In [17]:

```
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 [18]:

```
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)


    straight = np.genfromtxt('data/straight.txt')
```

```
    sim.reset_meep()
    Ts = []
    for i in range(nfreq):
        Ts = np.append(Ts, bend[i]/straight[i])
    return Ts
```

In [19]:

```
prediction = np.genfromtxt('../prediction.txt')
freq = np.genfromtxt('data/freq.txt')
```

In [20]:

```
for i in range(0, len(prediction) - 2, 2):
    Ts = spectrum_generator(polar_to_xy(prediction[i]))
    freq = np.genfromtxt('data/freq.txt')
    print("Prediction from Tandem:")
    print(prediction[i])
    plt.ylim(0, 1.1)
    plt.plot(freq, Ts)
    plt.show()
    Ts = spectrum_generator(polar_to_xy(prediction[i + 1]))
    freq = np.genfromtxt('data/freq.txt')
    print("Prediction from DFNN:")
    print(pridiction[i + 1])
    plt.ylim(0, 1.1)
    plt.plot(freq, Ts)
    plt.show()
```

```
<ipython-input-20-4bc614ec4bde> in <module>
      1 for i in range(0, len(prediction) - 2, 2):
----> 2     Ts = spectrum_generator(polar_to_xy(prediction[i]))
      3     freq = np.genfromtxt('data/freq.txt')
      4     print("Prediction from Tandem:")
      5     print(prediction[i])

<ipython-input-17-0a707849724a> in spectrum_generator(shape)
     36     print(len(vertices))
     37     # calculate transmission
---> 38     return get_trans(vertices)
     39

<ipython-input-18-b52bab4580d2> in get_trans(vertices)
     50                            size = mp.Vector3(sx,sy,0))
     51     trans = sim.add_flux(fcen, df, nfreq, freg)
---> 52     sim.run(until = echo)
     53     bend = mp.get_fluxes(trans)
     54
```

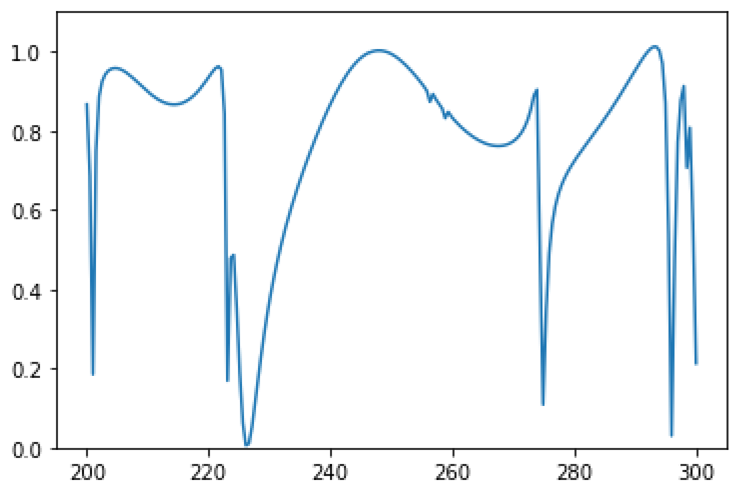
In [87]:

```
T_shape = [0.5, 0.4, 0.4, 0.4, 0.5, 0.4     ]
P_shape = [0.39288008, 0.41066885, 0.49765998, 0.4136067, 0.40488076, 0.4802
```

In [88]:
```
Ts = spectrum_generator(polar_to_xy(P_shape))
```

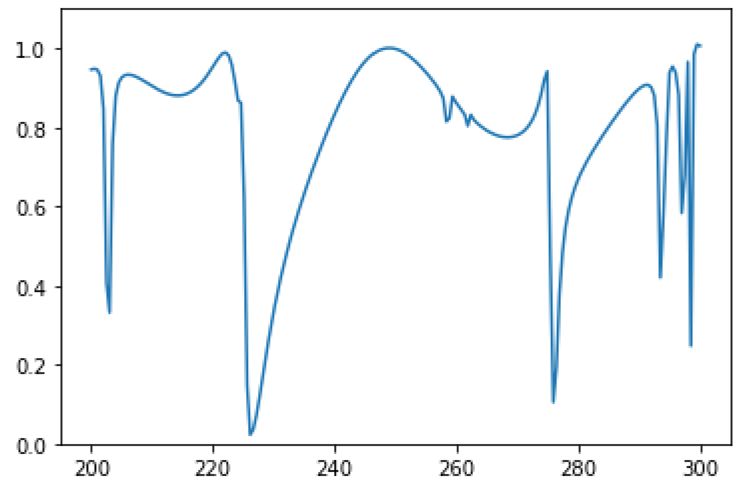
```
40
-----------
Initializing structure...
Meep: using complex fields.
Meep progress: 15.733333333333333/1000.0 = 1.6% done in 4.0s, 250.4s to g
o
Meep progress: 31.7/1000.0 = 3.2% done in 8.0s, 244.6s to go
Meep progress: 45.833333333333336/1000.0 = 4.6% done in 12.0s, 250.2s to
go
Meep progress: 61.0/1000.0 = 6.1% done in 16.0s, 246.6s to go
Meep progress: 76.1/1000.0 = 7.6% done in 20.0s, 243.2s to go
Meep progress: 91.56666666666666/1000.0 = 9.2% done in 24.0s, 238.5s to g
o
Meep progress: 106.23333333333333/1000.0 = 10.6% done in 28.0s, 235.9s to
go
Meep progress: 120.66666666666667/1000.0 = 12.1% done in 32.0s, 233.6s to
go
Meep progress: 135.33333333333334/1000.0 = 13.5% done in 36.1s, 230.3s to
go
```

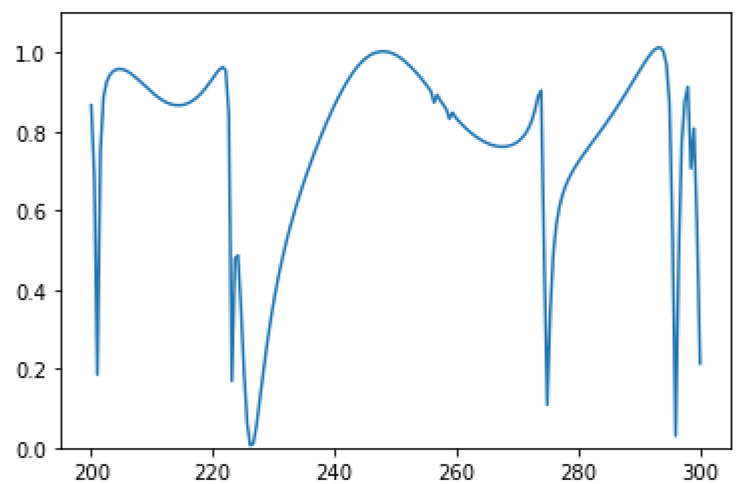
In [89]:
```
freq = np.genfromtxt('data/freq.txt')
plt.ylim(0, 1.1)
plt.plot(freq, Ts)
plt.show()
```



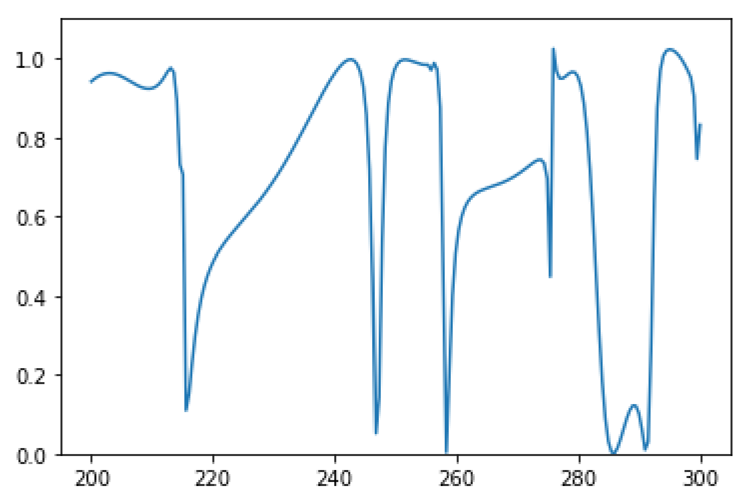
Four Peak

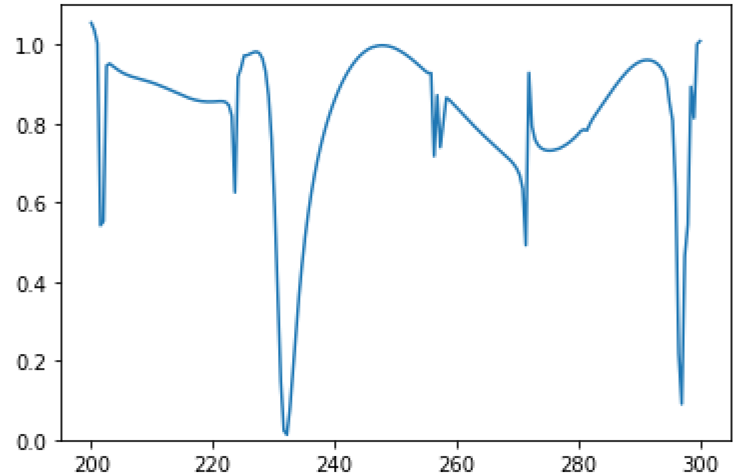
T_shape = [0.5, 0.4, 0.4, 0.4, 0.5, 0.4]
P_shape = [0.39288008, 0.41066885, 0.49765998, 0.4136067, 0.40488076, 0.48022035]



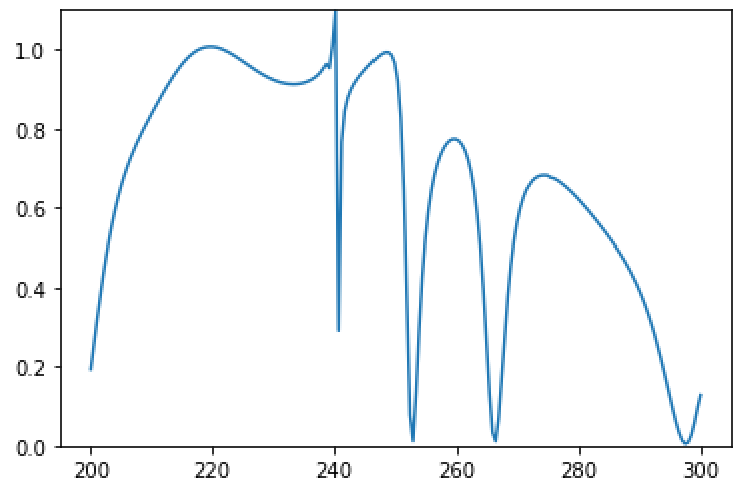


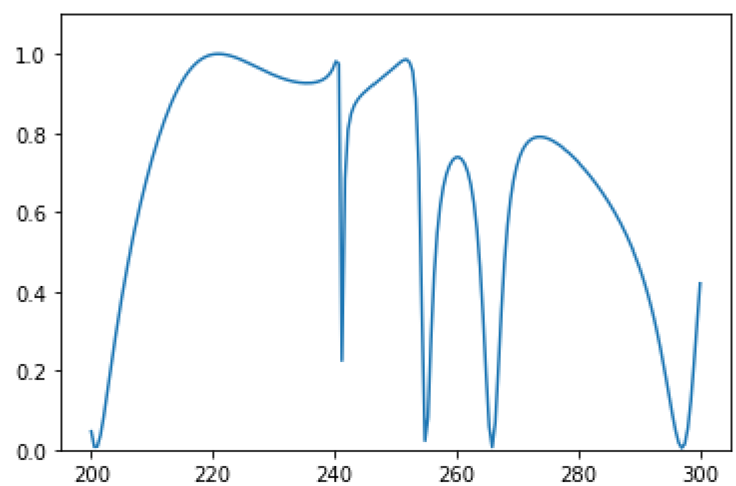
T_shape = [0.49384417, 0.5, 0.5, 0.4389726, 0.4, 0.3]
P_shape = [0.3724088, 0.4390575, 0.37945476, 0.4336163, 0.4642915, 0.50040674]



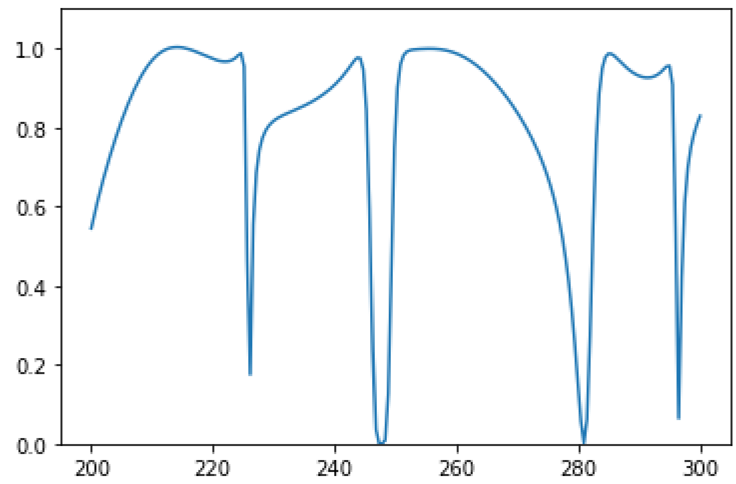


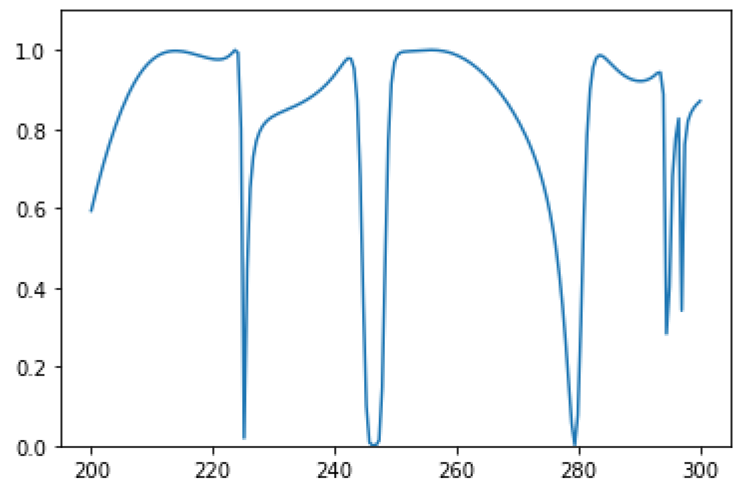
T_shape = [0.3, 0.338636, 0.4, 0.4, 0.3, 0.4]
P_shape = [0.39551848, 0.28155077, 0.40026873, 0.31739536, 0.3657673, 0.3019538]



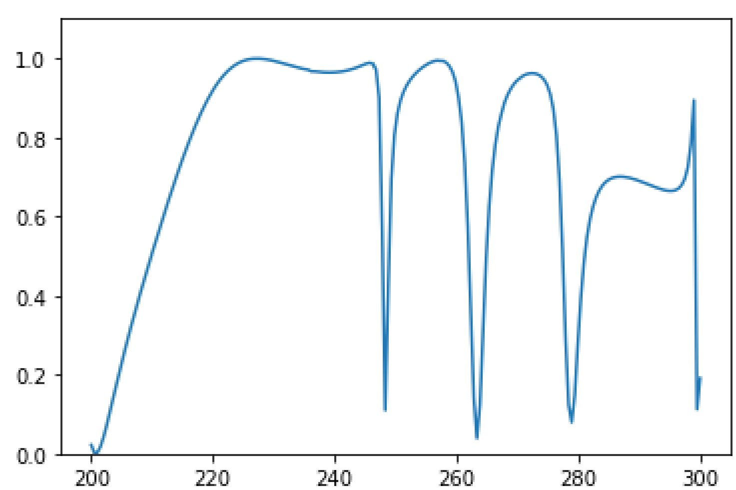


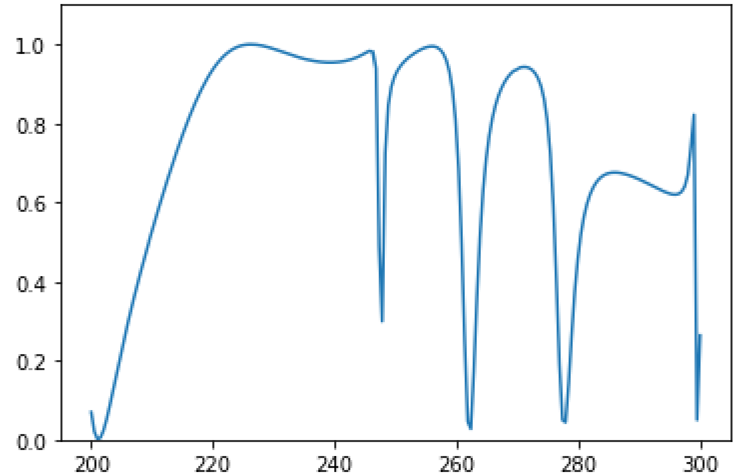
T_shape = [0.39507534, 0.4, 0.4, 0.3, 0.338636, 0.4]
P_shape = [0.39159143, 0.4019267, 0.39261207, 0.40591186, 0.316923, 0.31880692]





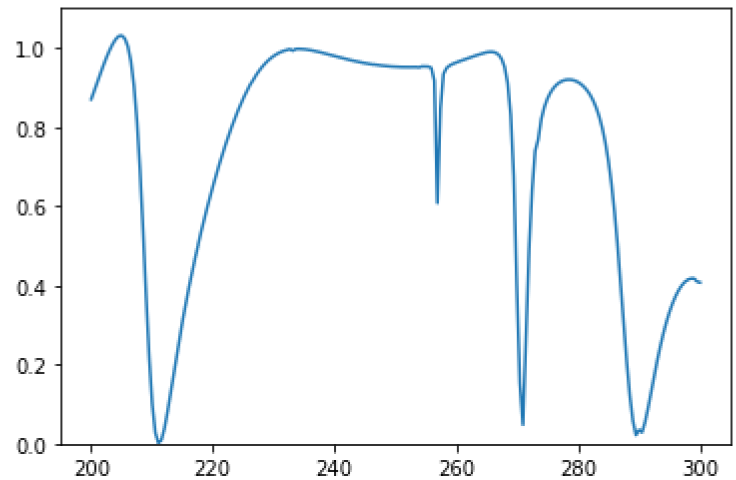
T_shape = [0.2, 0.3, 0.3, 0.338636, 0.4, 0.5]
P_shape = [0.3195423, 0.22433162, 0.28953445, 0.3793049, 0.39726943, 0.48038176]

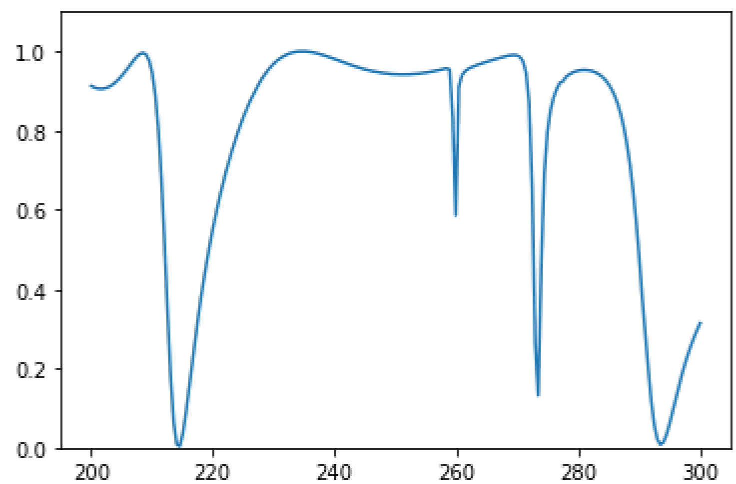




Three peak

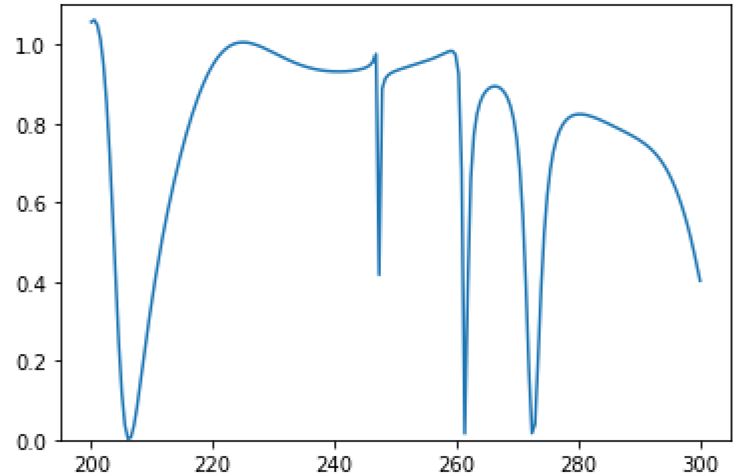
T_shape = [0.2, 0.3, 0.4, 0.3, 0.3, 0.3]
P_shape = [0.28688523, 0.27381307, 0.29683, 0.42090017, 0.26823282, 0.22643891]

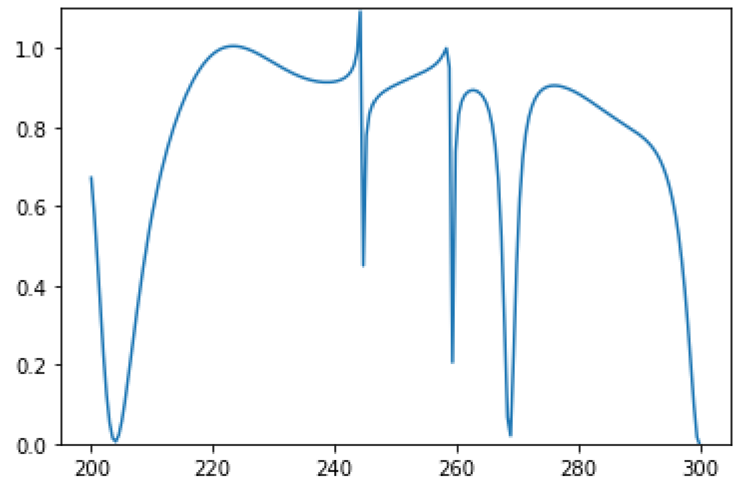




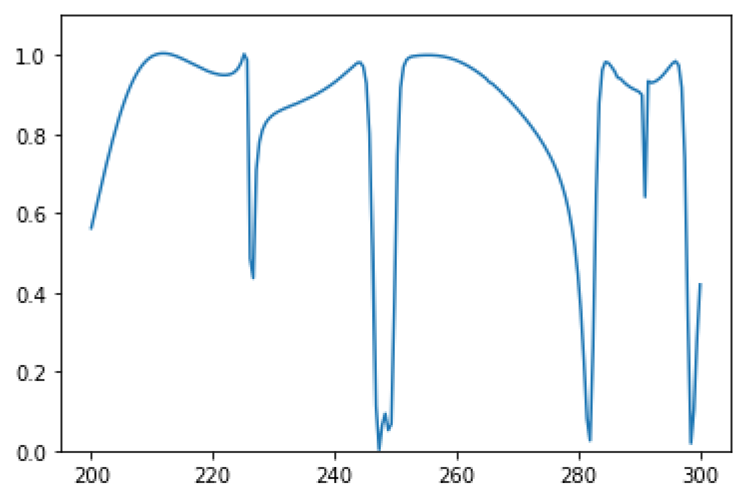
T_shape = [0.28531695, 0.28887347, 0.3, 0.4, 0.338636, 0.3]

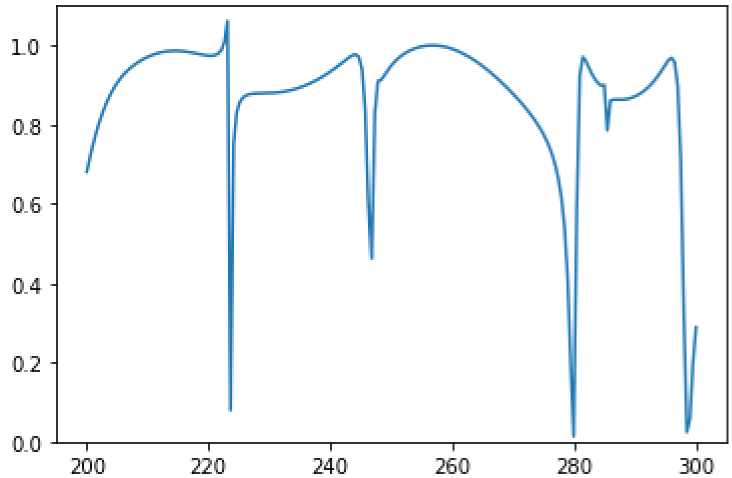
P_shape = [0.3102833, 0.3020802, 0.31310302, 0.33439282, 0.385919, 0.3369464]



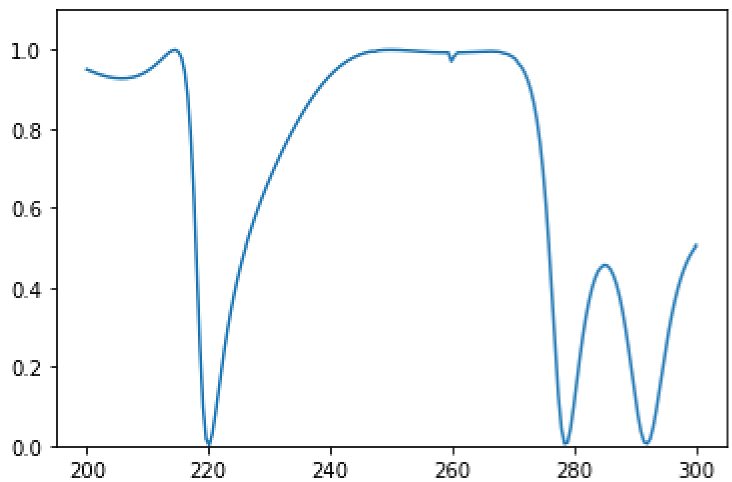


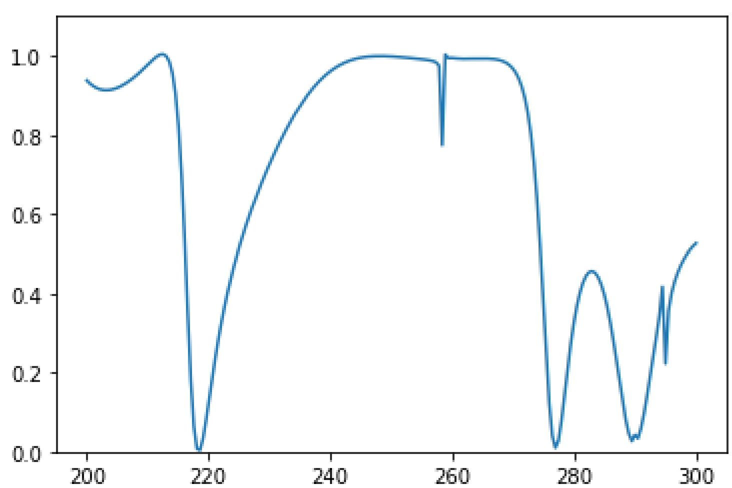
T_shape = [0.4, 0.3, 0.4, 0.4, 0.38516462, 0.38042261]
P_shape = [0.3134778, 0.31828544, 0.39894003, 0.38048273, 0.4231855, 0.41256857]



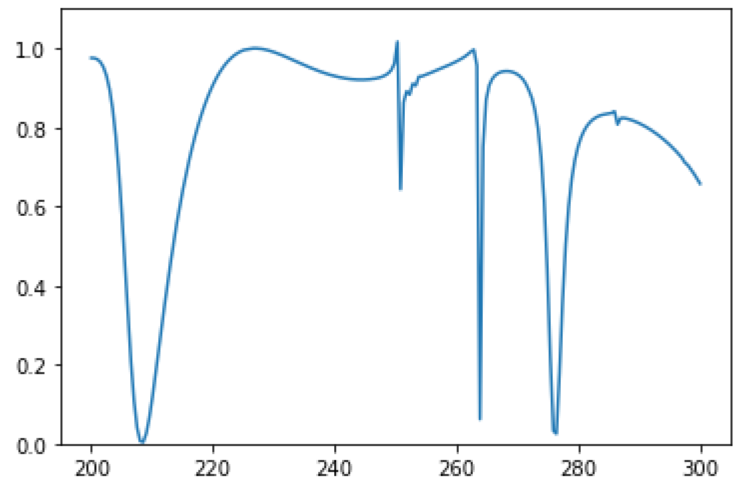


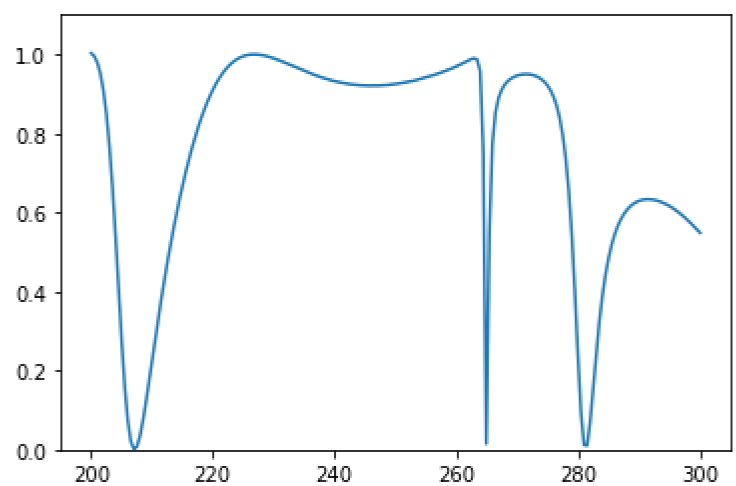
T_shape = [0.3, 0.338636, 0.4, 0.3, 0.2, 0.1]
P_shape = [0.31730723, 0.3913097, 0.34715515, 0.30192134, 0.2105813, 0.11027607]



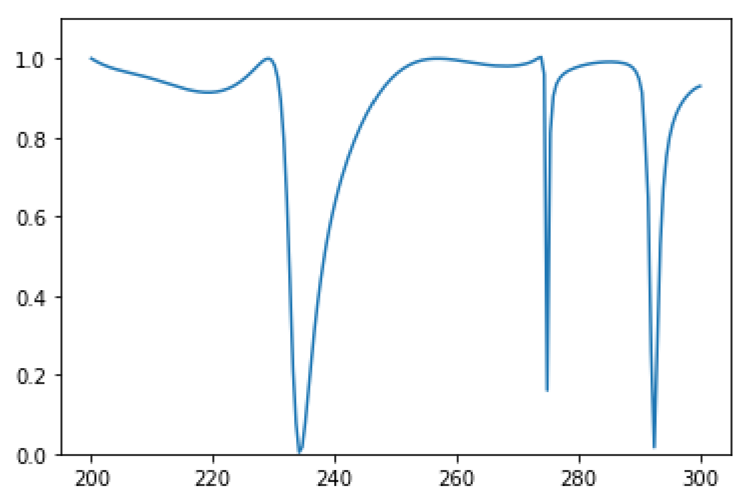


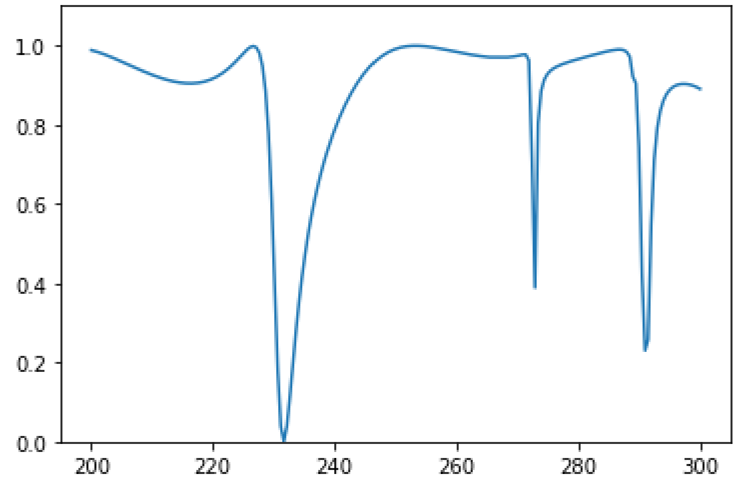
T_shape = [0.3, 0.2963065, 0.3, 0.3, 0.4, 0.3]
P_shape = [0.2947067, 0.29308522, 0.3010015, 0.28868848, 0.39579213, 0.36385483]



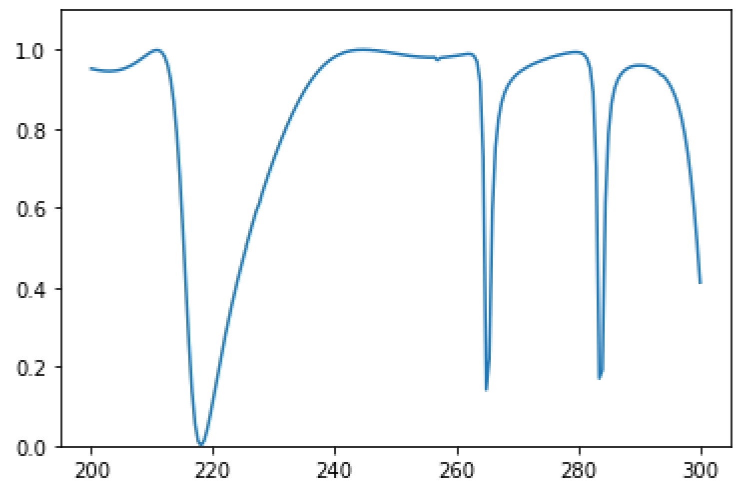


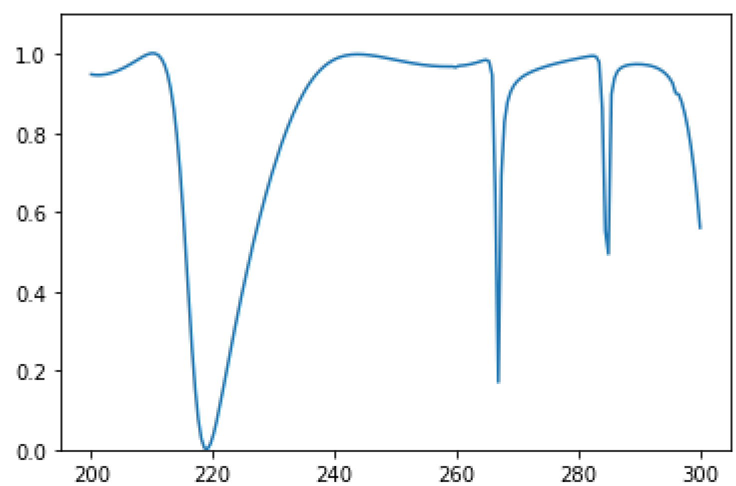
T_shape = [0.4, 0.3, 0.2370452, 0.2, 0.2, 0.3]
P_shape = [0.40134832, 0.29707843, 0.20395061, 0.30986106, 0.20724583, 0.21303698]





T_shape = [0.2, 0.2, 0.3, 0.4, 0.3, 0.3]
P_shape = [0.22393832, 0.19892702, 0.3096208, 0.35105002, 0.30258188, 0.34191588]

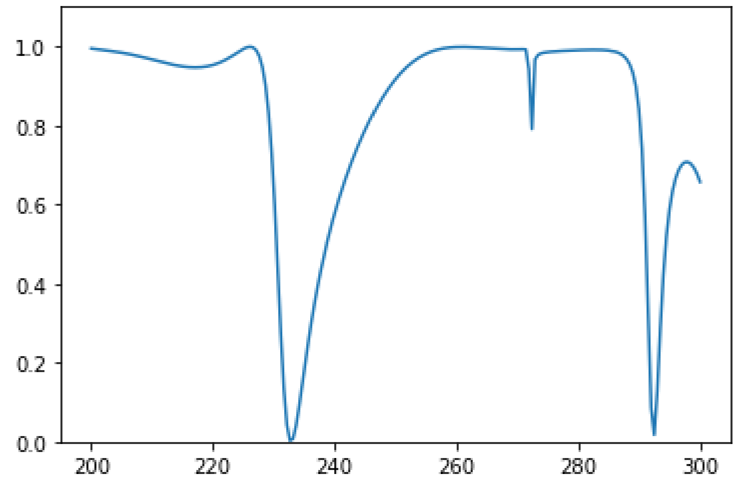


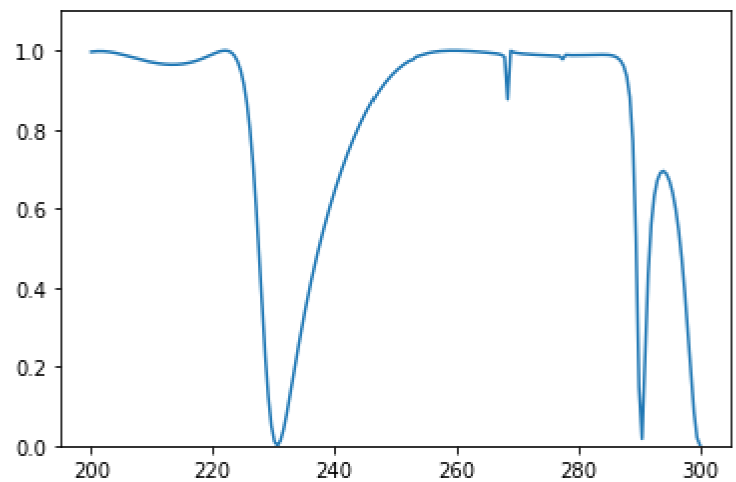


Two Peak

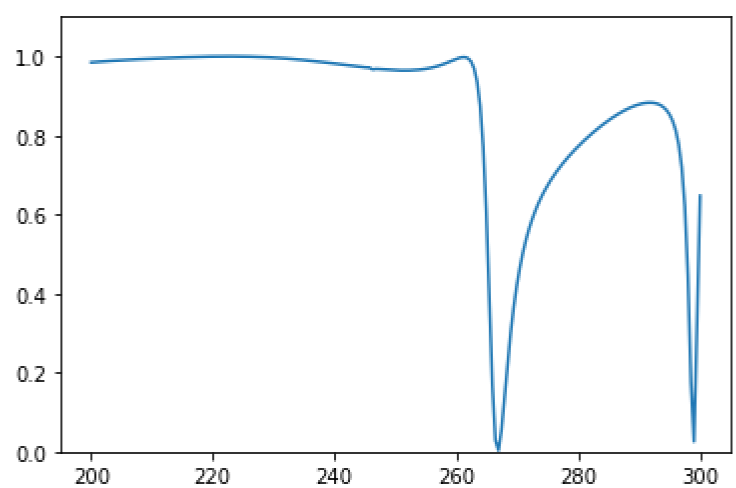
T_shape = [0.4, 0.3, 0.3, 0.2370452, 0.2, 0.1]

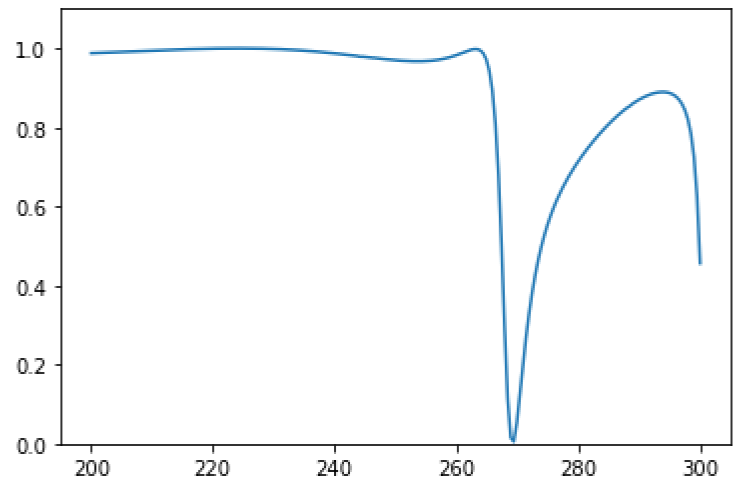
P_shape = [0.3282399, 0.30413926, 0.31323737, 0.29685694, 0.19540852, 0.07528802]



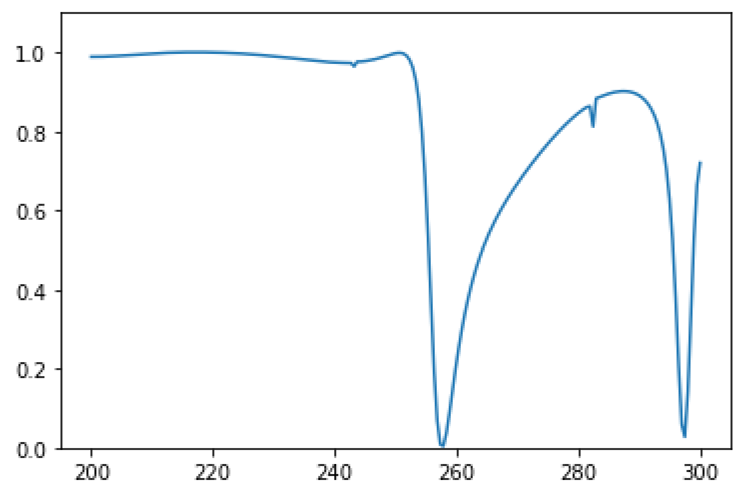


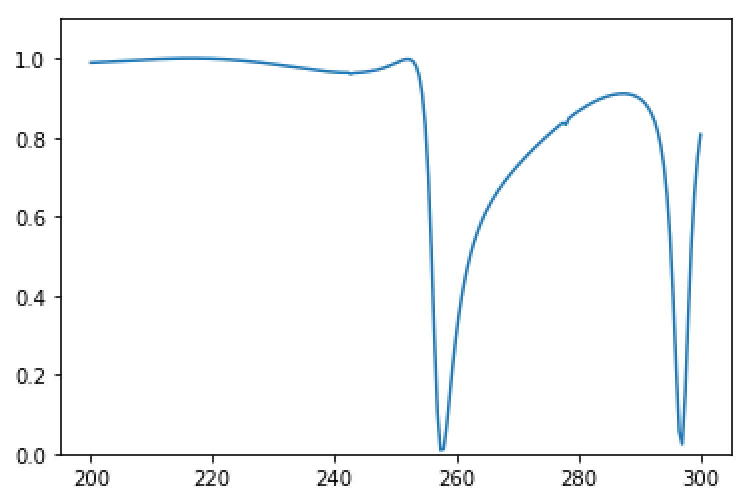
T_shape = [0.1, 0.1, 0.2, 0.3, 0.2370452, 0.2]
P_shape = [0.191461, 0.22202569, 0.2837429, 0.21161926, 0.12629329, 0.14049928]



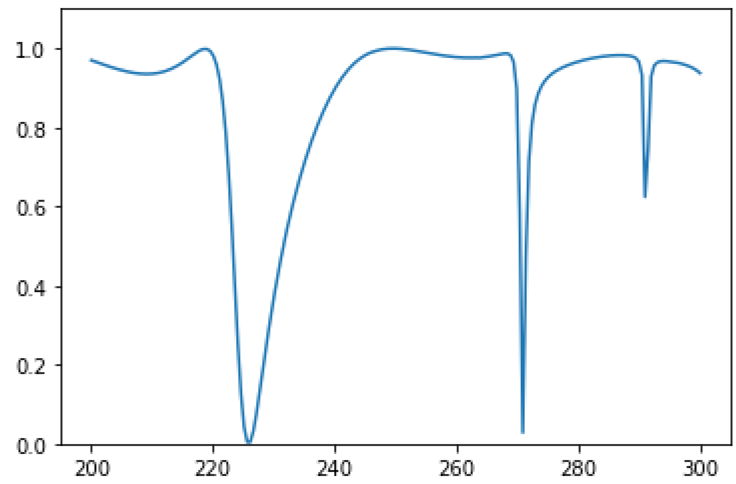


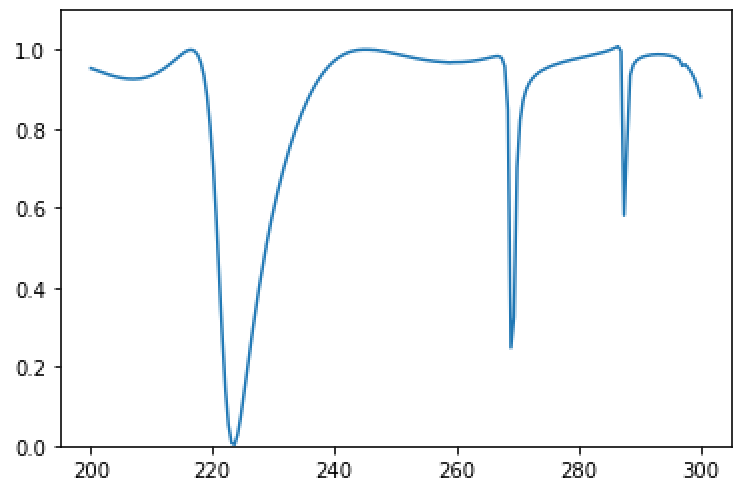
T_shape = [0.1, 0.13169178, 0.2, 0.2, 0.3, 0.3]
P_shape = [0.1342206, 0.12316048, 0.11376825, 0.20642188, 0.2912258, 0.40865904]



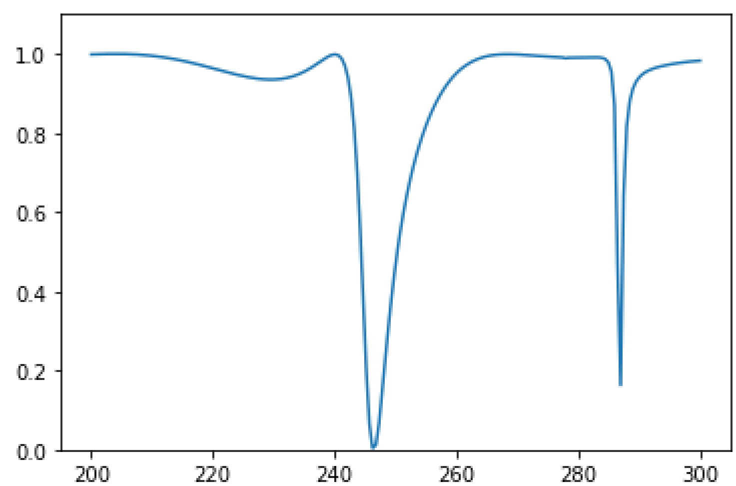


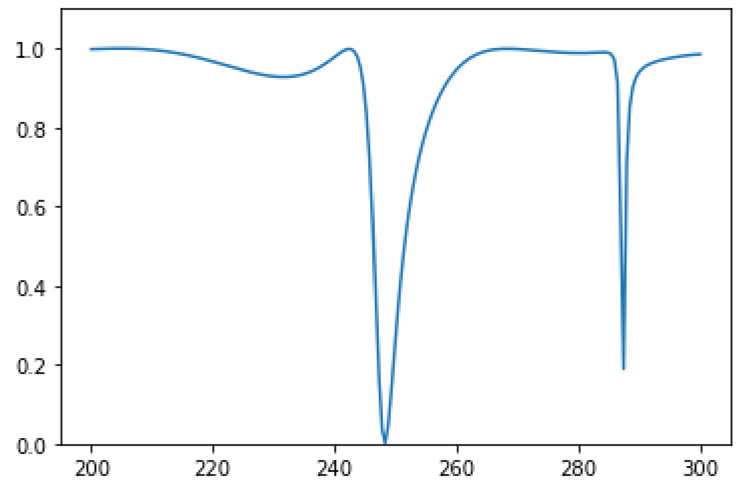
T_shape = [0.2, 0.19753767, 0.2, 0.3, 0.338636, 0.4]
P_shape = [0.26524678, 0.20846984, 0.2208249, 0.29262394, 0.32078624, 0.4105975]



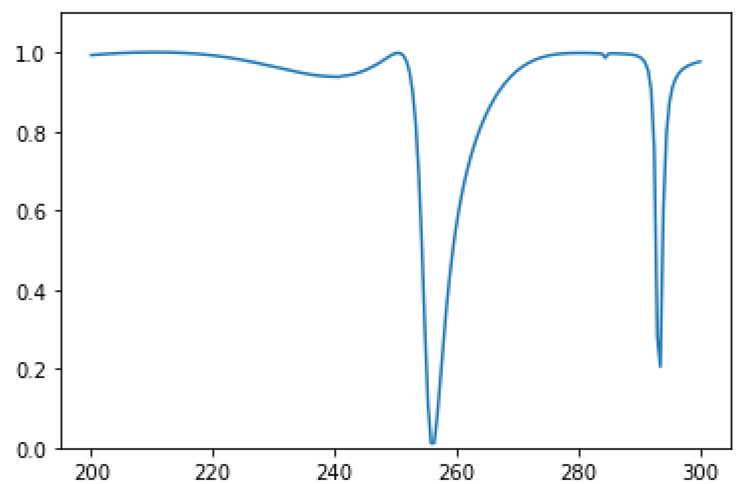


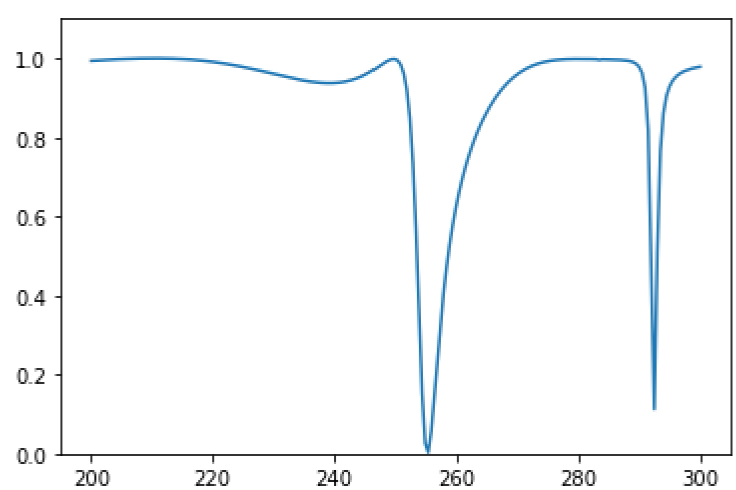
T_shape = [0.3, 0.2, 0.2, 0.21968701, 0.25051498, 0.3]
P_shape = [0.19265735, 0.20528206, 0.31247187, 0.21626106, 0.23505212, 0.23231103]





T_shape = [0.3, 0.2370452, 0.2, 0.2, 0.19753767, 0.2]
P_shape = [0.30729768, 0.23584536, 0.1915276, 0.19995084, 0.20841531, 0.20474266]

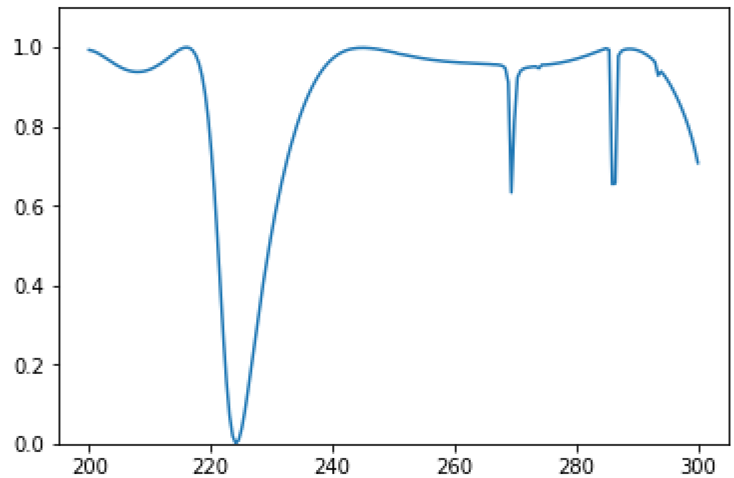


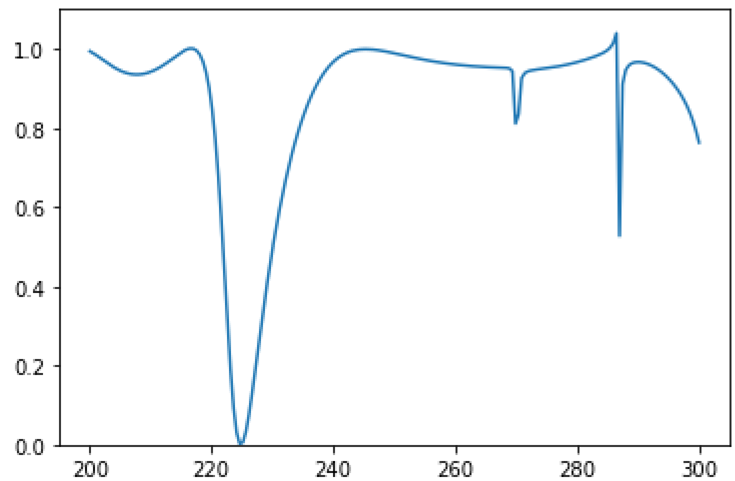


One peak

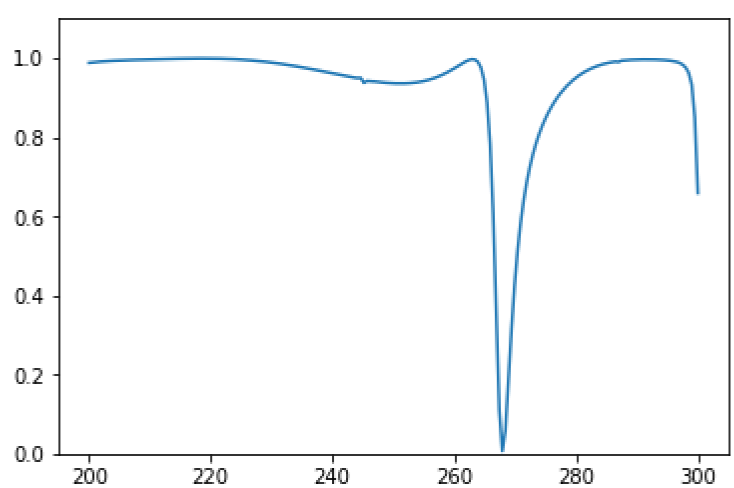
T_shape = [0.2, 0.3, 0.2963065, 0.3, 0.3, 0.2]

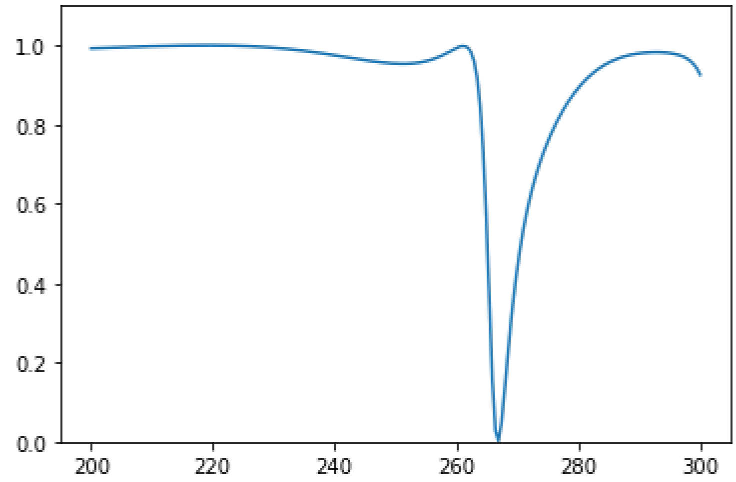
P_shape = [0.30232832, 0.29570624, 0.29942632, 0.19961643, 0.3002607, 0.29632616]



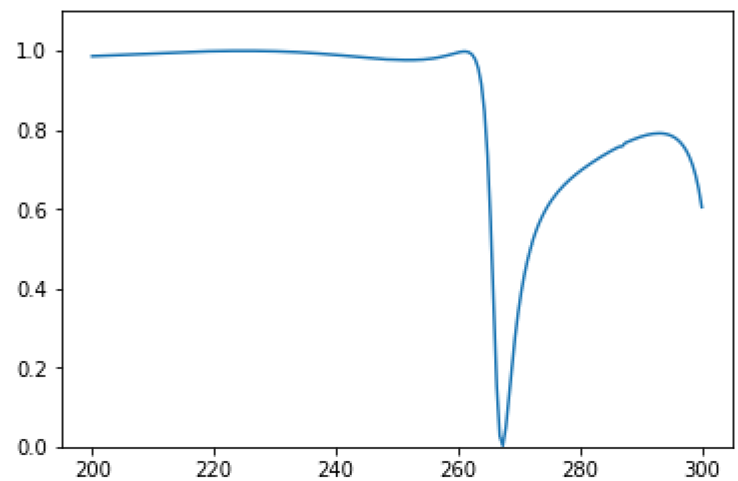


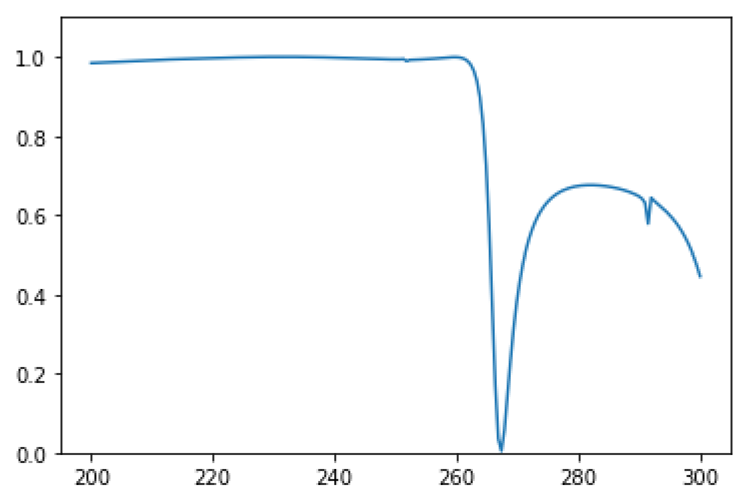
T_shape = [0.2, 0.1, 0.2, 0.2, 0.3, 0.2]
P_shape = [0.19365516, 0.18154019, 0.18587601, 0.20272604, 0.21093479, 0.24012172]



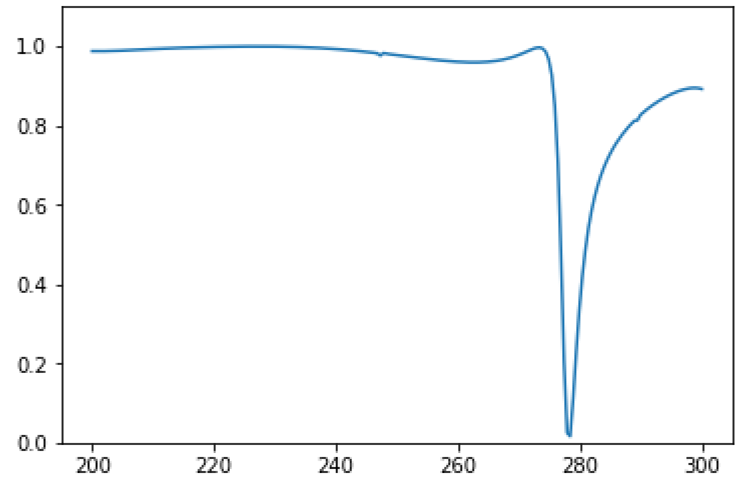


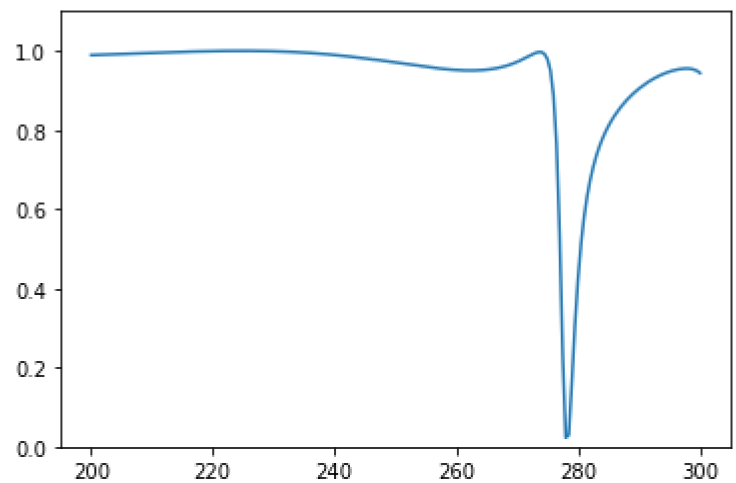
T_shape = [0.09876883, 0.1, 0.13169178, 0.2, 0.3, 0.2963065]
P_shape = [0.06425053, 0.07707888, 0.15710208, 0.22385162, 0.2692031, 0.31148475]





T_shape = [0.2, 0.13169178, 0.1, 0.2, 0.2370452, 0.3]
P_shape = [0.2710782, 0.22791114, 0.21213835, 0.1409609, 0.12369569, 0.17378637]


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


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