260223-plotting-index-v-freq-in-class

February 25, 2026

1 Index of Refraction

We have had several equations for the index of refraction and I want to show them all to you and the different features and effects in each of them.

First, we will import some things, then I'll show the first equation, then the next, and finally I'll show what happens when light hits a metal.

[15]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_theme()
```

So the first equation that we had for the index of refraction was this:

$$n = 1 + \frac{N * q_e^2}{2\epsilon_0 m_e(\omega_0^2 - \omega^2)}$$

For convenience, I'll replace

$$\frac{Nq_e^2}{\epsilon_0 m_e} = \omega_p^2$$

So that,

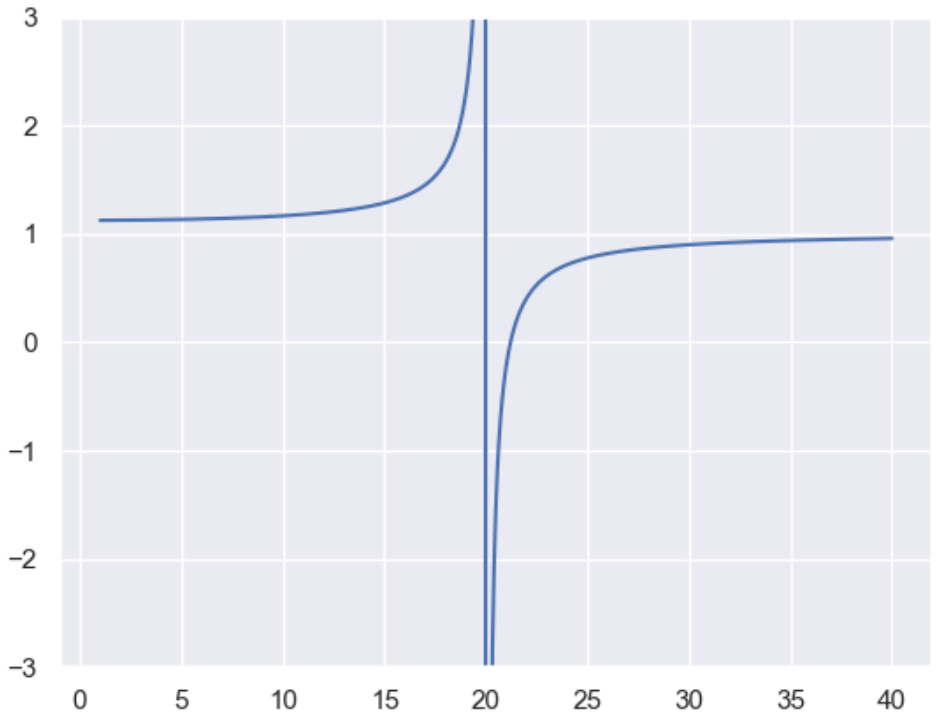
$$n = 1 + \frac{\omega_p^2}{\omega_0^2 - \omega^2}$$

That function is always real, so what does that look like?

[16]:
```
def n(w, wp, w0):
    return(1+wp**2/(2*(w0**2 - w**2)))

w = np.linspace(1, 40, 1000)

n_air = n(w, 10, 20)

fig0, ax0 = plt.subplots()
ax0.plot(w, n_air)
```

```
ax0.set_ylim(-3, 3)
```

[16]: (-3.0, 3.0)



So what do we see there? This is a rational function that is almost always 1, but as the frequency of light increases then the index increases. This approaches a barrier, where I would not imagine that our approximations are very good. But still it does match our intuition that the index of refraction will generally increase for increasing frequency, but then confusing things might happen if that gets too high.

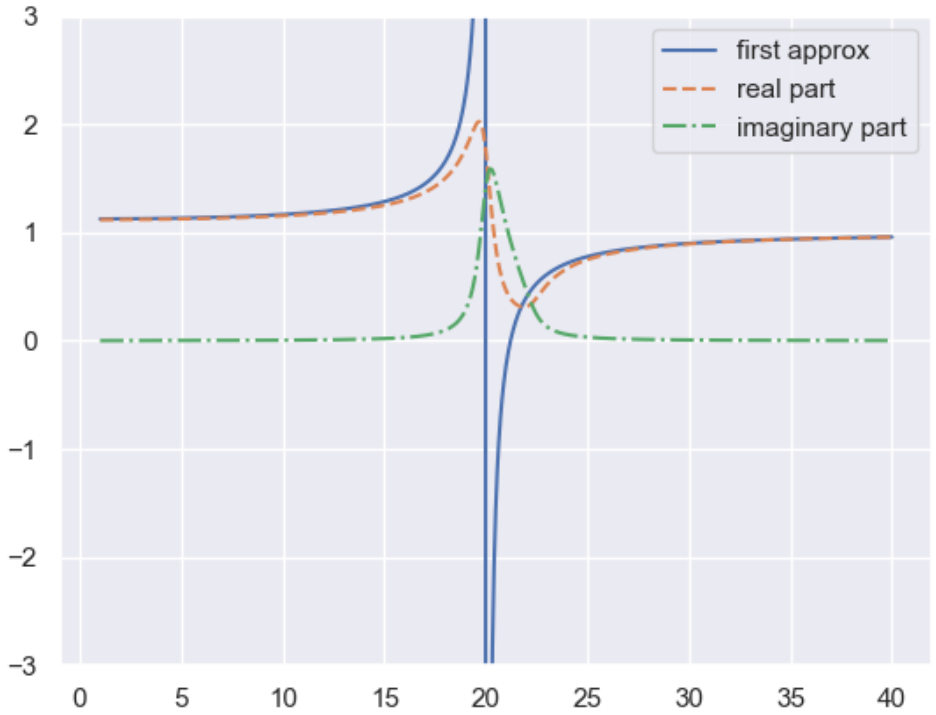
So now, what about our next approximation? That one is based off the polarization of the material, and it involves using Maxwell's equations so we would expect it to be better:

$$\tilde{n} = n + ik = \sqrt{1 + \frac{\omega_p^2}{\omega_0^2 - i\omega\gamma - \omega^2}}$$

Now this function is complex, so we will expect a real and imaginary part to come out, but that is fine since we can plot them separately.

[17]:
```
def complexn(w, w0, wp, g):
    return(np.sqrt(1+wp**2/(w0**2 - 1j*w*g-w**2)))
```

```
ntilde = complexn(w, 20, 10, 1)

# now add these plots to the previous one
fig1, ax1 = plt.subplots()
ax1.plot(w, n_air)
ax1.plot(w, ntilde.real, '--')
ax1.plot(w, ntilde.imag, '-.')
ax1.legend(['first approx', 'real part', 'imaginary part']);
ax1.set_ylim(-3, 3);
```



So hopefully you can see there how we are refining the method and making it more realistic.