HW 9

Lewis Collum

Updated: April 24, 2020

1

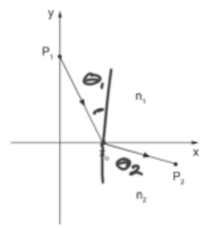
$$u_p = \frac{c}{n}$$

$$t = \frac{\sqrt{x_0^2 + P_{1y}^2}}{c/n_1} + \frac{\sqrt{(P_{2x} - x_0)^2 + P_{2y}^2}}{c/n_2}$$

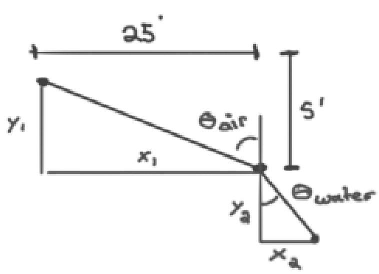
$$\frac{dt}{dx} = \frac{n_1 x_0}{c\sqrt{x_0^2 + P_{1y}^2}} + \frac{-n_2(1-x_0)}{c\sqrt{(1-x_0)^2 + P_{1y}^2}} = 0 \quad \text{minimizing time}$$

$$\rightarrow \frac{n_1 x_0}{\sqrt{x_0^2 + P_{1y}^2}} = \frac{n_2(1-x_0)}{\sqrt{(1-x_0)^2 + P_{1y}^2}}$$

$$n_1 \sin\theta_1 = n_2 \sin\theta_2$$



2



$$90° - \theta_{air} = \tan^{-1}\left(\frac{y_1}{x_1}\right)$$
$$= \tan^{-1}\left(\frac{5}{25}\right)$$
$$= 11.31°$$

$$\theta_{air} = 78.69°$$

$$n_{air}\sin\theta_{air} = n_{water}\sin\theta_{water}$$

$$\frac{\sin\theta_{air}}{\sin\theta_{water}} = \frac{n_{water}}{n_{air}} = \frac{\sqrt{81}}{\sqrt{1}} = 9$$

$$\sin\theta_{water} = \frac{1}{9}\sin\theta_{air}$$

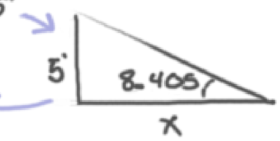
$$\theta_{water} = \sin^{-1}\left(\frac{1}{9}\sin\theta_{air}\right)$$

$$\boxed{\theta_{water} = 6.255°}$$

$$\frac{\sin\theta_{air}}{\sin\theta_{water}} = \sqrt{\frac{\mu_1\varepsilon_1}{\mu_2\varepsilon_2}}$$

$$\sin\theta_{air} = \sqrt{\frac{1.8}{1}}\sin\theta_{water}$$

$$\theta_{air} = 8.405°$$



$$\tan\theta_{air} = \frac{5}{x}$$

$$x = \frac{5}{\tan(8.405°)}$$

$$\boxed{= 33.84'}$$

3

```python
from numpy import *
nAir = 1
nGan = 3.7
criticalAngle = arcsin(nAir/nGan)
print(f'\[\\theta_c ='
      f'{rad2deg(criticalAngle):.3f}^{{\circ}}\]')

solidAngleCone = 2*pi*(1-cos(criticalAngle)) #4*pi
solidAngleHemisphere = 2*pi

coneCount = 2
solidAngleRatio = coneCount * solidAngleCone/solidAngleHemisphere
print(f'\[\\text{{solid Angle Ratio}} = {solidAngleRatio:.3f}\]')

import pint
unit = pint.UnitRegistry()

wavelength = 560
h = 6.63E-34
c = 3E8
E = solidAngleRatio * h*c/wavelength
print(f'\[\\text{{fraction of optical energy}} = \\text{{{E:.2E} J}}\]')
```

$$\theta_c = 15.680°$$

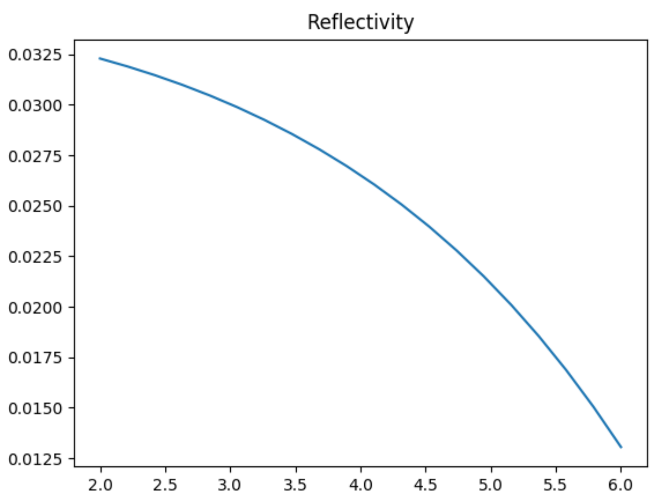
$$\text{solid Angle Ratio} = 0.074$$

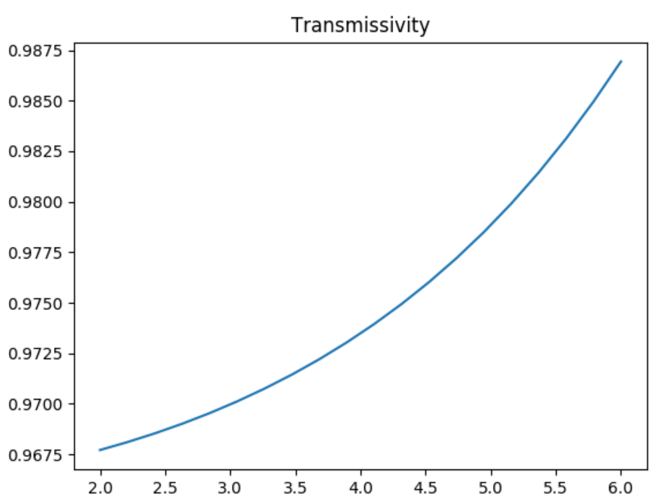
$$\text{fraction of optical energy} = \text{2.64E-29 J}$$

4

```python
from matplotlib import pyplot
import numpy

x = numpy.linspace(2, 6, 20) #wavelenth

#impedance for quartz
eta2 = 1/numpy.sqrt(1 +
              0.6961663*x**2 / (x**2 - 0.0684043**2) +
              0.4079426*x**2 / (x**2 - 0.1162414**2) +
              0.8974794*x**2 / (x**2 - 9.896161**2))
```

```python
#impedance for air
eta1 = 1

reflectionCoefficient = (eta2 - eta1)/(eta1 + eta2)
transmissionCoefficient = 1 + reflectionCoefficient

reflectivity = reflectionCoefficient**2
transmissivity = transmissionCoefficient**2 * (eta1/eta2)

pyplot.plot(x, reflectivity)
pyplot.title('Reflectivity')
pyplot.savefig('figure/4-re.png')
pyplot.show()

pyplot.plot(x, transmissivity)
pyplot.title('Transmissivity')
pyplot.savefig('figure/4-tr.png')
pyplot.show()
```

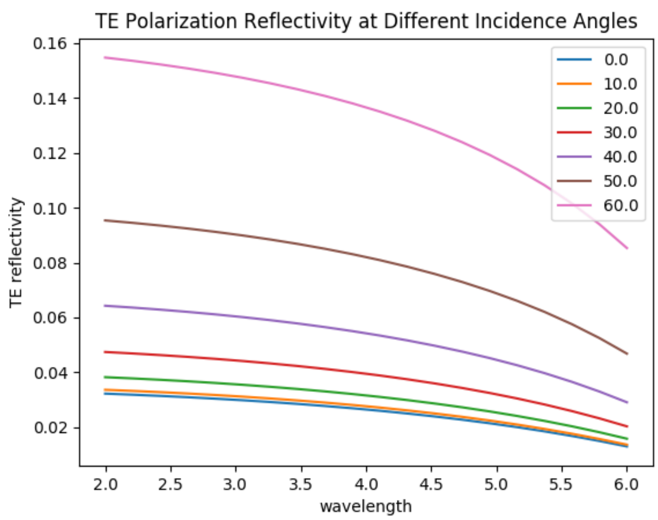


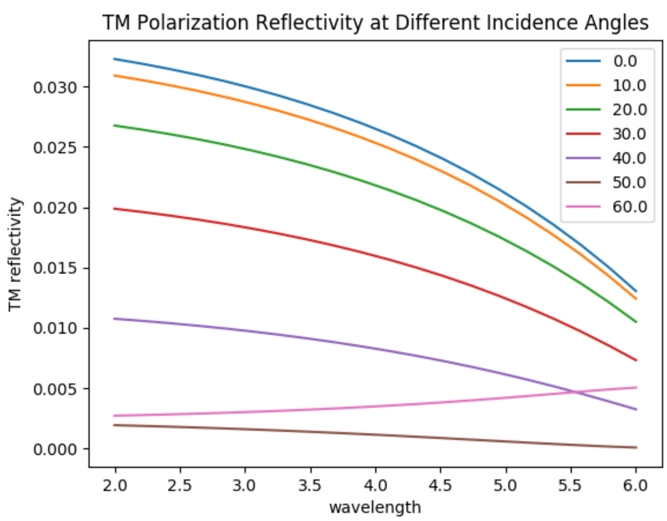


5

```python
from matplotlib import pyplot
import numpy
from numpy import cos, arcsin, sin
```

```python
def ganIndex(wavelength):
    return numpy.sqrt(
        0.6961663*wavelength**2 / (wavelength**2 - 0.0684043**2)
        + 0.4079426*wavelength**2 / (wavelength**2 - 0.1162414**2)
        + 0.8974794*wavelength**2 / (wavelength**2 - 9.896161**2)
        + 1).reshape(-1, 1)

def degreeArangeToRadians(start, stop, spacing):
    degreeRange = numpy.arange(start, stop+spacing, spacing)
    radianRange = numpy.deg2rad(degreeRange)
    radianRangeAsColumnVector = radianRange.reshape(-1, 1).T
    return radianRangeAsColumnVector

def snellsTransmissionAngle(eta1, eta2, a1):
    return arcsin(sin(a1)*(eta2/eta1))

def perpendicularReflection(eta1, eta2, a1, a2):
    return ((eta2*cos(a1) - eta1*cos(a2))
            / (eta2*cos(a1) + eta1*cos(a2)))**2

def parallelReflection(eta1, eta2, a1, a2):
    return ((eta2*cos(a2) - eta1*cos(a1))
            / (eta2*cos(a2) + eta1*cos(a1)))**2



wavelength = numpy.linspace(2, 6, 20)
n2 = ganIndex(wavelength)
eta1 = 1 #Air
eta2 = 1/n2 #by the relationship eta2/eta1 = n1/n2

incidenceAngles = degreeArangeToRadians(
    start = 0,
    stop = 60,
    spacing = 10)

transmissionAngles = snellsTransmissionAngle(
    eta1 = eta1,
    eta2 = eta2,
    a1 = incidenceAngles)

transverseElectricReflectivity = perpendicularReflection(
    eta1 = eta1,
    eta2 = eta2,
    a1 = incidenceAngles,
    a2 = transmissionAngles)

transverseMagneticReflectivity = parallelReflection(
    eta1 = eta1,
    eta2 = eta2,
    a1 = incidenceAngles,
    a2 = transmissionAngles)


pyplot.plot(wavelength, transverseElectricReflectivity)
pyplot.title('TE Polarization Reflectivity at Different Incidence Angles')
pyplot.xlabel('wavelength')
pyplot.ylabel('TE reflectivity')
pyplot.legend(*numpy.round(numpy.rad2deg(incidenceAngles)))
pyplot.savefig('figure/5-te.png')
pyplot.show()

pyplot.plot(wavelength, transverseMagneticReflectivity)
pyplot.title('TM Polarization Reflectivity at Different Incidence Angles')
pyplot.xlabel('wavelength')
pyplot.ylabel('TM reflectivity')
pyplot.legend(*numpy.round(numpy.rad2deg(incidenceAngles)))
pyplot.savefig('figure/5-tm.png')
pyplot.show()
```





6 - Brewster's Angle

```
from numpy import arctan, sqrt, rad2deg

wavelength = 580E-9
schottIndex = 1.5427
airIndex = 1
brewsterAngle = rad2deg(arctan(sqrt(schottIndex/airIndex)))

print(f'\[\\theta_B'
      f' = \\tan^{{-1}}\sqrt{{\dfrac{{\epsilon_2}}{{\epsilon_1}}}}'
      f' = {brewsterAngle:.2f}^{{\circ}}\]')
```

$$\theta_B = \tan^{-1}\sqrt{\dfrac{\epsilon_2}{\epsilon_1}} = 51.16^{\circ}$$

Supplemental

1 - Waveguides

`phaseVelocity`: The speed of light, since the guide is hollow.
`f`: Operating frequency.
`f10`: Dominant Mode cutoff frequency.
`f01`: Second Mode cutoff frequency.

```
import numpy
from numpy import sqrt, pi

phaseVelocity = 3E8

f = 4E9
f10 = (1-0.12)*f #dominant mode
f01 = (1+0.15)*f #second mode

a = phaseVelocity/(2*f10)
b = phaseVelocity/(2*f01)

print(f'\[a = {a*100:.2f}\si{{\cm}}\]')
print(f'\[b = {b*100:.2f}\si{{\cm}}\]')
```

$$a = 4.26\text{cm}$$

$$b = 3.26\text{cm}$$

2 - Resonant Cavities

```
from numpy import sqrt, pi

a, b, d = (6.4E-2, 4E-2, 5E-2)

#Quality Factor
mu0 = 4*pi*1E-7
sigmaC = 5.8E7
phaseVelocity = 3E8
m, n, p = (1, 0, 1)
f101 = phaseVelocity/2 * sqrt((m/a)**2 + (n/b)**2 + (p/d)**2)
skinDepth = 1/sqrt(pi*f101*mu0*sigmaC)
Q = 1/skinDepth * a*b*d*(a**2+d**2)/(a**3*(d+2*b) + d**3*(a+2*b))
print(f'Q = {Q:.0f}\n')


#Sorted Modes
cutoffs = []
for i in range(0,2):
    for j in range(0,2):
        for k in range(0,2):
            cutoff = phaseVelocity/2 * sqrt((i/a)**2 + (j/b)**2 + (k/d)**2)
            cutoffs.append((f'{i}{j}{k}', cutoff))

cutoffs = sorted(cutoffs, key=lambda pair: pair[1])
for mode, cutoff in cutoffs:
    print(f'Mode {mode} with cutoff {cutoff:.2E} Hz')
```

```
Q = 15136

Mode 000 with cutoff 0.00E+00 Hz
Mode 100 with cutoff 2.34E+09 Hz
Mode 001 with cutoff 3.00E+09 Hz
Mode 010 with cutoff 3.75E+09 Hz
Mode 101 with cutoff 3.81E+09 Hz
Mode 110 with cutoff 4.42E+09 Hz
Mode 011 with cutoff 4.80E+09 Hz
Mode 111 with cutoff 5.34E+09 Hz
```

Answer: The Quality factor is **15136**. Also, I've listed the sorted modes (by cutoff frequency). The first four lowest-order modes (ignoring mode 000), are: **100, 001, 010, 101**.