	Double	End frequency (Hz). $>$ StartFrequency.
TDataType enum values (U8)			
dtAccel	—	value 0	Accelerance (acceleration / force).
dtMob	—	value 1	Mobility (velocity / force).
dtRecept	—	value 2	Receptance (displacement / force).
dtMic	—	value 3	Microphone response.
TSingleArray (MtxVec internal array descriptor — all fields 4 bytes)			
fComplex	4	Single	Array-level complex flag: `0.0` = real, `1.0` = complex.
fLength	4	Single	Number of elements in the data array (stored as float).
a[2]	4	U32	Reserved / alignment padding.
fConditionCheck	4	Single	Internal integrity / validation value.
Precision	4	U32	`0` = Single (32-bit), `1` = Double (64-bit).
MtxVecVersion	4	U32	Version of MtxVec library that wrote the file.
SizeOf(TSample)	4	U32	Byte size of each data sample.
Tag	4	U32	User-defined tag or identifier.
MtxVecFileCode	4	U32	Magic number identifying this as a MtxVec file.
a[9]	4	U32	Reserved / alignment padding.
a[10]	4	U32	Reserved / alignment padding.
Payload			
caption	4	String	Short label, 4 bytes, null-padded (`\00\00\00\00` if empty).
FRF data	$N \times 16$	Double[]	Complex FRF samples as interleaved Re / Im DBL pairs, little-endian. N = number of frequency lines.

5 Adiabatic Wave Equation

If the fluid is adiabatic, then

$$p = K\rho^\gamma$$

so

$$\frac{\partial p}{\partial x} = K\gamma\rho^{\gamma-1}\frac{\partial \rho}{\partial x} = c^2\frac{\partial \rho}{\partial x}$$

and

$$\frac{\partial p}{\partial t} = K\gamma\rho^{\gamma-1}\frac{\partial \rho}{\partial t} = c^2\frac{\partial \rho}{\partial t}$$

But from continuity

$$\frac{\partial \rho}{\partial t} + \rho_0\frac{\partial u}{\partial x} = 0$$

so

$$\frac{\partial p}{\partial t} + \frac{\rho_0}{c^2}\frac{\partial u}{\partial x} = 0$$

$$\frac{\partial p}{\partial t} = -\frac{\rho_0}{c^2}\frac{\partial^2 y}{\partial x \partial t}$$

or integrating by time,

$$p = -\frac{\rho_0}{c^2}\frac{\partial y}{\partial x} = -\kappa\frac{\partial y}{\partial x}$$

where κ is the adiabatic bulk modulus

and

$$\rho_0\frac{\partial u}{\partial t} + \frac{\partial p}{\partial x} = 0$$

or

$$\frac{\partial p}{\partial x} = -\rho_0\frac{\partial^2 y}{\partial t^2}$$

and

$$p = K\rho^\gamma$$

so

$$\frac{\partial p}{\partial x} = K\gamma\rho^{\gamma-1}\frac{\partial \rho}{\partial x} = c^2\frac{\partial \rho}{\partial x}$$

5.1 wave equation

Assume that the pressure is p and the speed of a fluid particle is u with a mean of zero and the density everywhere is $\rho_0 + \rho$. Then the 1D continuity gives us:

$$\frac{\partial(\rho_0 + \rho)}{\partial t} + \frac{\partial(\rho_0 + \rho)u}{\partial x} = 0$$

or

$$\frac{\partial \rho}{\partial t} + \rho_0\frac{\partial u}{\partial x} + \frac{\partial(\rho u)}{\partial x} = 0$$

and assuming the last term is small compared to the second to last, you get:

$$\frac{\partial \rho}{\partial t} + \rho_0\frac{\partial u}{\partial x} = 0$$

or

$$\frac{\partial^2 \rho}{\partial t^2} + \rho_0 \frac{\partial^2 u}{\partial x \partial t} = 0$$

The momentum equation (1D, no gravity or friction terms) is:

$$\frac{\partial((\rho_0 + \rho)u)}{\partial t} + (\rho_0 + \rho)u\frac{\partial u}{\partial x} + \frac{\partial p}{\partial x} = 0$$

or

$$\rho_0 \frac{\partial u}{\partial t} + \rho \frac{\partial u}{\partial t} + u\frac{\partial \rho}{\partial t} + \rho_0 u\frac{\partial u}{\partial x} + \rho u\frac{\partial u}{\partial x} + \frac{\partial p}{\partial x} = 0$$

Assuming the fluctuations are small:

$$\rho_0 \frac{\partial u}{\partial t} + \rho_0 u\frac{\partial u}{\partial x} + \frac{\partial p}{\partial x} = 0$$

Differentiate with respect to x:

$$\rho_0 \frac{\partial^2 u}{\partial t \partial x} + \rho_0 u\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 p}{\partial x^2} = 0$$

Dropping the middle term since derivatives in u should be about the same in both x and t and therefore the first term should be a lot bigger than the second - and combining both equations you get:

$$\frac{\partial^2 p}{\partial x^2} = \frac{\partial^2 \rho}{\partial t^2}$$

If we assume an adiabatic process, then

$$p = K\rho^\gamma$$

So

$$\frac{\partial p}{\partial t} = \gamma K \rho^{\gamma-1}\frac{\partial \rho}{\partial t}$$

or

$$\frac{\partial^2 p}{\partial t^2} = \gamma K \rho^{\gamma-1}\frac{\partial^2 \rho}{\partial t^2} + \gamma(\gamma-1)K\rho^{\gamma-2}(\frac{\partial \rho}{\partial t})^2$$

Linearizing loses the last term:

$$\frac{\partial^2 p}{\partial t^2} = \gamma K \rho^{\gamma-1}\frac{\partial^2 \rho}{\partial t^2}$$

or

$$\frac{\partial^2 p}{\partial t^2} = \gamma \frac{p}{\rho^\gamma}\rho^{\gamma-1}\frac{\partial^2 \rho}{\partial t^2}$$

or

$$\frac{\partial^2 p}{\partial t^2} = \gamma RT\frac{\partial^2 \rho}{\partial t^2}$$

Combining then gives

$$\frac{\partial^2 p}{\partial x^2} = \frac{\partial^2 p}{\partial t^2}\frac{1}{\gamma RT}$$

or

$$\frac{\partial^2 p}{\partial t^2} = \gamma RT\frac{\partial^2 p}{\partial x^2}$$

5.2 Acoustic Impedance

Next, we want to look at the acoustic impedance of the plate (acoustic impedance, Z, is the ratio of pressure to volume flow, specific acoustic impedance, z, is the ratio of pressure to volume flow per unit area - or acoustic flow velocity).

$$p = ZU$$

For an infinitely long pipe, cross-sectional area S, we get:

$$Z = \frac{p}{U} = \frac{F}{SU} = ... = \frac{\rho c}{S}$$

- I do not see it

Defining specific acoustic impedance to be:

$$p = zu$$

where a change in pressure (p) drives the fluid (u) like a voltage drives a current.

then

$$\frac{F/A}{u} = z$$

We can look at the forces on the plate, there should be the force due to pressure, a spring-like restoring force and a damping force due to the fact that the material is hardly a perfect spring:

$$\sum F = ma = pA - kx - \beta u$$

or

$$z = \frac{ma + kx + \beta u}{Au} = \frac{m}{A}\frac{\ddot{x}}{\dot{x}} + \frac{\beta}{A} + \frac{k}{A}\frac{x}{\dot{x}}$$

If we assume a solution of the form

$$u = \dot{x} = Ce^{i\omega}$$

then the impedance becomes:

$$z = \frac{m}{A}\frac{i\omega Ce^{i\omega}}{Ce^{i\omega}} + \frac{\beta}{A} + \frac{k}{A}\frac{Ce^{i\omega}/i\omega}{Ce^{i\omega}}$$

$$z = \frac{m}{A}i\omega + \frac{\beta}{A} + \frac{k}{A}\frac{1}{i\omega}$$

or

$$z = \frac{\beta}{A} + i\omega\frac{m}{A} - \frac{i}{\omega\frac{A}{k}}$$

or

$$z = r + i\omega M - \frac{i}{\omega C}$$

where

$$r = \frac{\beta}{A}$$

$$M = \frac{m}{A}$$

$$C = \frac{A}{k}$$

5.3 Sound Intensity Measurements

Sound intensity is defined as the sound power per unit surface area. For a point source:

$$I = \frac{P}{4\pi r^2}$$

So the Sound Power Level is referenced to $P_0 = 10^{-12}W$ and the Sound Intensity Level is referenced to $I_0 = 10^{-12}W/m^2$

$$SWL = 10log_{10}(P/P_0) and SIL = 10log_{10}(I/I_0)$$

Since

$$I = pu = \frac{p^2}{z}$$

then the sound pressure level is:

$$SPL = 10log_{10}(p^2/p_0^2) = 20log_{10}(p/p_0)$$

Sound hole is modeled with a frictional resistance R_h:

$$m_{hole-air}\ddot{x}_h = A_h\Delta p - R_h\dot{x}_h$$

and the top plate has a restoring force:

$$m_{plate}\ddot{x} = F - k_p x_p - R_p\dot{x}_p + A_p\Delta p$$

5.4 Quality factor

Usually defined as

$$Q = \frac{f_{center}}{\Delta f} = \frac{\sqrt{m}k}{R} = \frac{\omega_0}{2r}$$

were the spread of the peak (Δf) is typically measured by the 1/2 power reduction point, r is the exponential decay of the signal and R is the damping coefficient ($m\ddot{x} + R\dot{x} + kx = 0$)