In [1]:

```
from math import sin, cos, pi
from matplotlib import pyplot as plt
import numpy as np

d_to_r = pi/180
LINE = 6
ANGLE = [0, 9,18,27,36, 45]
SYMMETRY = [(-1, 1), (-1, -1), (1, -1)]

def polar_to_xy(polar):
    coor = []
    for i in range(LINE):
        x = cos(ANGLE[i] * d_to_r) * polar[i]
        y = sin(ANGLE[i] * d_to_r) * polar[i]
        coor.append([x, y])
    for i in range(LINE-1, -1, -1):
        coor.append([coor[i][1], coor[i][0]])
    quarter = 1
    for dx, dy in SYMMETRY:
        if quarter%2 == 1:
            for i in range(LINE*2 - 1, -1, -1):
                coor.append([coor[i][0]*dx, coor[i][1]*dy])
        else:
            for i in range(LINE*2):
                coor.append([coor[i][0]*dx, coor[i][1]*dy])
        quarter += 1
    return coor

def spectrum_generator(shape):
    vertices = [mp.Vector3(shape[0][0], shape[0][1])]
    for i in range(1, len(shape) - 1):
        # eliminate duplicate point
        if abs(shape[i][0] - shape[i-1][0]) < 1e-5 and abs(shape[i][1] - shap
            continue
        vertices.append(mp.Vector3(shape[i][0], shape[i][1]))
        print(shape[i])
    # calculate transmission
    return get_trans(vertices)
```

In [2]:
```
from matplotlib import pyplot as plt
import numpy as np
import math
import meep as mp
import cmath

shape_size = 48

sx, sy, sz = 1, 1, 4
h = 1.25
dpml = 0.5
b_m, c_m = 1.4, 3.54
res = 15
echo = 1000
cell_size = mp.Vector3(sx,sy,sz)
fcen = 0.5
df = 0.2
theta = math.radians(0)
nfreq = 200

# k with correct length (plane of incidence: XZ)
k = mp.Vector3(math.sin(theta),0,math.cos(theta)).scale(fcen)
def pw_amp(k, x0):
    def _pw_amp(x):
        return cmath.exp(1j * 2 * math.pi * k.dot(x + x0))
    return _pw_amp

def get_trans(vertices):
    geometry = [mp.Block(size = cell_size, material=mp.Medium(index=b_m)),
                mp.Prism(vertices,
                         height=h,
                         material=mp.Medium(index=c_m),
                         center=mp.Vector3()
                        )]
    pml_layers = [mp.PML(thickness=1, direction = mp.Z, side=mp.High),
                  mp.Absorber(thickness=1,direction = mp.Z, side=mp.Low)]
    src_pos = -(sz/2 - dpml - 0.5)
    src = [mp.Source(src = mp.GaussianSource(fcen, fwidth=df),
                     component = mp.Ey,
                     center = mp.Vector3(0,0,src_pos),
                     size = mp.Vector3(sx,sy,0),
                     amp_func=pw_amp(k,mp.Vector3(0,0,src_pos)))]
    sim = mp.Simulation(resolution=res,
                        cell_size=cell_size,
                        boundary_layers=pml_layers,
                        sources=src,
                        geometry=geometry,
                        k_point=k)
    freg = mp.FluxRegion(center=mp.Vector3(0,0,-src_pos),
                         size = mp.Vector3(sx,sy,0))
    trans = sim.add_flux(fcen, df, nfreq, freg)
    sim.run(until = echo)
    bend = mp.get_fluxes(trans)

    #get straight
    sim.reset_meep()
```

```
    geometry = [mp.Block(size = cell_size, material=mp.Medium(index=b_m))]
    pml_layers = [mp.PML(thickness= 1, direction = mp.Z, side=mp.High),
                  mp.Absorber(thickness=1,direction = mp.Z, side=mp.Low)]
    src = [mp.Source(src = mp.GaussianSource(fcen, fwidth=df),
                     component = mp.Ey,
                     center = mp.Vector3(0,0,src_pos),
                     size = mp.Vector3(sx,sy,0),
                     amp_func=pw_amp(k,mp.Vector3(0,0,src_pos)))]
    sim = mp.Simulation(resolution=res,
                        cell_size=cell_size,
                        boundary_layers=pml_layers,
                        sources=src,
                        geometry=geometry,
                        k_point=k)
    freg = mp.FluxRegion(center=mp.Vector3(0,0,-src_pos),
                         size = mp.Vector3(sx,sy,0))
    trans = sim.add_flux(fcen, df, nfreq, freg)
    sim.run(until = echo)
    straight = mp.get_fluxes(trans)
    flux_freqs = mp.get_flux_freqs(trans)
    sim.reset_meep()
    c = 300
    p = 0.6
    Ts = []
    for i in range(nfreq):
        Ts = np.append(Ts, bend[i]/straight[i])
    return np.multiply(flux_freqs, c/p),Ts
```

In [18]:

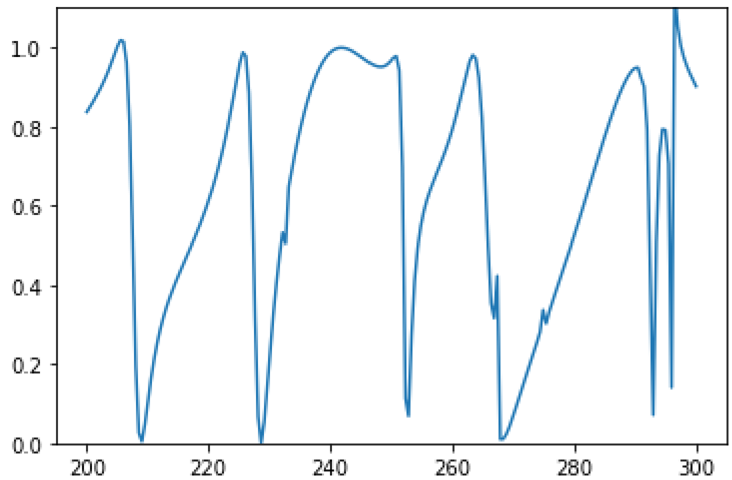
```
T_shape = [0.47552826, 0.48145578, 0.5,        0.51662739, 0.54836575, 0.6
P_shape = [0.49873263, 0.48748794, 0.48267844, 0.49323902, 0.49500504, 0.5012
```

In [19]:

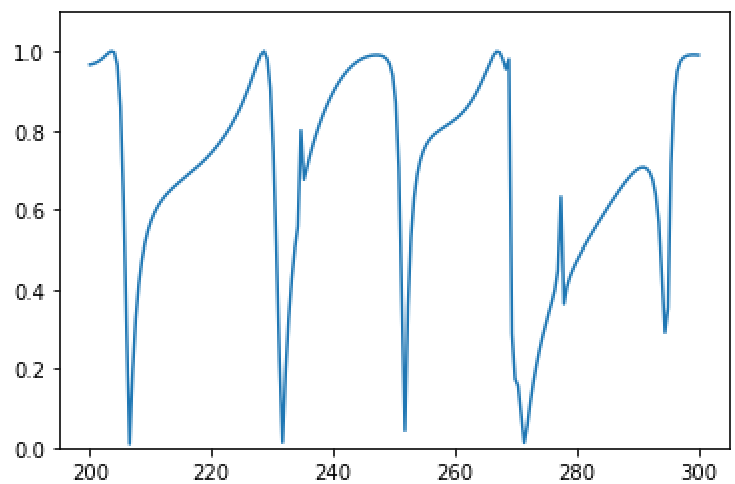
```
freq, Ts = spectrum_generator(polar_to_xy(P_shape))
```

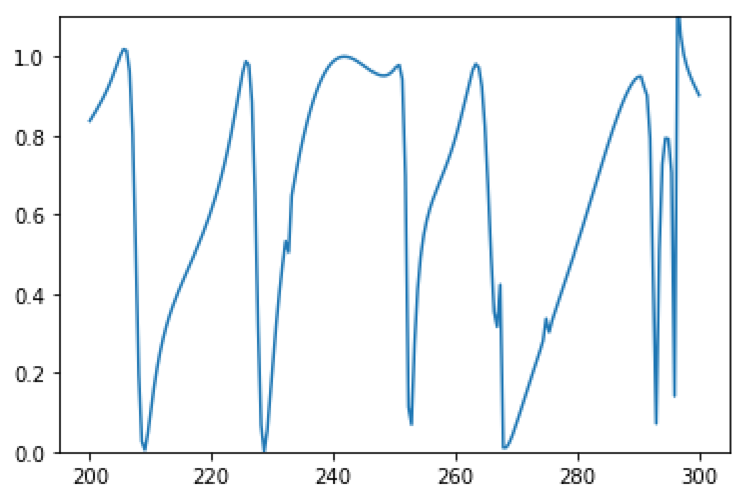
```
[0.4814861545187421, 0.07625991510746416]
[0.45905447563717927, 0.1491558407783884]
[0.43947918480427683, 0.22392582918084428]
[0.4004674896612506, 0.29095666232244577]
[0.35446806149900983, 0.3544680614990098]
[0.29095666232244577, 0.4004674896612506]
[0.22392582918084428, 0.43947918480427683]
[0.1491558407783884, 0.45905447563717927]
[0.07625991510746416, 0.4814861545187421]
[0.0, 0.49873263]
[-0.07625991510746416, 0.4814861545187421]
[-0.1491558407783884, 0.45905447563717927]
[-0.22392582918084428, 0.43947918480427683]
[-0.29095666232244577, 0.4004674896612506]
[-0.3544680614990098, 0.35446806149900983]
[-0.4004674896612506, 0.29095666232244577]
[-0.43947918480427683, 0.22392582918084428]
[-0.45905447563717927, 0.1491558407783884]
[-0.4814861545187421, 0.07625991510746416]
```

In [20]:
```
plt.ylim(0, 1.1)
plt.plot(freq, Ts)
plt.show()
```



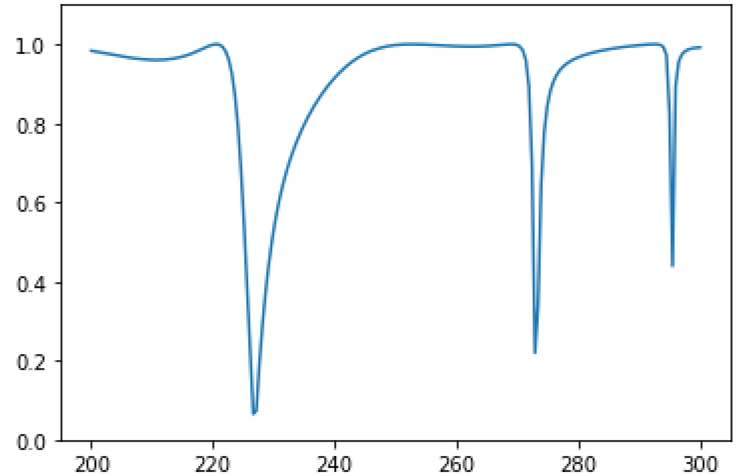
T_shape = [0.47552826, 0.48145578, 0.5, 0.51662739, 0.54836575, 0.6]
P_shape = [0.49873263, 0.48748794, 0.48267844, 0.49323902, 0.49500504, 0.50129354]

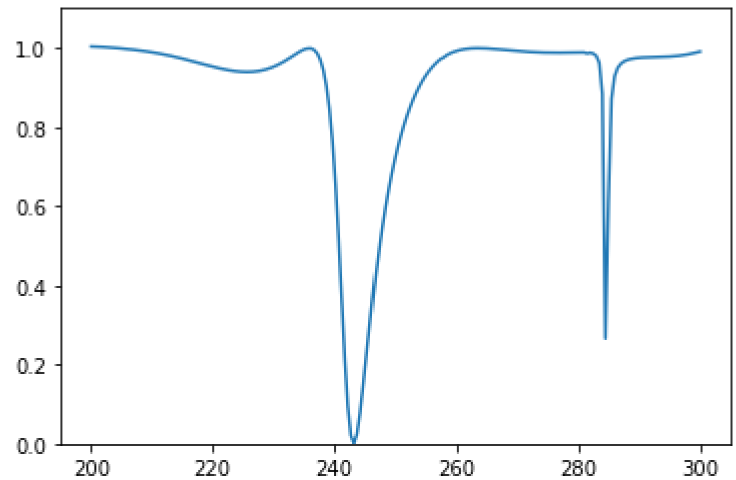




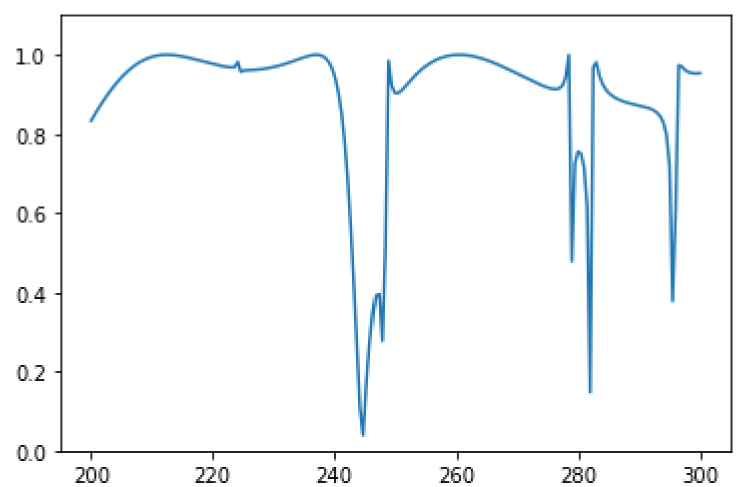
T_shape = [0.28531695, 0.28887347, 0.3, 0.2, 0.2370452, 0.3]

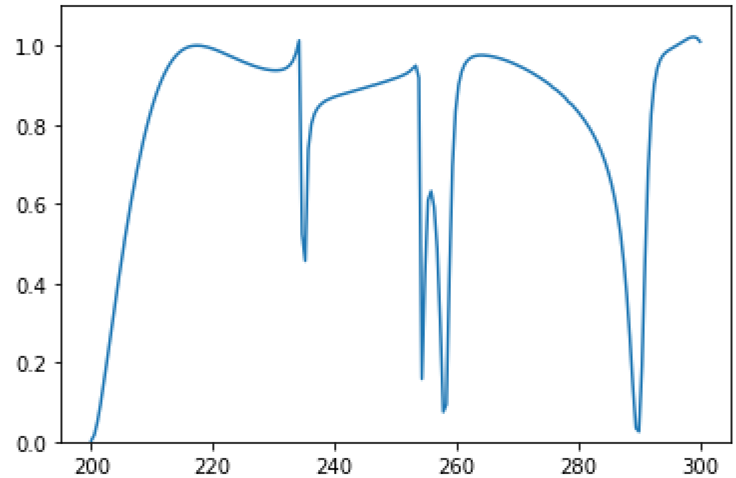
P_shape = [0.2604493, 0.25205013, 0.24875128, 0.22955596, 0.21976031, 0.2150829]



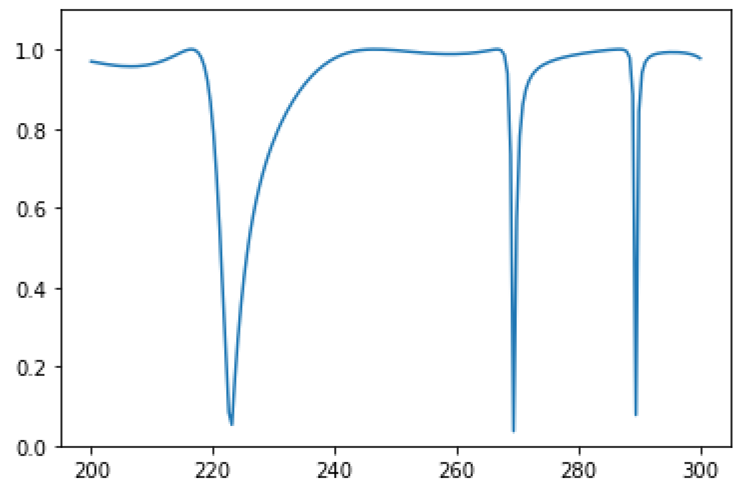


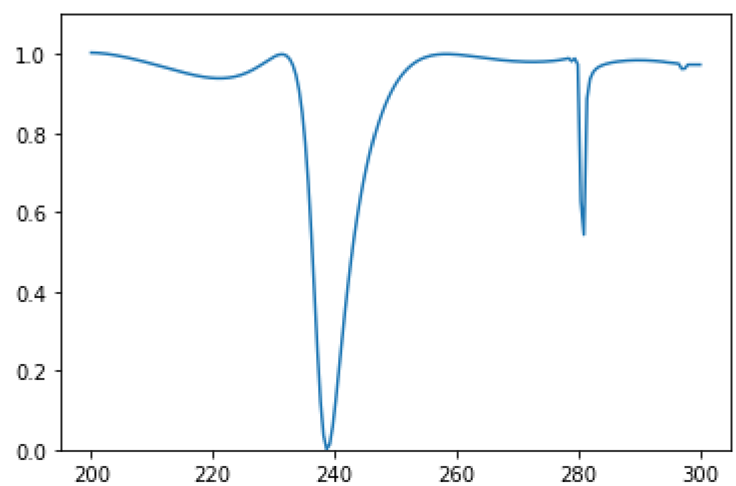
T_shape = [0.4, 0.39015067, 0.39015067, 0.4, 0.338636, 0.3]
P_shape = [0.3636202, 0.34087867, 0.325329 , 0.33002156, 0.35687244, 0.39777377]



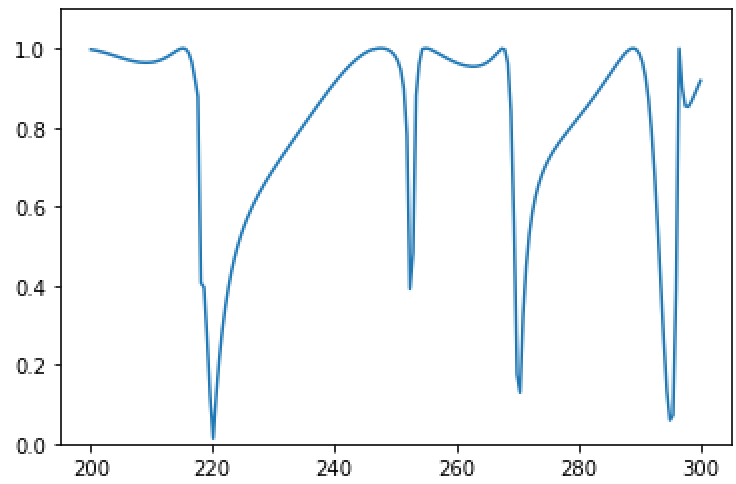


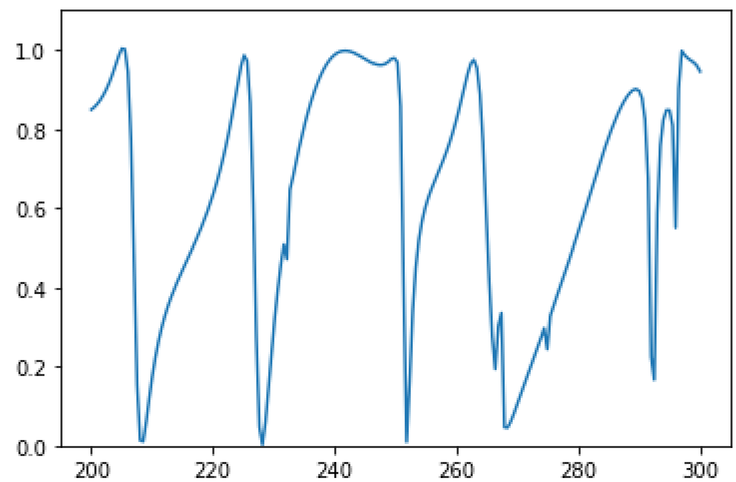
T_shape = [0.2963065, 0.3, 0.2370452, 0.2, 0.19753767, 0.2]
P_shape = [0.2783215, 0.26100606, 0.24721411, 0.23570469, 0.23045738, 0.23399867]



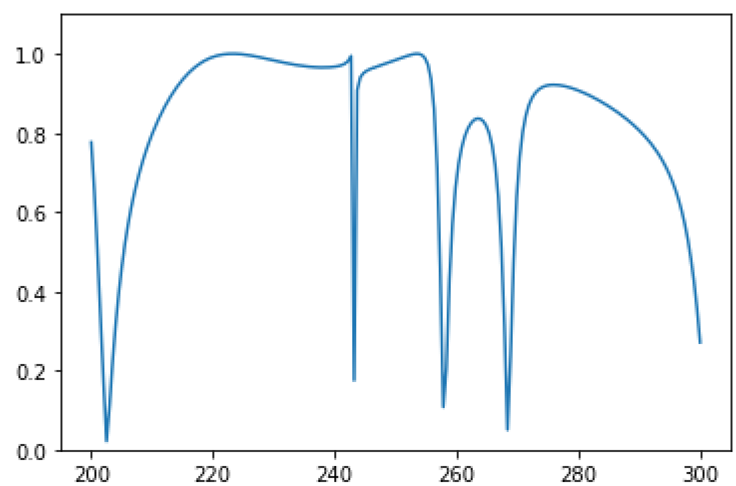


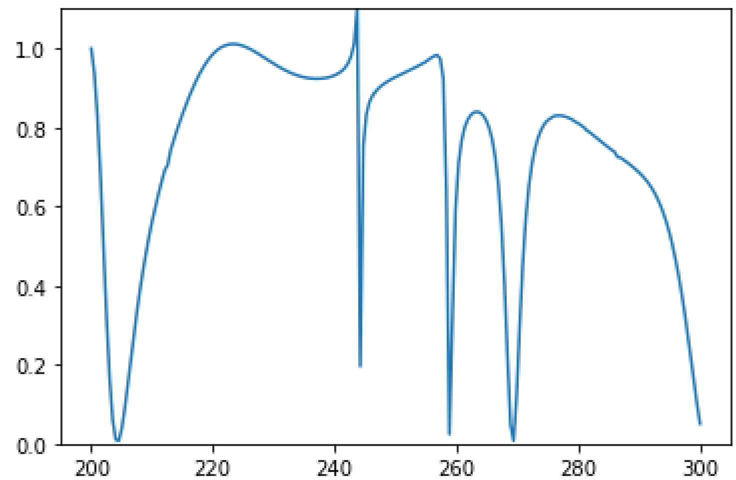
T_shape = [0.4, 0.41826582, 0.44988746, 0.5, 0.4389726, 0.4]
P_shape = [0.49462396, 0.4893507, 0.48621473, 0.4991828, 0.49169588, 0.49366987]





T_shape = [0.2963065, 0.3, 0.338636, 0.4, 0.338636, 0.3]
P_shape = [0.28571936, 0.31026313, 0.3503182, 0.3859476, 0.32345802, 0.28863484]




1.0
0.8
0.6
0.4
0.2
0.0
200
220
240
260
280
300