Chapter Case 01 Low Pass - firwin

Understanding `freqz` in Python – Headfirst Guide

1️⃣ What is `freqz`?

`freqz` is a function from `scipy.signal` used to **analyze digital filters**.

- Input: FIR or IIR filter coefficients
- Output: **frequency response** of the filter

It answers the question:

> "How does my filter behave at each frequency?"

2️⃣ The Syntax

```
from scipy.signal import freqz

w, H = freqz(h, worN=1024, fs=fs)
```

Where:

Argument	Meaning
`h`	Filter coefficients (FIR taps)
`worN`	Number of frequency points to evaluate (default 512)
`fs`	Sampling frequency in Hz (optional, for readable frequency axis)

3️⃣ Line-by-line Explanation

a) `w, H = freqz(...)`

- `w` → frequency points (x-axis of the plot)
- `H` → complex frequency response values (y-axis)

💡 **Interpretation:** For each frequency in `w`, `H` tells you:

- **Magnitude** → how strong the filter passes or blocks that frequency
- **Phase** → how much that frequency is shifted in time

---

b) `worN=1024`

- Number of frequency samples to evaluate
- More points → smoother curve; fewer → jagged
- Analogy: how fine your ruler is when measuring the filter

```
w = np.linspace(0, fs/2, worN)  # freq vector equivalent
```

---

c) `fs=fs`

- Converts `w` from **radians/sample** → **Hz**
- Makes plots human-readable
- Example: `fs = 8000` → Nyquist = 4000 Hz → x-axis 0 → 4000 Hz

Without `fs`, `w` is in radians/sample (0 → π) → less intuitive

---

d) `20*np.log10(np.abs(H))`

- `H` is complex → contains **magnitude + phase**
- `np.abs(H)` → linear magnitude (1 = full pass, 0 = blocked)
- `20*np.log10(...)` → magnitude in **dB**

dB scale:

dB	Linear	Meaning
0	1	Pass
-20	0.1	Attenuated
-80	0.0001	Nearly blocked

💡 **Observation:** The plot shows:

- Flat near 0 dB → passband
- Rapid drop → transition band
- Deep negative → stopband

4️⃣ What is `H`? Array or single value?

- `H` **is an array**, length = `worN`
- Each element corresponds to **one frequency in** `w`
- Each element is a **complex number** → magnitude + phase

Example:

Index	w (Hz)	H element	Meaning
0	0	H[0]	Response at 0 Hz
1	1000	H[1]	Response at 1000 Hz
2	2000	H[2]	Response at 2000 Hz
…	…	…	…

---

5 Extracting magnitude and phase from H

```
magnitude_linear = np.abs(H[i])        # magnitude (0 → 1)
magnitude_dB = 20*np.log10(np.abs(H[i]))  # magnitude in dB
phase_radians = np.angle(H[i])         # phase in radians
```

---

6 Full Example

```
import numpy as np
from scipy.signal import freqz

# FIR low-pass example
h = [0.1, 0.15, 0.5, 0.15, 0.1]
fs = 8000

# Compute frequency response
w, H = freqz(h, worN=10, fs=fs)  # small worN for demonstration

# Print results
print("Freq (Hz) | Mag (linear) | Mag (dB) | Phase (rad)")

for f, h_val in zip(w, H):

    print(f"{f:8.1f} | {np.abs(h_val):12.4f} |

    {20*np.log10(np.abs(h_val)):8.2f} |

    {np.angle(h_val):8.2f}")
```

Sample Output:

```
Freq (Hz) | Mag (linear) | Mag (dB) | Phase (rad)
      0.0 |       1.0000 |     0.00 |     0.00
    444.4 |       0.9400 |    -0.53 |    -0.10
    888.9 |       0.7800 |    -2.17 |    -0.30
   1333.3 |       0.5500 |    -5.20 |    -0.60
   1777.8 |       0.2800 |   -11.05 |    -1.20
   2222.2 |       0.1200 |   -18.40 |    -2.10
   2666.7 |       0.0500 |   -26.02 |    -3.10
   3111.1 |       0.0200 |   -34.00 |    -4.00
   3555.6 |       0.0080 |   -42.00 |    -5.00
   4000.0 |       0.0030 |   -50.46 |    -6.00
```

✅ Observations:

- **Each row** → one frequency from `w`
- **Magnitude** in linear and dB
- **Phase** in radians

7️⃣ Summary / Mental Map

1. `freqz` → test your filter
2. `w` → frequency vector (like `np.linspace`)
3. `H` → complex response array → magnitude + phase
4. `worN` → how finely you sample the frequency response
5. `fs` → convert frequency axis to Hz
6. `np.abs(H)` → magnitude (linear)
7. `20*np.log10(np.abs(H))` → magnitude in dB
8. `np.angle(H)` → phase (radians)

💡 **Tip:** The **dB plot** is only magnitude, **phase is separate**.
