In [17]:

```
#%matplotlib notebook
import os
import sys
sys.path.append("numpy_path")
import numpy as np
import struct
from matplotlib import pyplot as plt
import keras
from keras.models import Sequential, load_model
from keras.layers import Dense, Dropout, Activation
from keras.optimizers import RMSprop
import keras.callbacks as cb
from keras.callbacks import EarlyStopping, ModelCheckpoint
from math import cos, sin, pi
from statistics import mean
import os.path
import math
shape_size = 48
```

In [18]:

```
# define loss  history
class LossHistory(cb.Callback):
    def on_train_begin(self, logs={}):
        self.losses = []
    def on_batch_end(self, batch, logs={}):
        batch_loss = logs.get('loss')
        self.losses.append(batch_loss)

#plot losses
def plot_losses(losses):
    plt.plot(losses)
    plt.title('Loss per batch')
    plt.show()

def feature_scaling(X):
    X = X.T
    for i in range(7):
        mean = X[i].mean()
        std = X[i].std()
        X[i] = [(x - mean)/std for x in X[i]]
    return X.T

# input dimension
in_dim = 6
out_dim = 200
def init_model():
    model = Sequential()
    model.add(Dense(20, input_dim=in_dim))
    model.add(Dropout(0.2))
    model.add(Activation('relu'))
    model.add(Dense(500))
    model.add(Dropout(0.5))
    model.add(Activation('relu'))
    model.add(Dense(500))
    model.add(Dropout(0.5))
    model.add(Activation('relu'))
    model.add(Dense(200))
    model.add(Dropout(0.5))
    model.add(Activation('relu'))
    model.add(Dense(200))
    model.add(Dropout(0.2))
    model.add(Activation('relu'))
#     model.add(Dense(70))
#     model.add(Dropout(0.2))
#     model.add(Activation('relu'))
#     model.add(Dense(100))
#     model.add(Activation('relu'))
    model.add(Dense(out_dim))
    model.add(Activation('sigmoid'))
    # use mean squared error to measure the looses
    model.compile(loss=keras.losses.mean_squared_error,
              optimizer=keras.optimizers.Adam(lr = 0.001),
              metrics=['accuracy'])
    return model
```

In [19]:

```
data_size = 0
dummy1 = [0]*200
dummy2 = [0]*6
SP = np.array(np.reshape(dummy1, (1, 200)))
SH = np.array(np.reshape(dummy2, (1, 6)))
for i in range(2, 65):
    path = 'meep_code/data/DATA'+str(i)
    if not os.path.exists(path):
        #miss.append(i)
        print('Missing batch:' + str(i))
        continue

    files = next(os.walk(path))[2] #dir is your directory path as string]
    num_data = len(files)
    data_size += num_data
    skip = []

    coordinates = np.genfromtxt('meep_code/data/DATA'+str(i)+'_sh.txt')
    xc, yc = coordinates[:, 0], coordinates[:, 1]
    xc = np.reshape(xc, (num_data, shape_size))
    yc = np.reshape(yc, (num_data, shape_size))

    for j in range(num_data):
        tmp = np.genfromtxt(path+'/'+'DATA'+str(i)+'_sp'+str(j)+'.txt')
        valid = True
        for q in range(200):
            if math.isnan(float(tmp[q])):
                print('Batch '+str(i)+'\tsample '+str(j)+' has NAN value')
                valid = False
                break
            if tmp[q] > 3:
                print('Batch '+str(i)+'\tsample '+str(j)+' has extreme value'
                valid = False
                break
        if not valid:
            #skip.append(j)
            continue
        SP = np.concatenate((SP, np.reshape(tmp, (1, 200))))
        tmp = []
        for q in range(6):
            tmp.append(math.sqrt(xc[j][q]**2 + yc[j][q]**2))
        SH = np.concatenate((SH, np.reshape(np.array(tmp), (1, 6))))
    print('Batch '+str(i)+' has \t'+str(num_data))
print('Total # of data: ' + str(len(SH)))
```

```
Batch 45 has    35
Batch 46 has    100
Batch 47 has    100
Batch 48 has    287
Batch 49 has    13
Batch 50 has    37
Batch 51 has    37
Batch 52 has    106
Batch 53 has    35
Batch 54 has    100
Batch 55 has    100
Batch 56 has    287
```

```
Batch 57 has    35
Batch 58 has    100
Batch 59 has    100
Batch 60 has    287
Batch 61 has    95
Batch 62 has    272
Batch 63 has    272
Batch 64 has    781
```

In [27]:

```
distribution = []
print('Total # of data: ' + str(len(SP)))
x = np.genfromtxt('meep_code/data/SP_xaxis.txt')
SP_F, SH_F = np.reshape(SP[1], (1, 200)),np.reshape(SH[1], (1, 6))
for i in range(2, len(SP)):
    peak = 0
    p_index = 0
    p_pos = [(0,0),(0,0)]
    for j in range(1, 200):
        if SP[i][j] < SP[i][p_index]:
            p_index = j
        if SP[i][j - 1] >= 0.6 >=SP[i][j]:
            peak += 1
            p_pos = [((j-1)/2+200,SP[i][j - 1]), (j/2+200, SP[i][j])]
    if peak == 1:
        distribution.append([1,p_index/2+200])
        SP_F = np.concatenate((SP_F, np.reshape(SP[i], (1, 200))))
        SH_F = np.concatenate((SH_F, np.reshape(SH[i], (1, 6))))
```

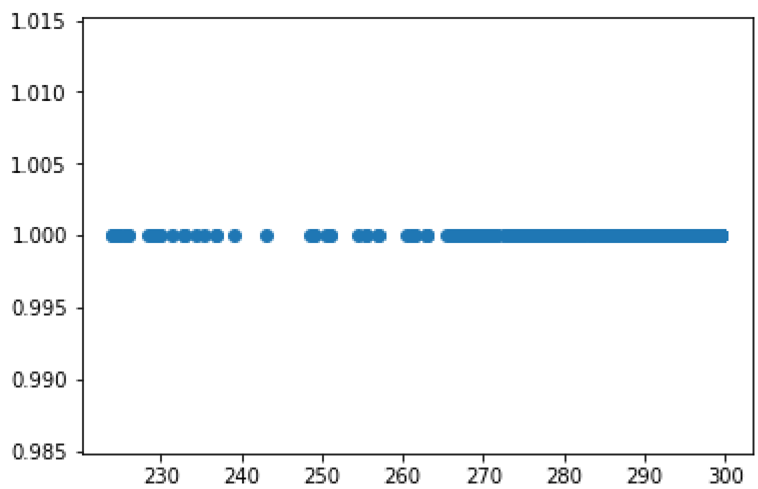
```
Total # of data: 4989
```

In [28]:

```
distribution = np.array(distribution)
plt.scatter(distribution[:,1],distribution[:,0])
```

Out[28]:

```
<matplotlib.collections.PathCollection at 0x17701ee0a90>
```

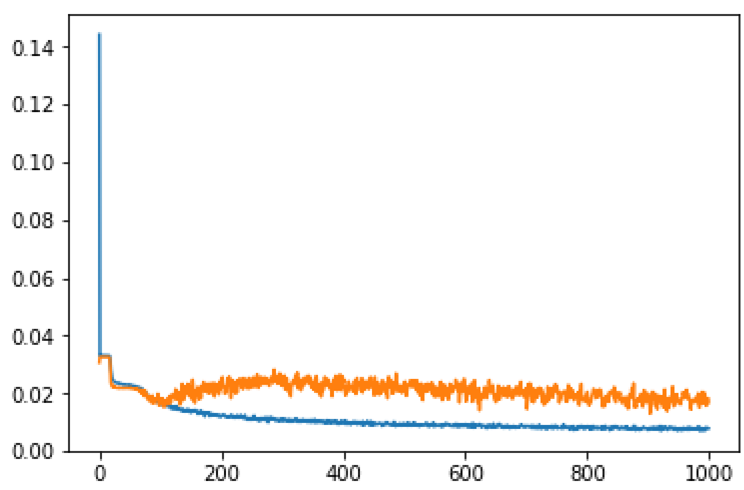


In [29]:
```
DATA = np.append(SP_F, SH_F, axis = 1)
np.random.shuffle(DATA)

Y = DATA[:, :200]
X = DATA[:,200:]

train_size = int(len(DATA) * 0.8)

train_X = X[0:train_size, :]
train_Y = Y[0:train_size, :]
test_X = X[train_size:, :]
test_Y = Y[train_size:, :]
```

In [31]:
```
model = init_model()
# history = LossHistory()
# when training, using minibatch seems to be pretty good
history = model.fit(train_X, train_Y,
                    epochs=1000,
                    batch_size=20,
                    validation_data=(test_X, test_Y),
                    verbose=2)

train_score = model.evaluate(train_X, train_Y, batch_size=100)
test_score = model.evaluate(test_X, test_Y, batch_size= 50)
print(train_score)
print(test_score)
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='test')
```

```
[0.015331682799163686, 0.0241935479264426]
[0.0180212862749574, 0.03225806399538953]
```

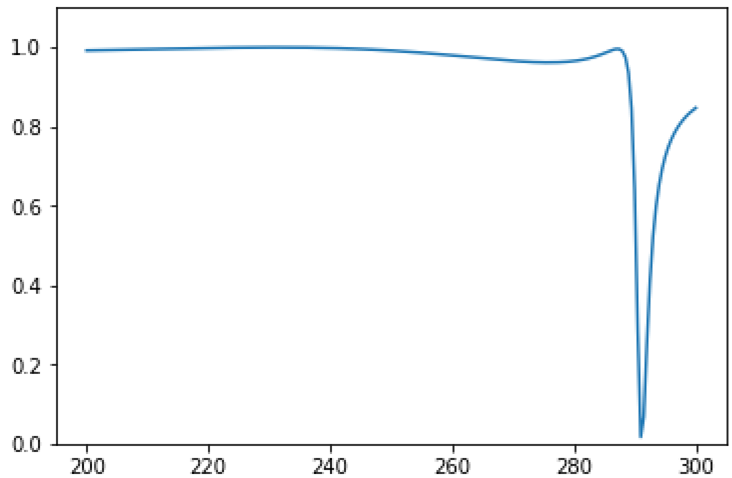
Out[31]: [<matplotlib.lines.Line2D at 0x177008ddc88>]



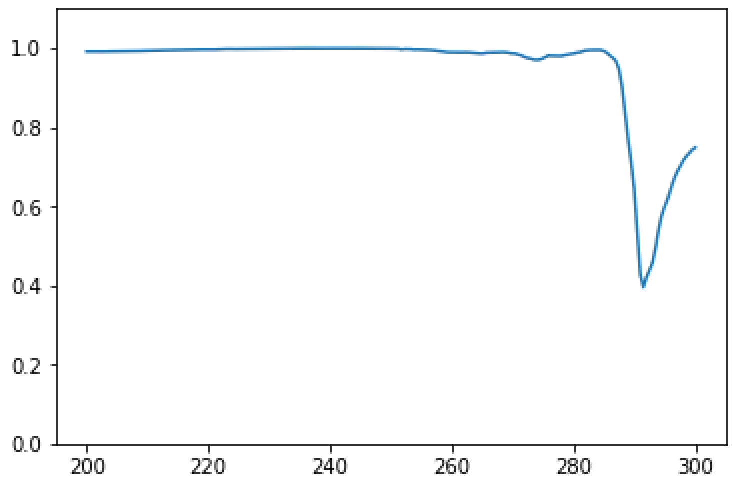
In [32]:

```
x = np.genfromtxt('meep_code/data/SP_xaxis.txt')
for i in range(len(test_X)):
    print('Test '+str(i))
    print('True spectrum: ')
    plt.ylim(0, 1.1)
    plt.plot(x, test_Y[i])
    plt.show()
    print('Predicted spectrum: ')
    plt.ylim(0, 1.1)
    plt.plot(x, np.reshape(model.predict(np.reshape(test_X[i], (1, 6))), (200
    plt.show()
```

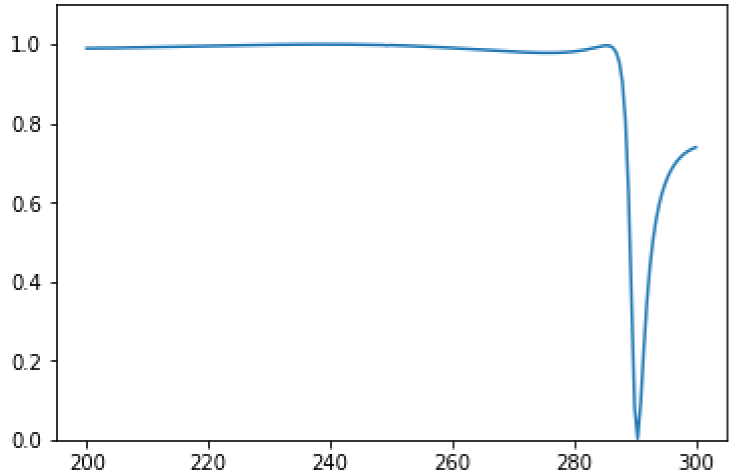
```
Test 0
True spectrum:
```



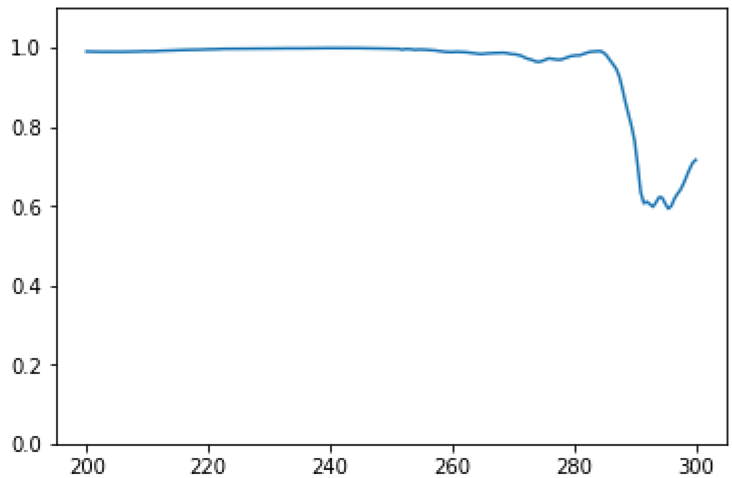

```
Predicted spectrum:
```



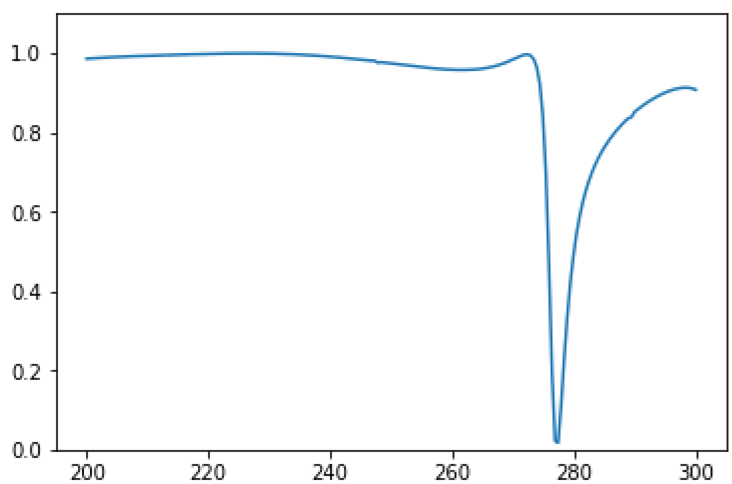

```
Test 1
True spectrum:
```



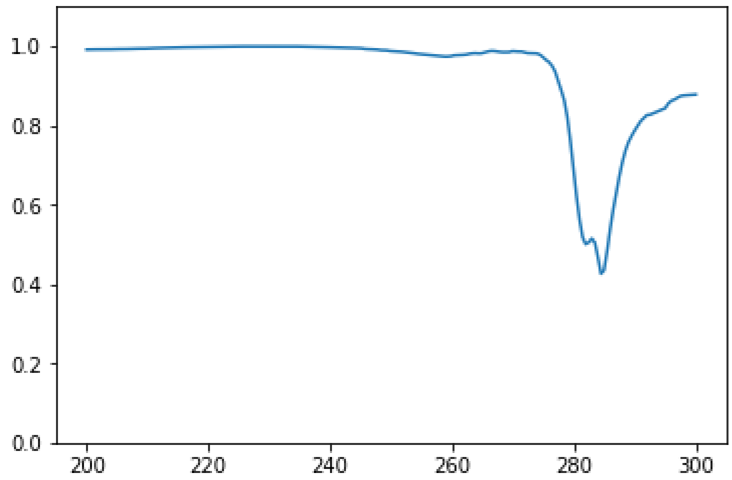
Predicted spectrum:



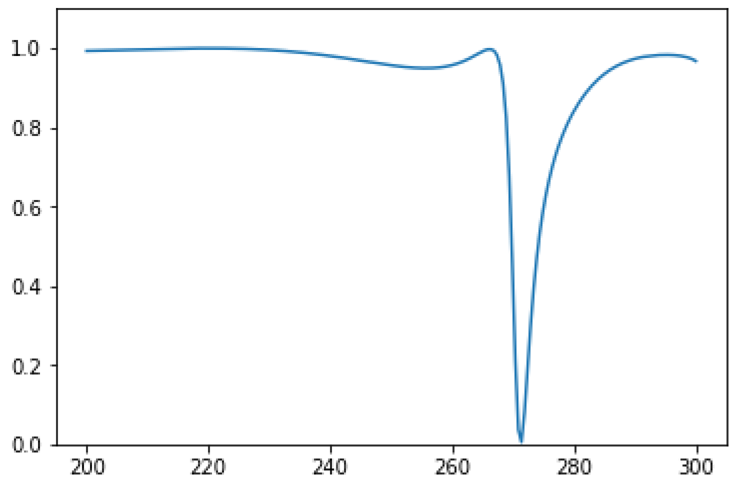
Test 2
True spectrum:



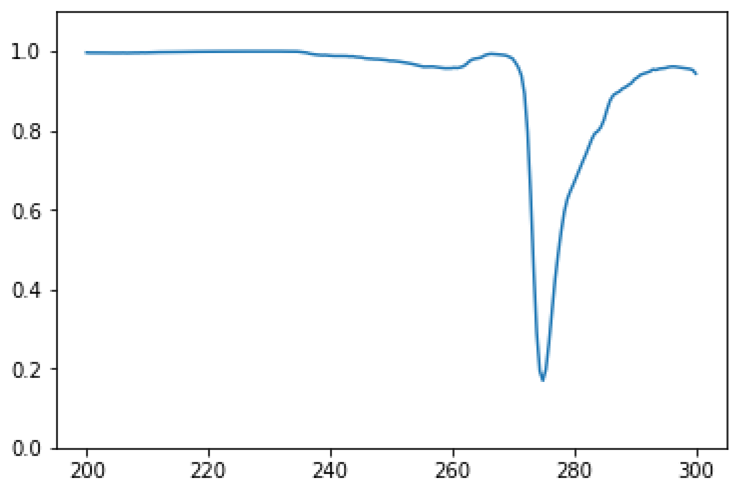
Predicted spectrum:



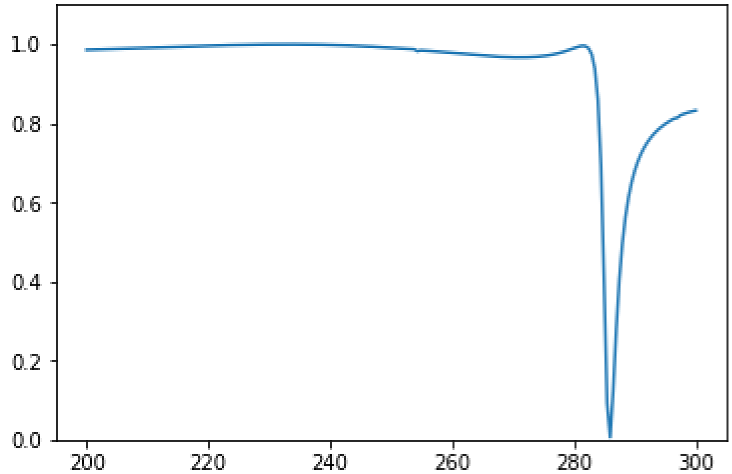
Test 3
True spectrum:



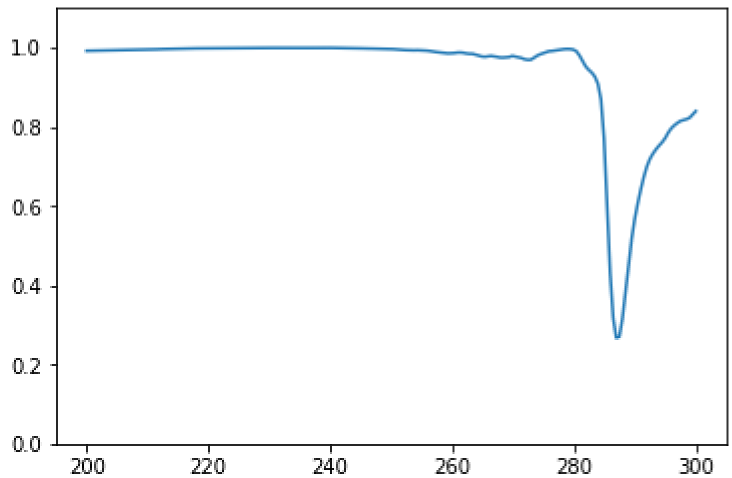
Predicted spectrum:



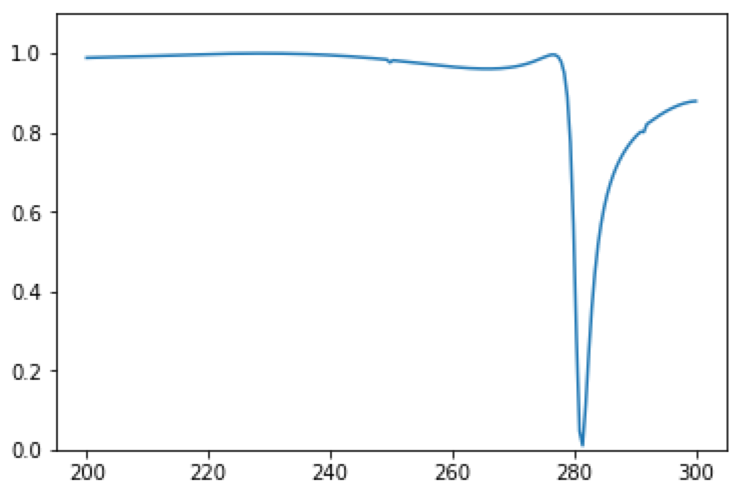
Test 4
True spectrum:



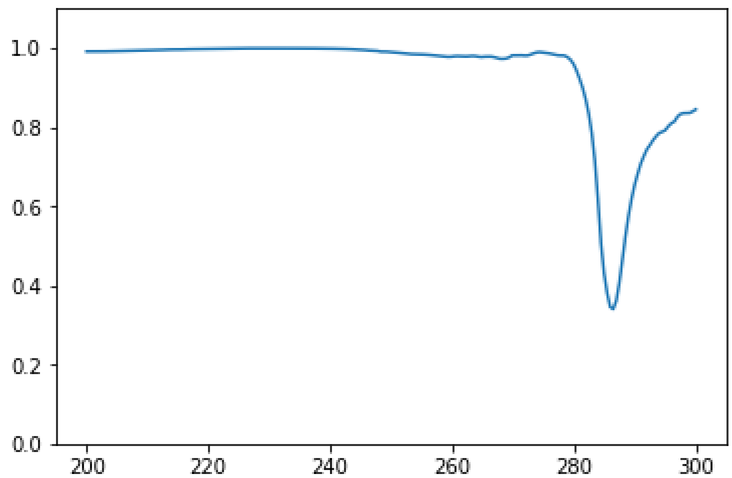

```
Predicted spectrum:
```



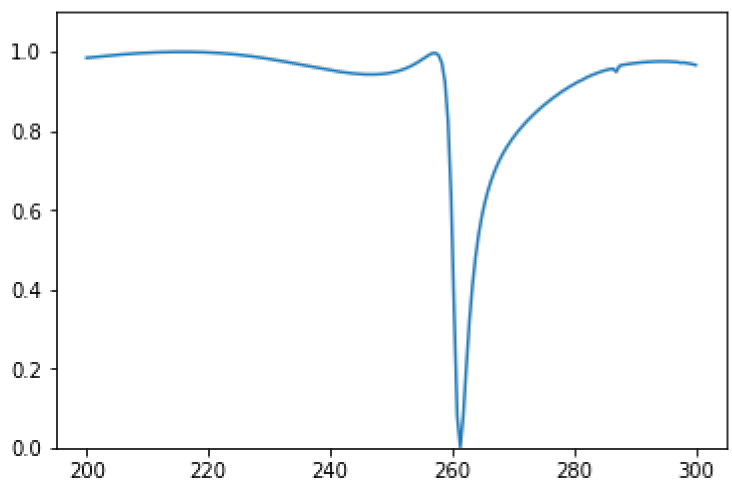

```
Test 5
True spectrum:
```



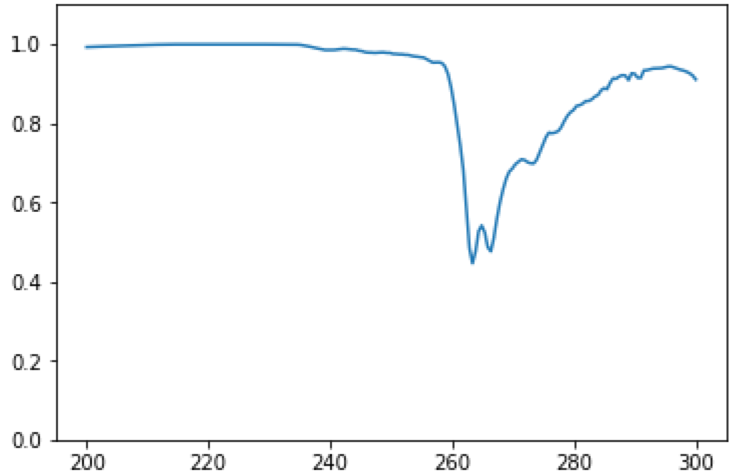
```
Predicted spectrum:
```



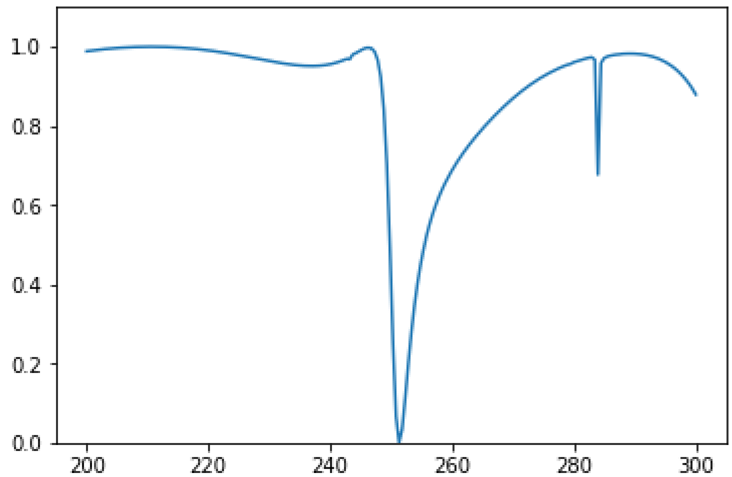

```
Test 6
True spectrum:
```



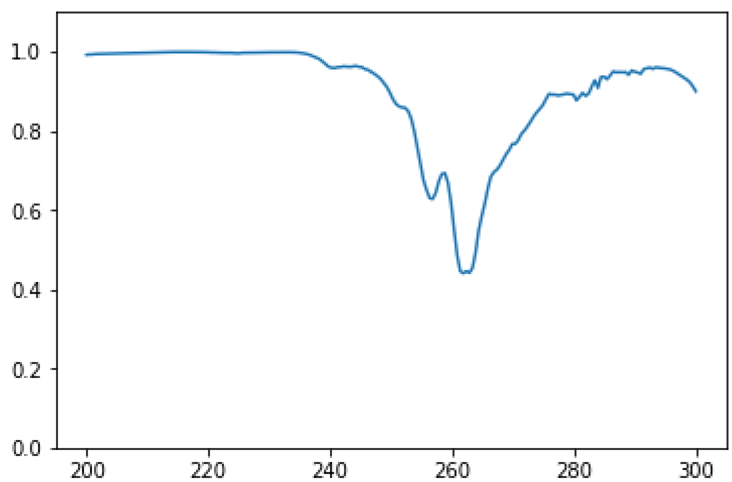

```
Predicted spectrum:
```



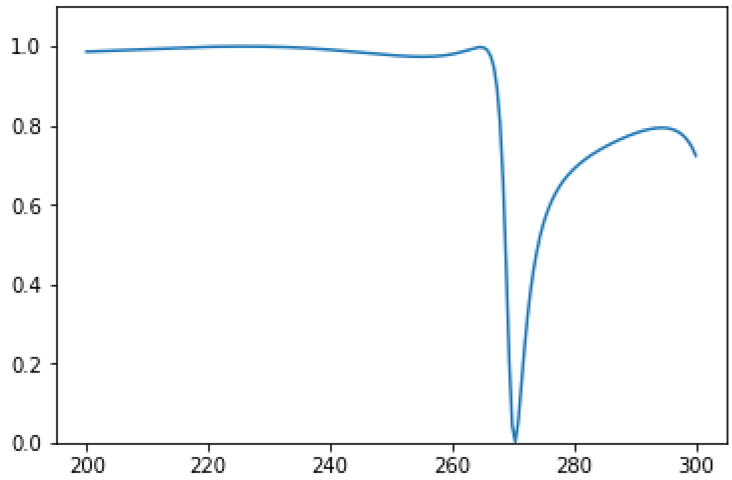

```
Test 7
True spectrum:
```



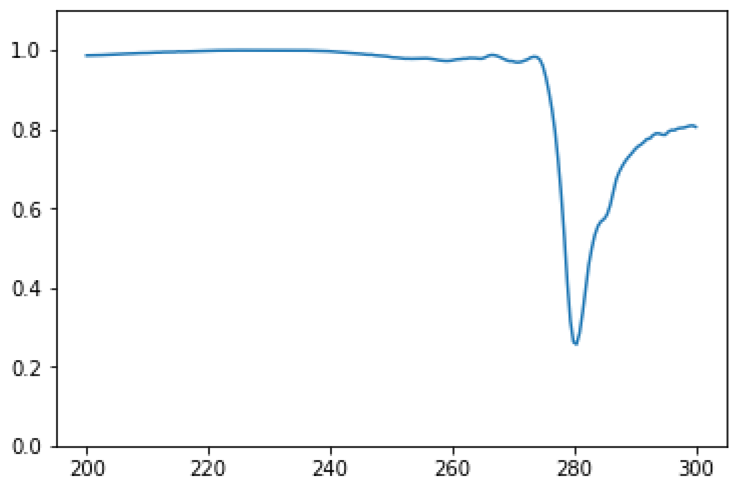

```
Predicted spectrum:
```



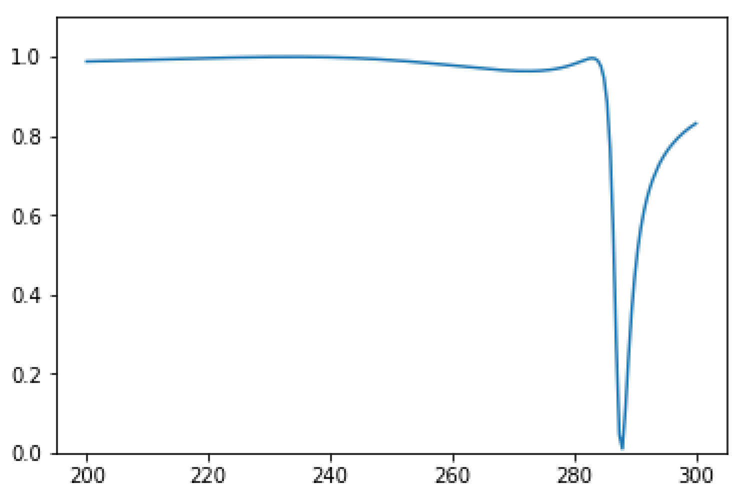

```
Test 8
True spectrum:
```



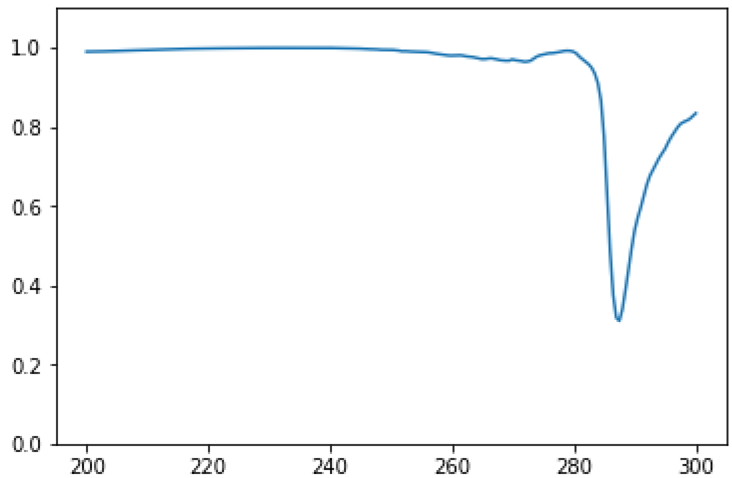

```
Predicted spectrum:
```



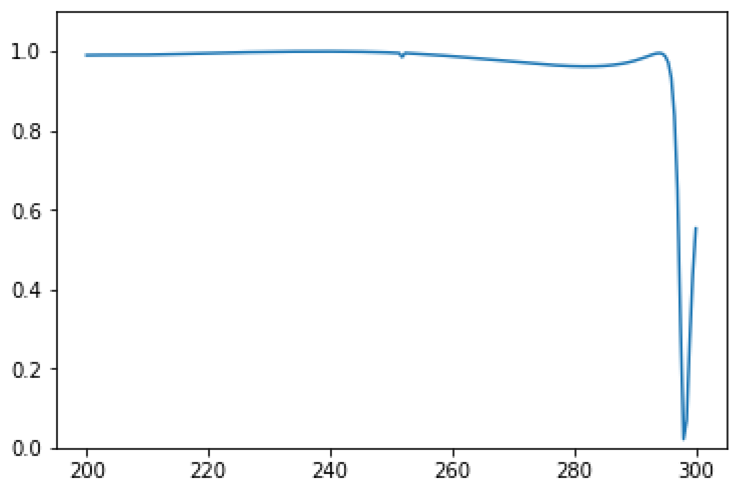

```
Test 9
True spectrum:
```



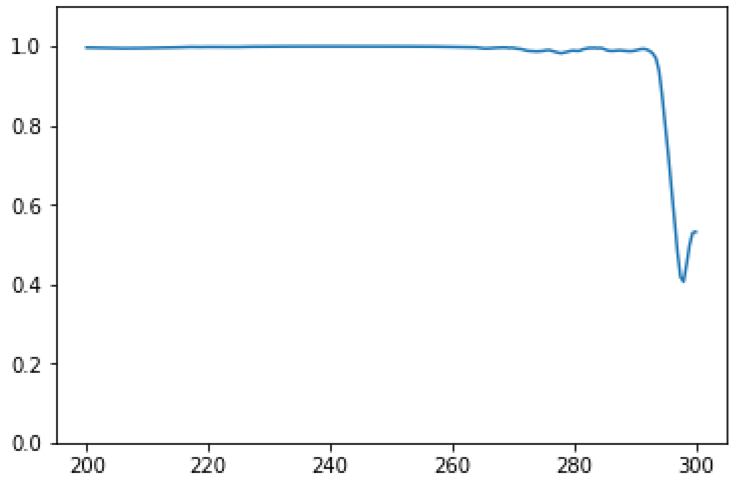
```
Predicted spectrum:
```



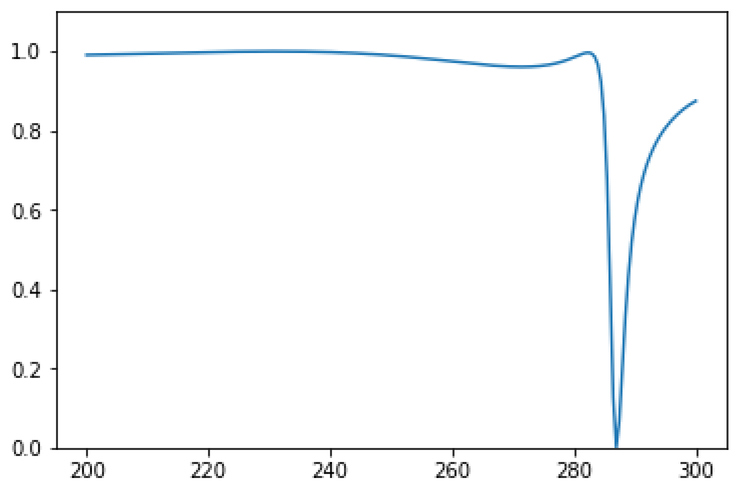

```
Test 10
True spectrum:
```



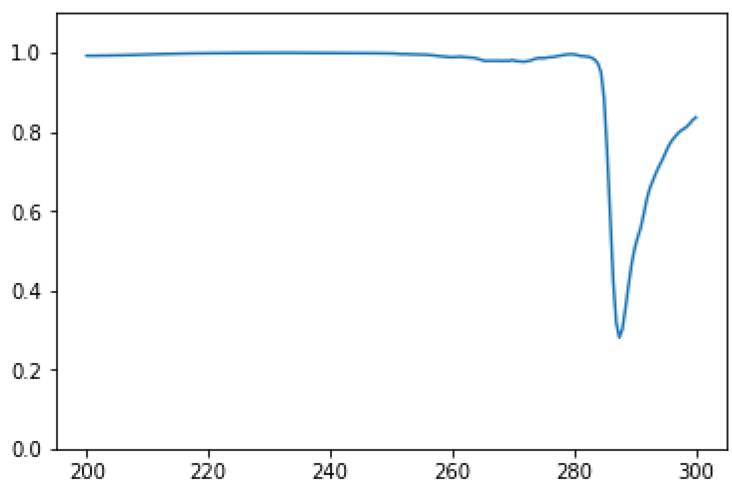

```
Predicted spectrum:
```



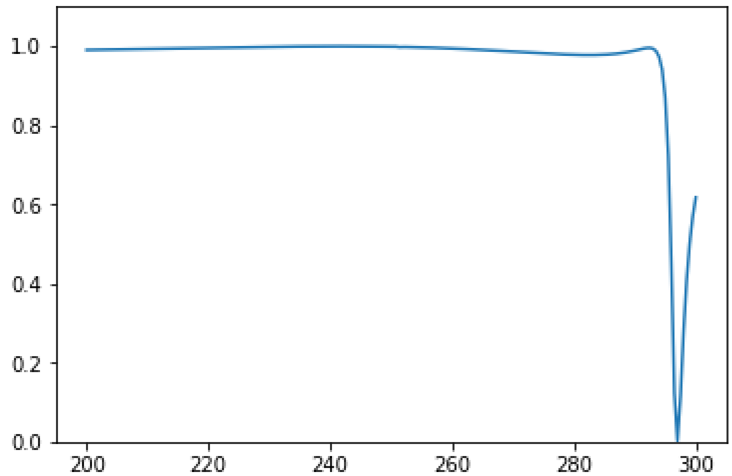

```
Test 11
True spectrum:
```



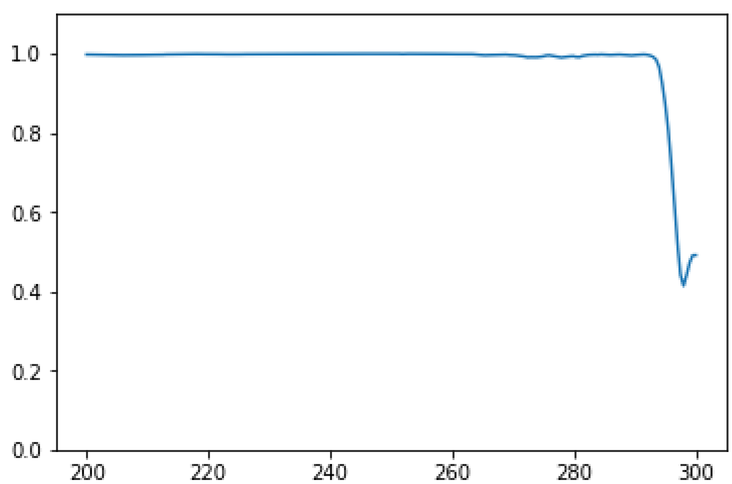

```
Predicted spectrum:
```




```
Test 12
True spectrum:
```

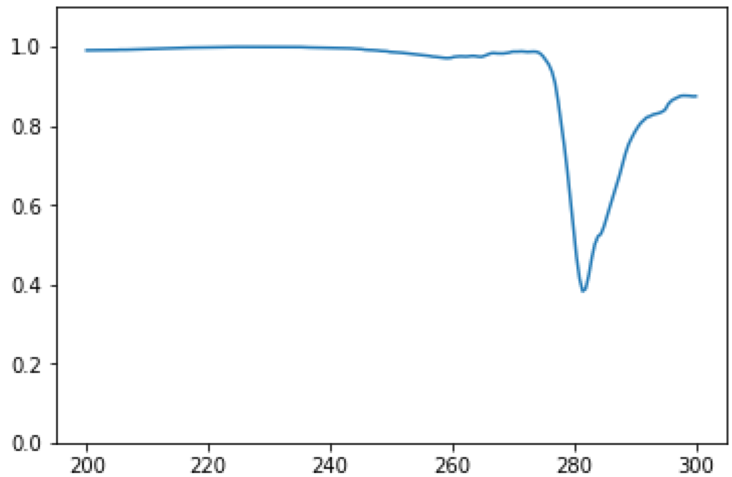


Predicted spectrum:

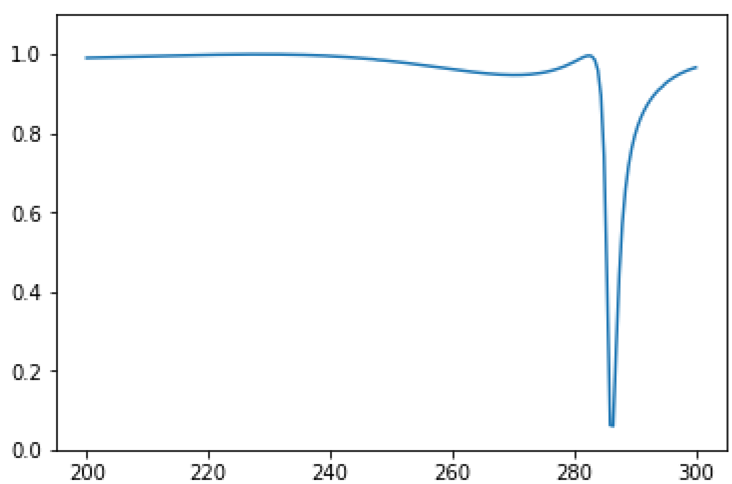


Test 13
True spectrum:

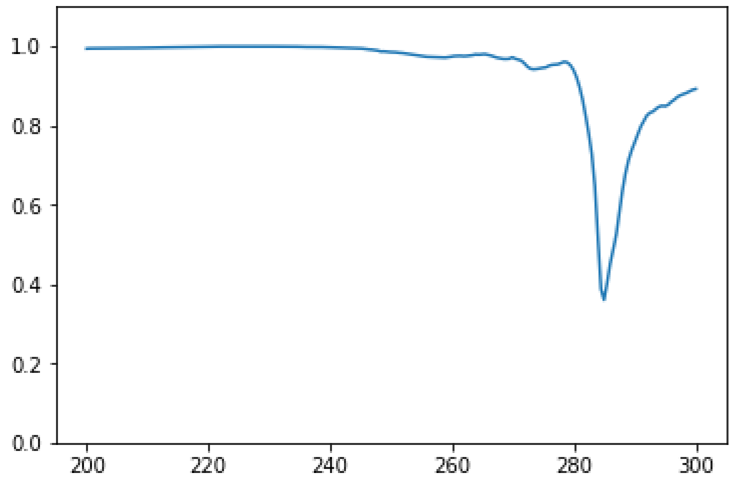
Predicted spectrum:



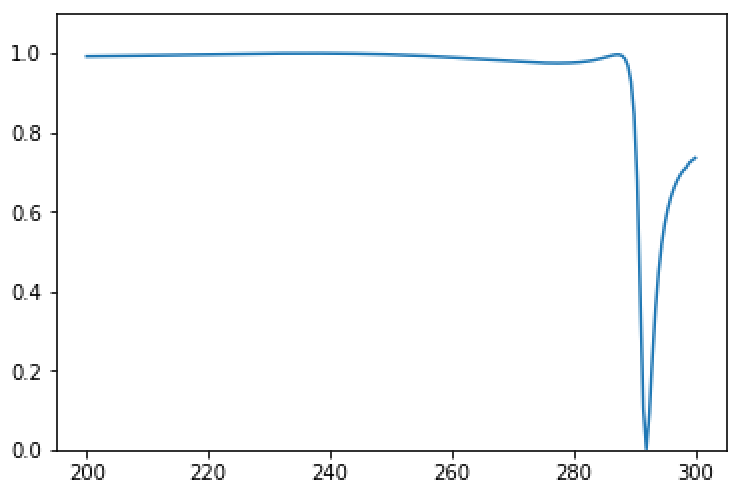
Test 14
True spectrum:



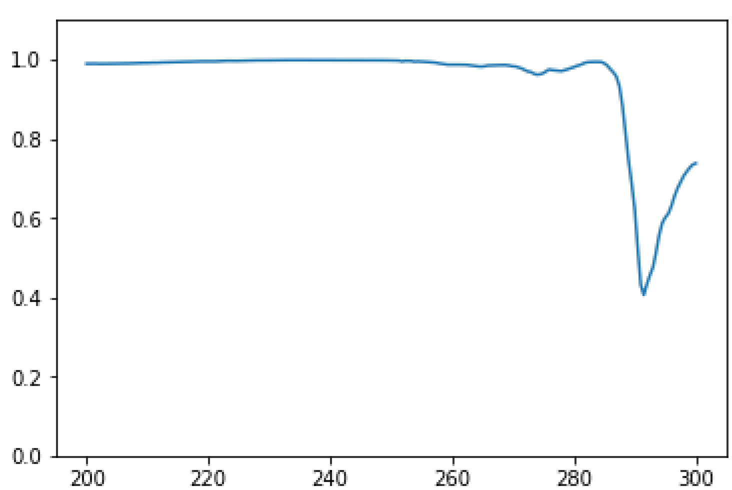
Predicted spectrum:



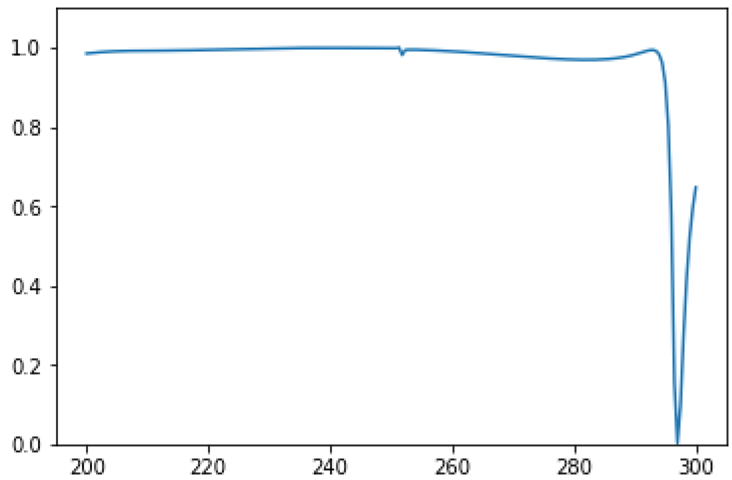

```
Test 15
True spectrum:
```



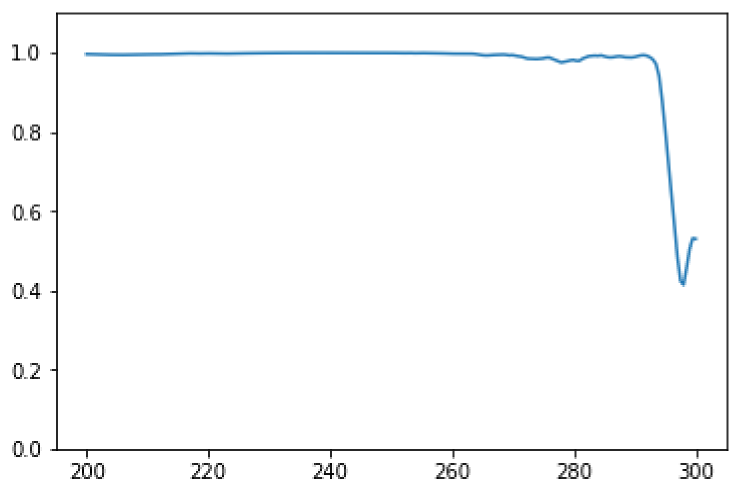

```
Predicted spectrum:
```



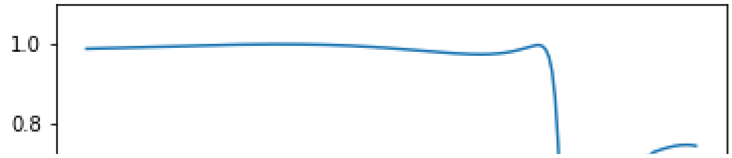
Test 16
True spectrum:



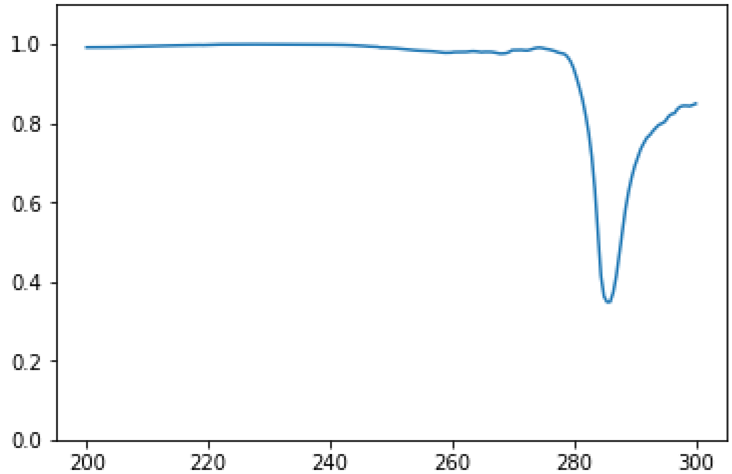
Predicted spectrum:



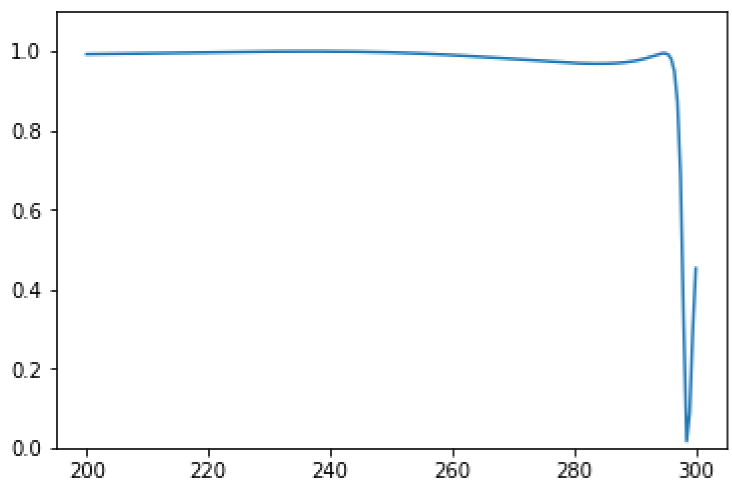
Test 17
True spectrum:



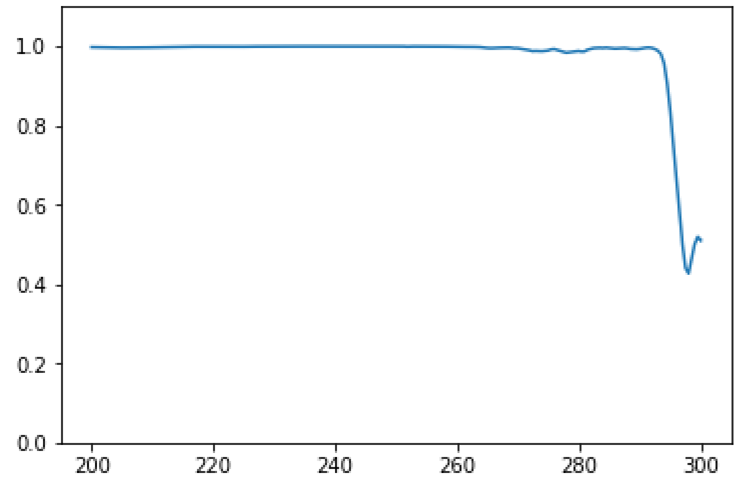
Predicted spectrum:



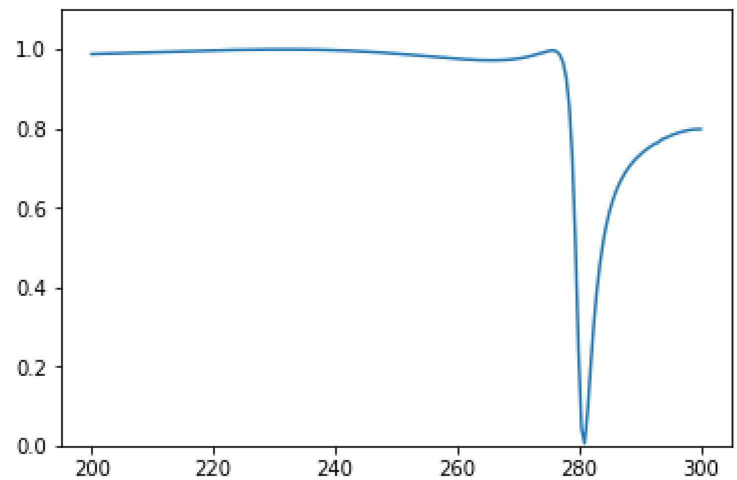
Test 18
True spectrum:



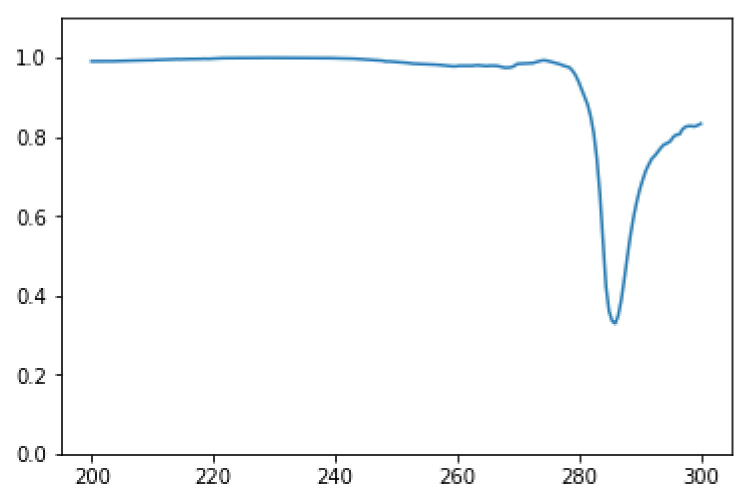
Predicted spectrum:



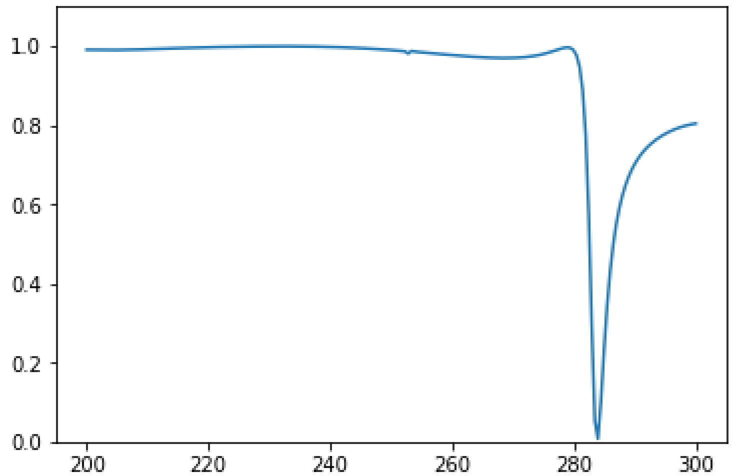
Test 19
True spectrum:



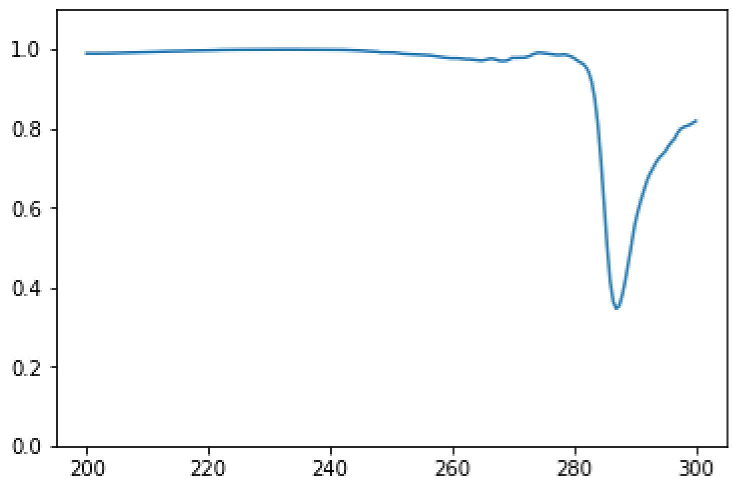
Predicted spectrum:



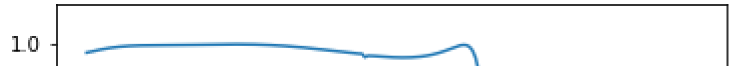
Test 20
True spectrum:



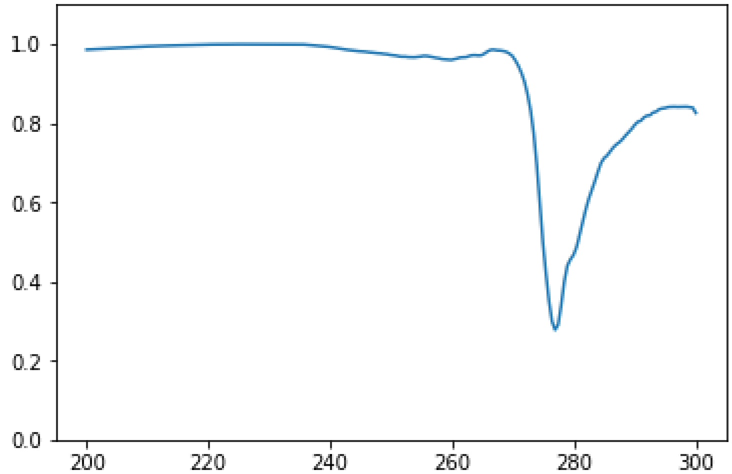
Predicted spectrum:



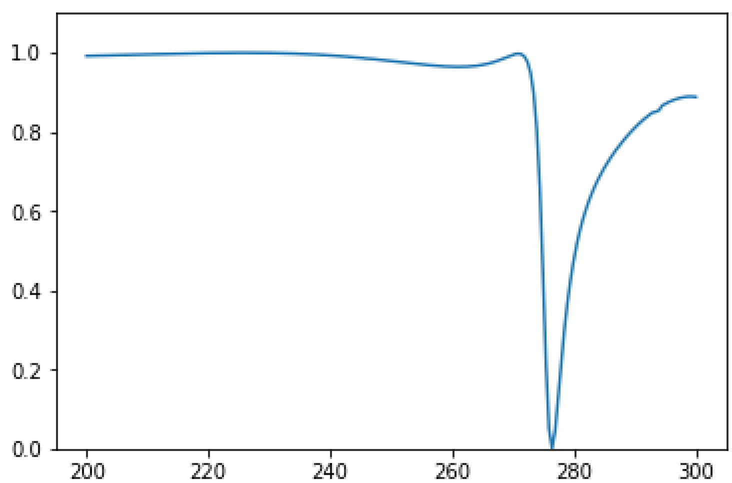
Test 21
True spectrum:



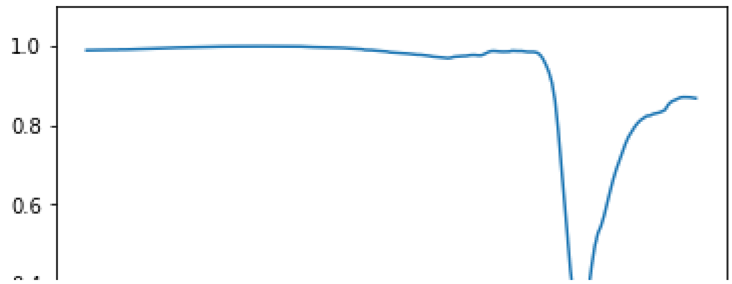
Predicted spectrum:



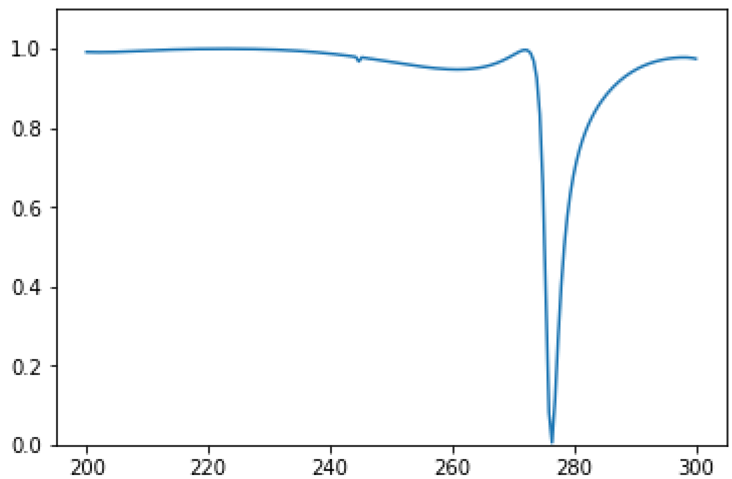
Test 22
True spectrum:



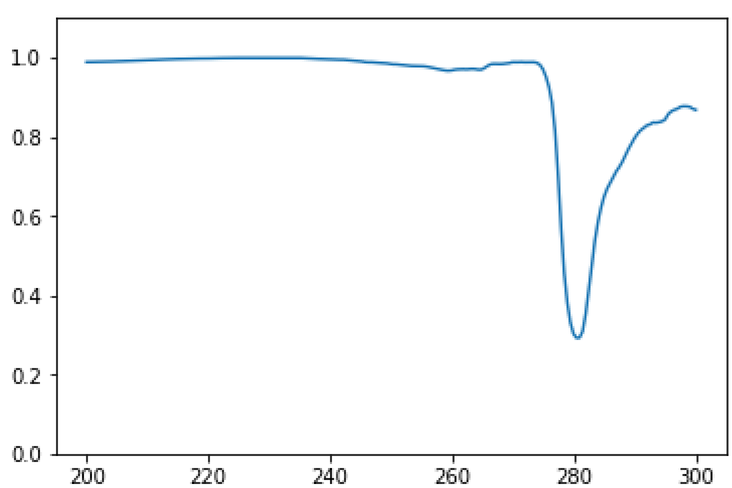
Predicted spectrum:



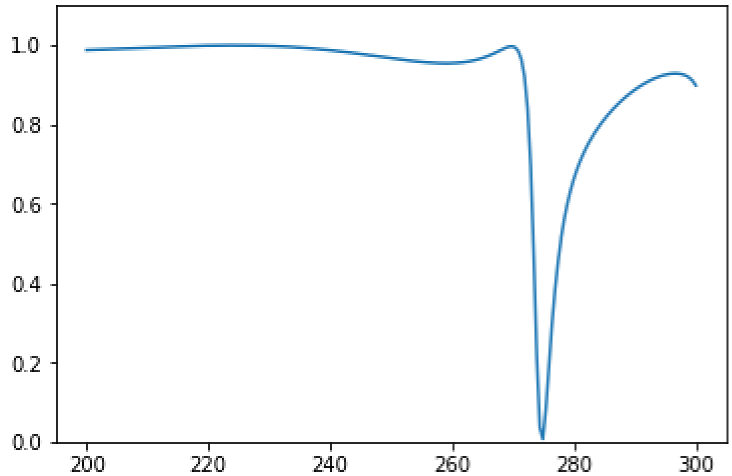

```
Test 23
True spectrum:
```



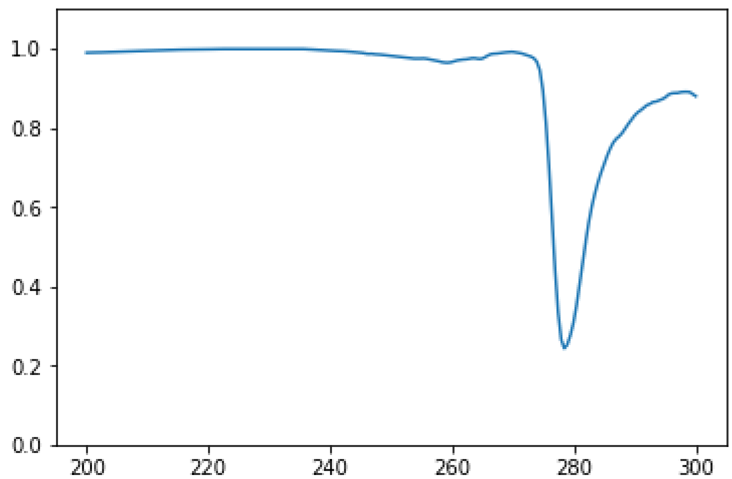

```
Predicted spectrum:
```



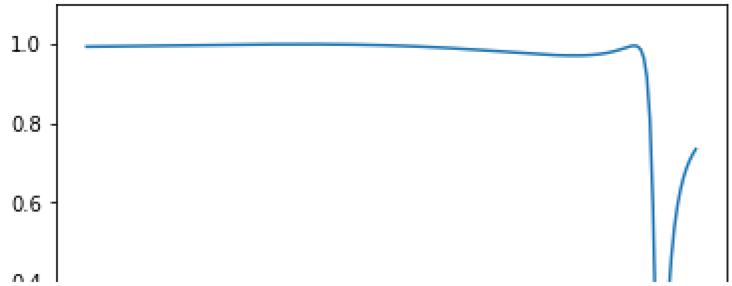
Test 24
True spectrum:



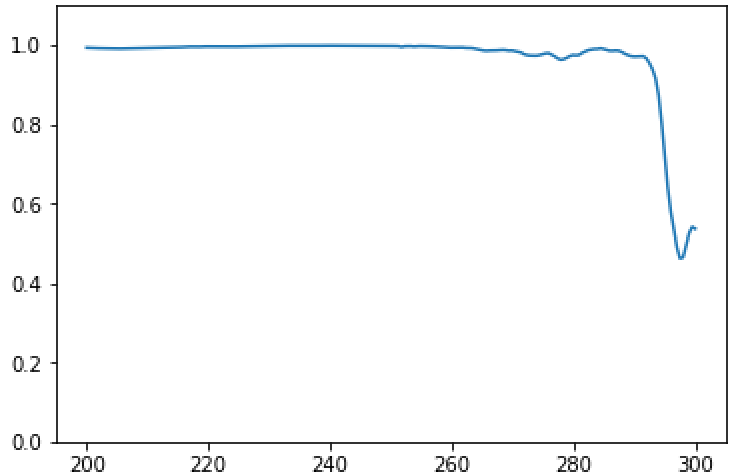
Predicted spectrum:



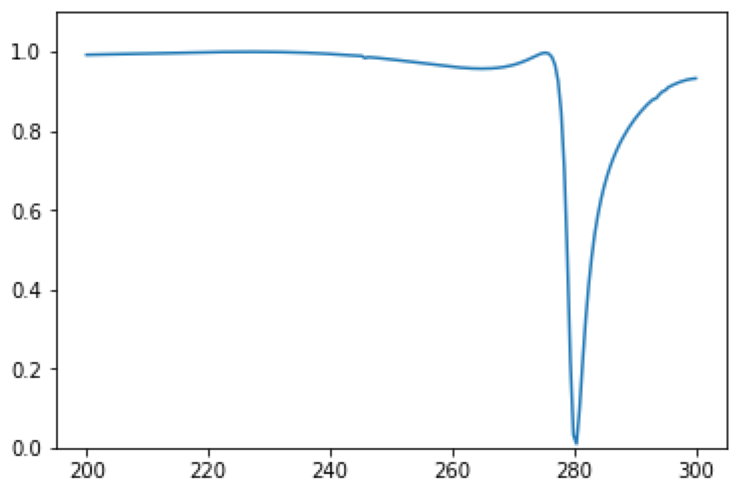
Test 25
True spectrum:



Predicted spectrum:

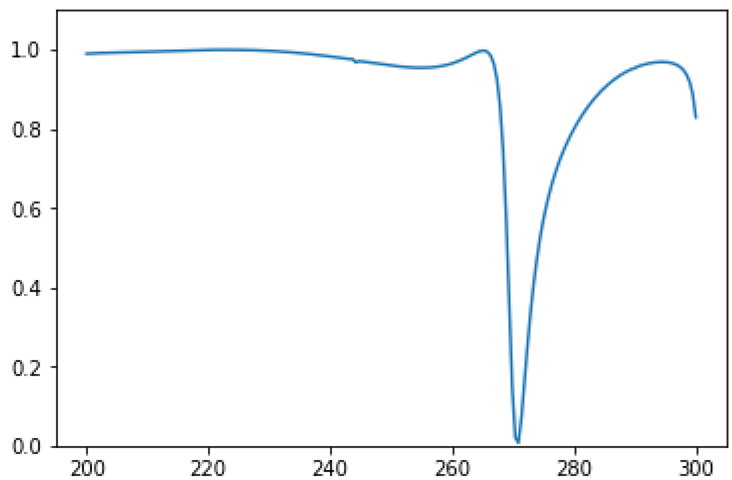


Test 26
True spectrum:

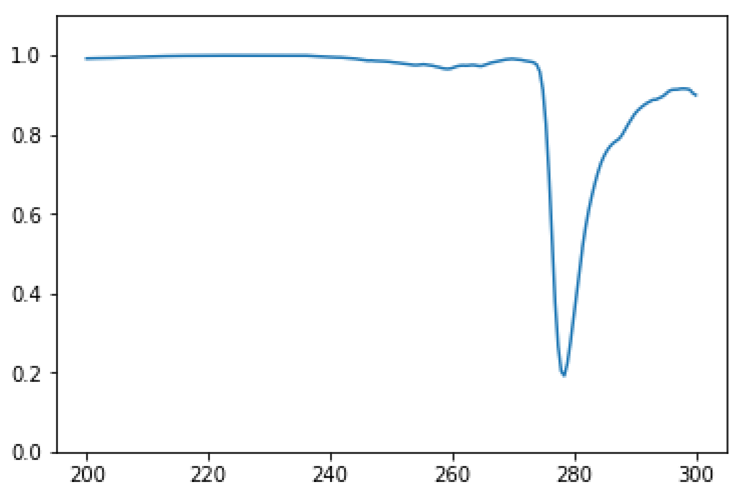


Predicted spectrum:

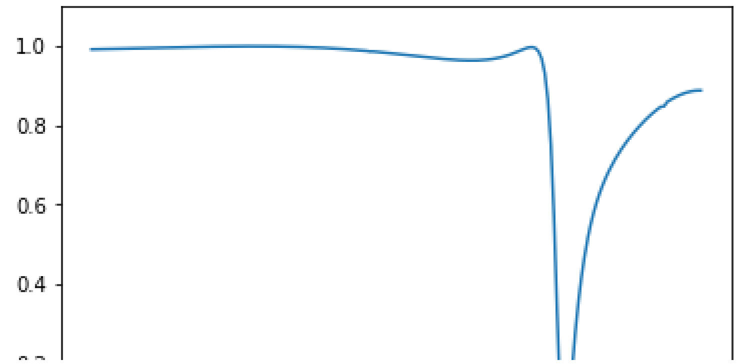

```
Test 27
True spectrum:
```



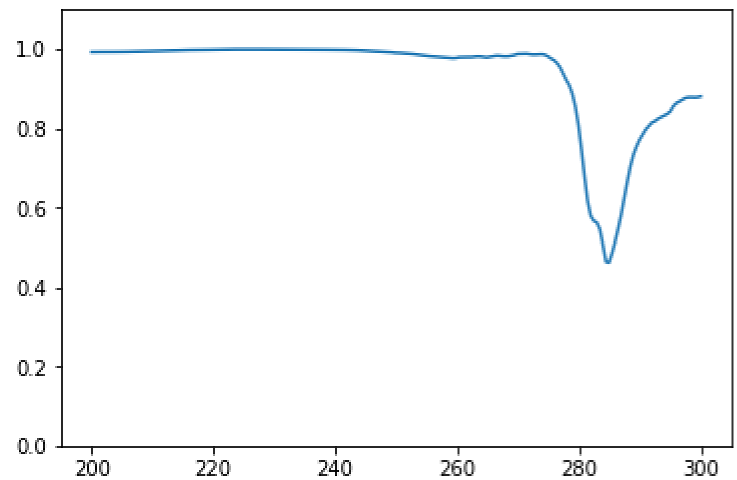

```
Predicted spectrum:
```



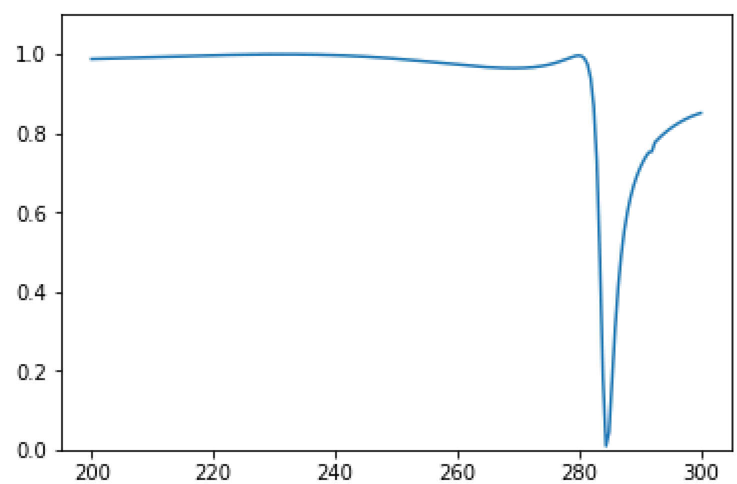

```
Test 28
True spectrum:
```



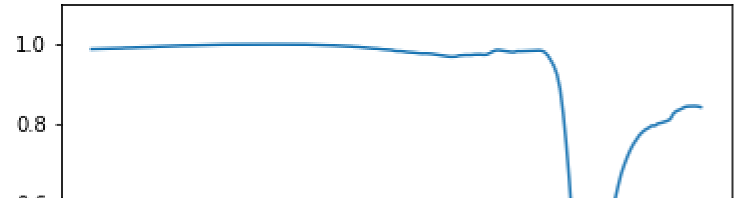
Predicted spectrum:



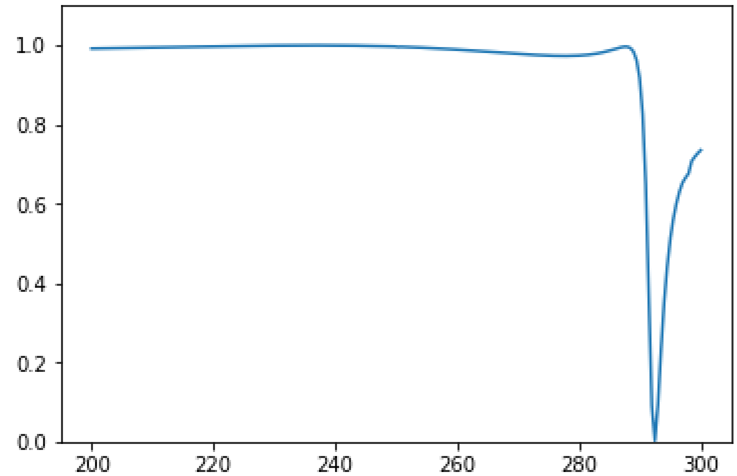
Test 29
True spectrum:



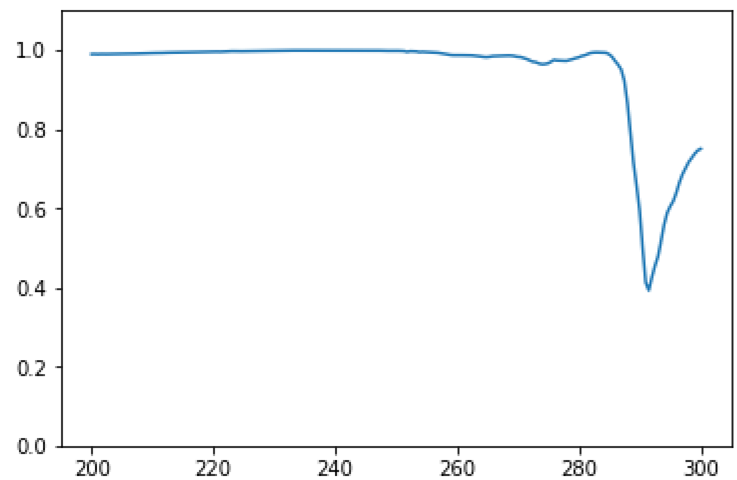
Predicted spectrum:



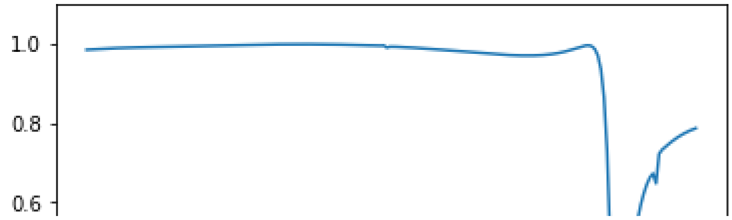

```
Test 30
True spectrum:
```



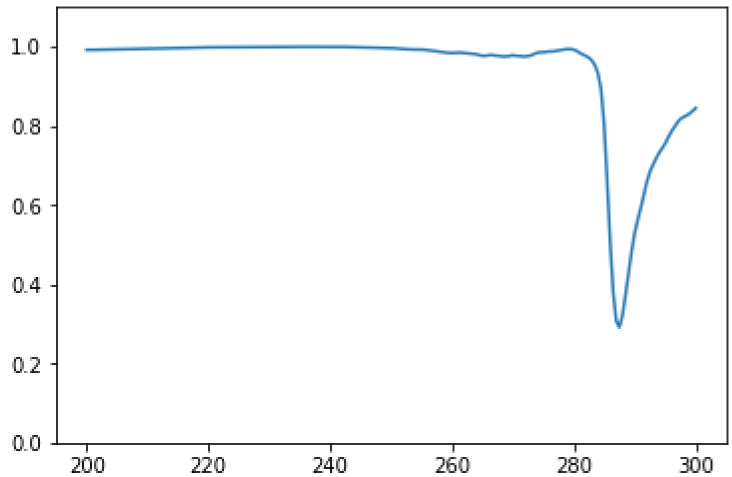

```
Predicted spectrum:
```



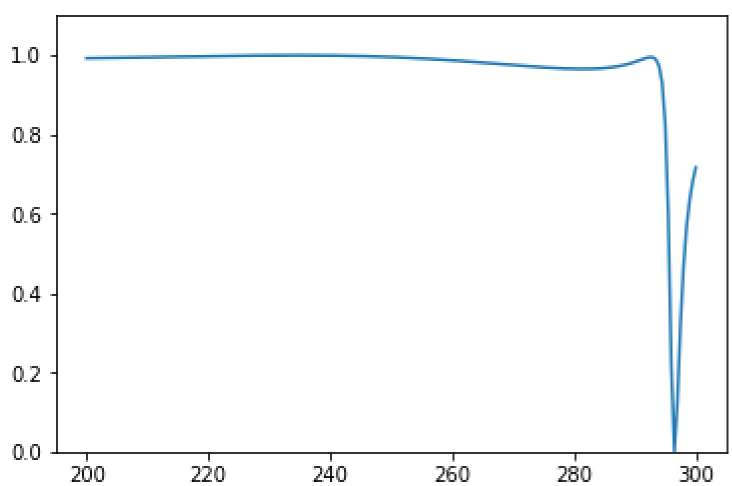

```
Test 31
True spectrum:
```



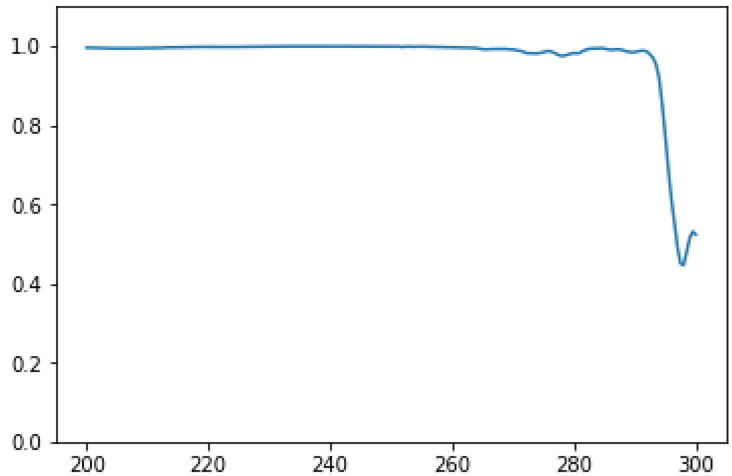
Predicted spectrum:



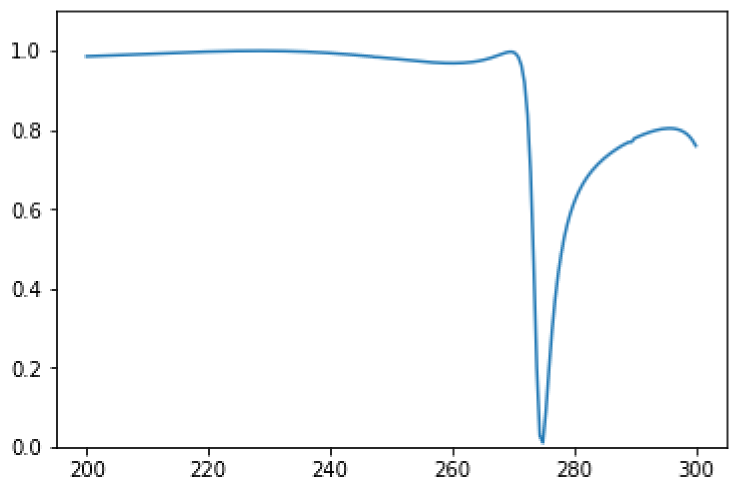
Test 32
True spectrum:



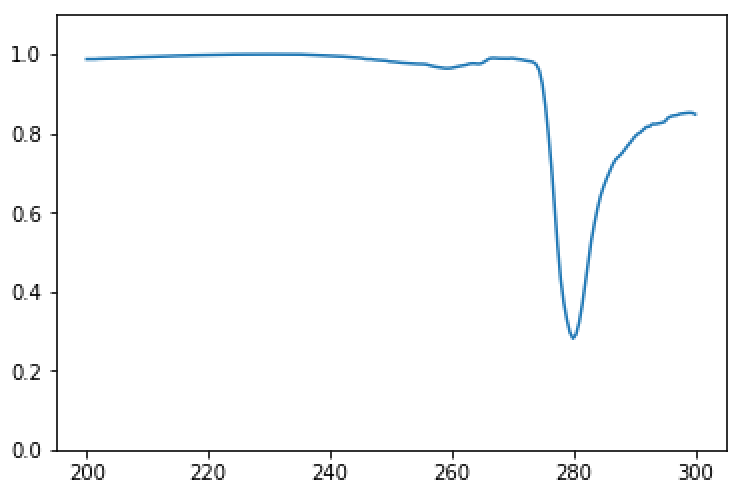
Predicted spectrum:



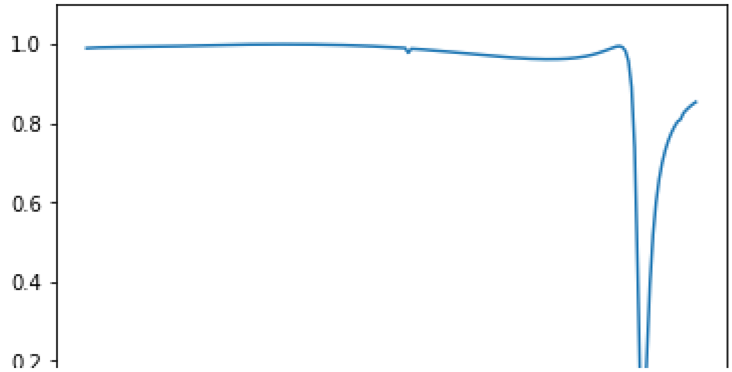

```
Test 33
True spectrum:
```



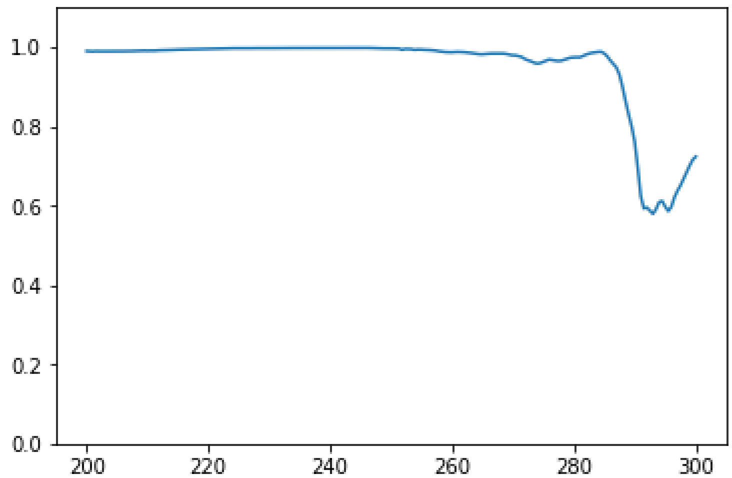

```
Predicted spectrum:
```



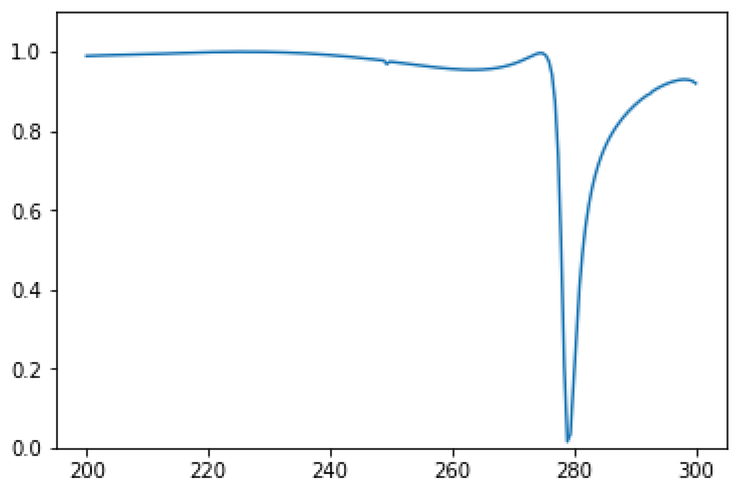

```
Test 34
True spectrum:
```



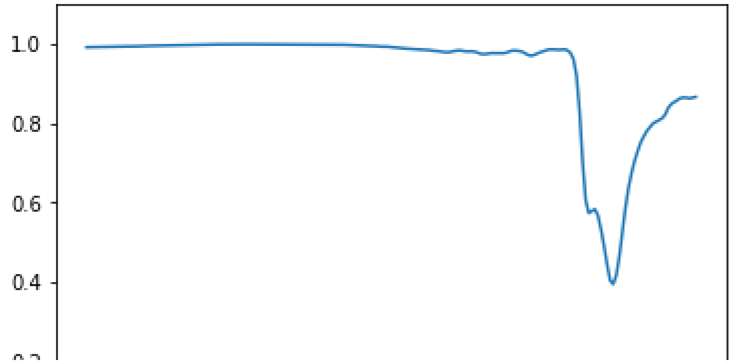

```
Predicted spectrum:
```



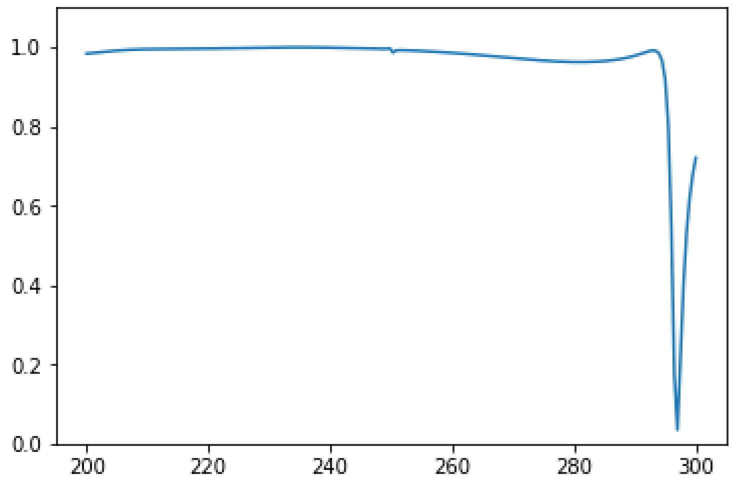

```
Test 35
True spectrum:
```



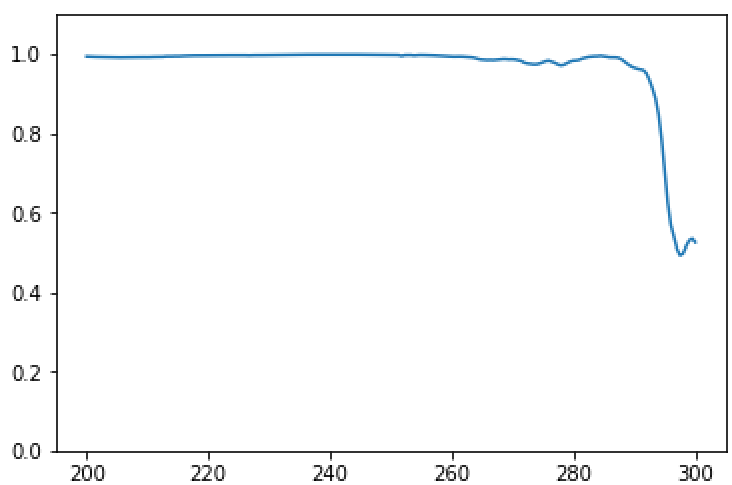

```
Predicted spectrum:
```



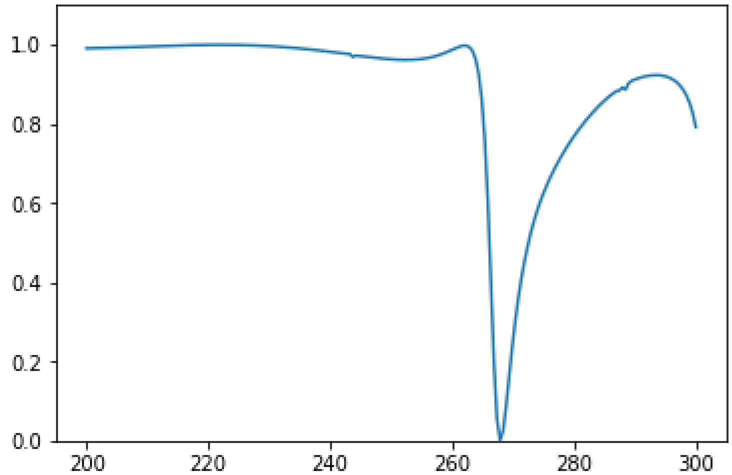

```
Test 36
True spectrum:
```



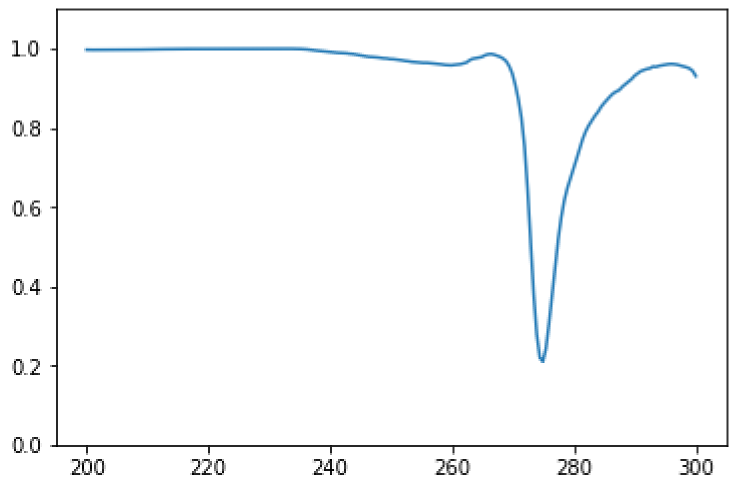

```
Predicted spectrum:
```



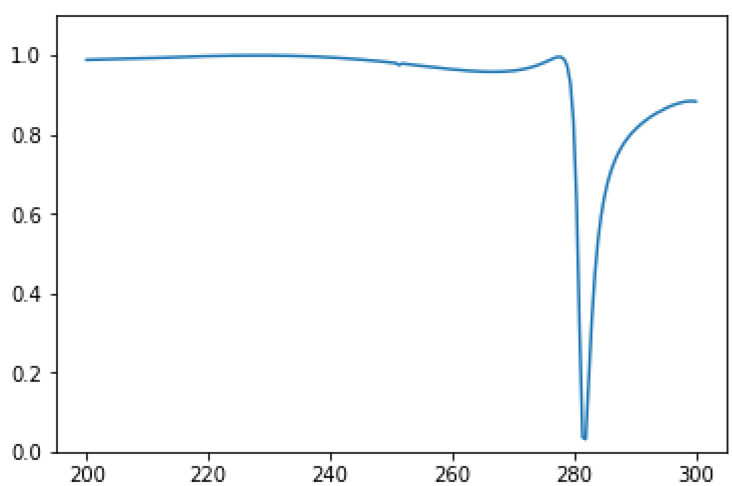
```
Test 37
True spectrum:
```



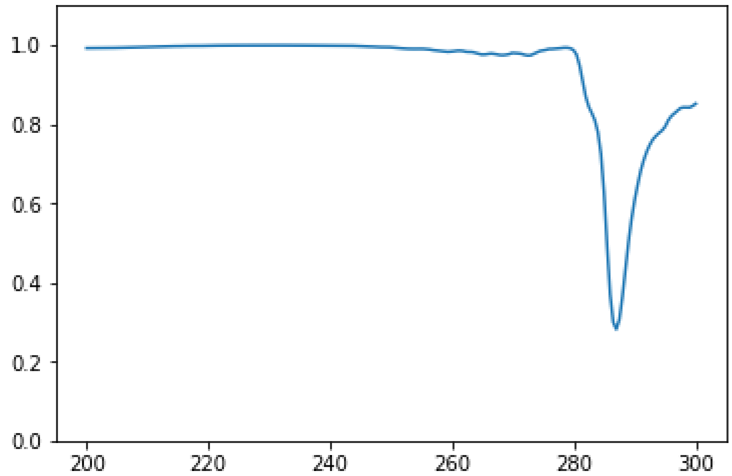

```
Predicted spectrum:
```



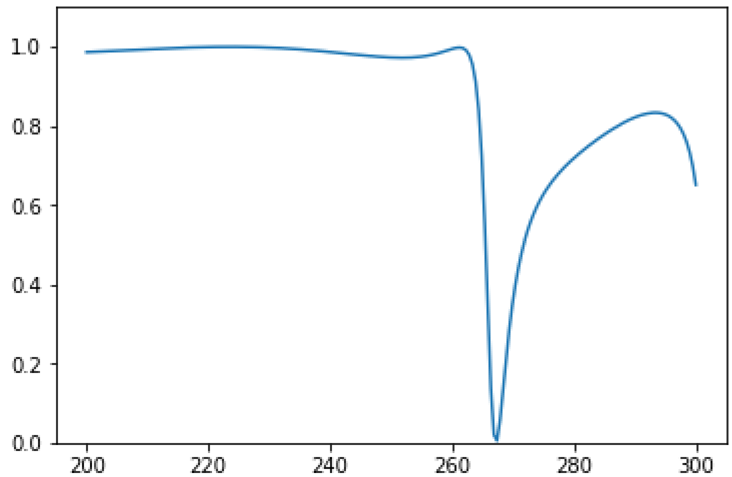

```
Test 38
True spectrum:
```



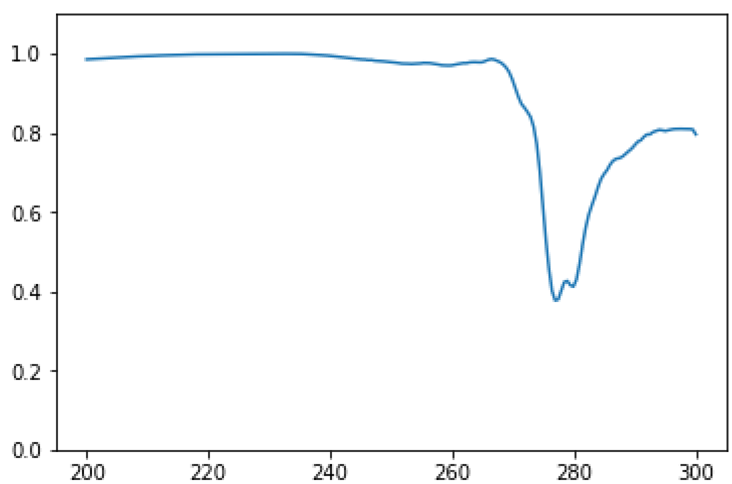

```
Predicted spectrum:
```



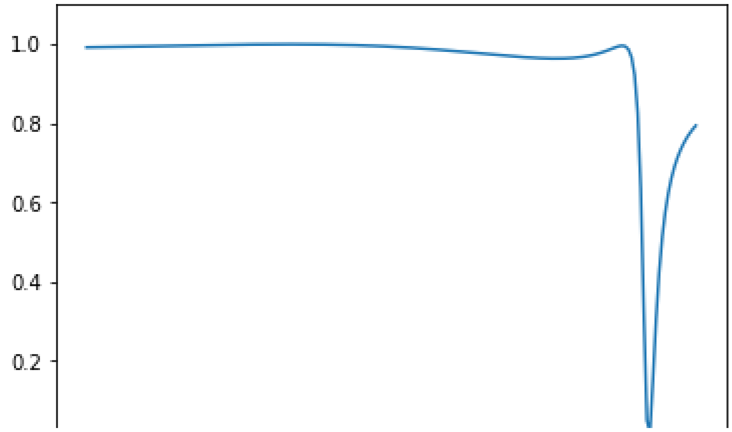

```
Test 39
True spectrum:
```



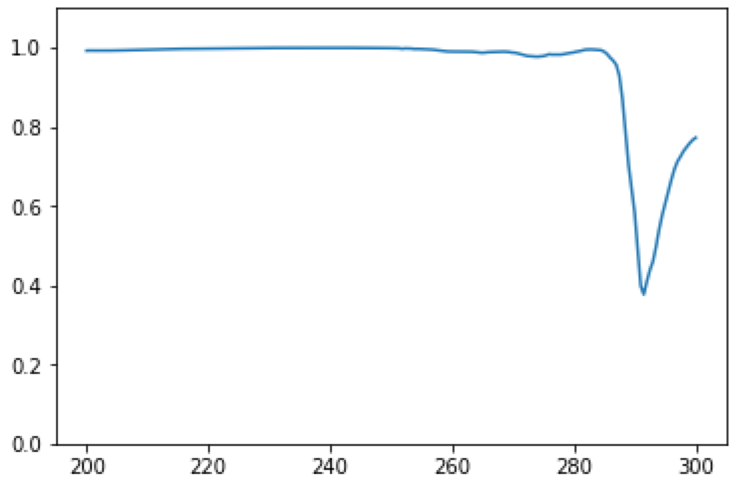

```
Predicted spectrum:
```



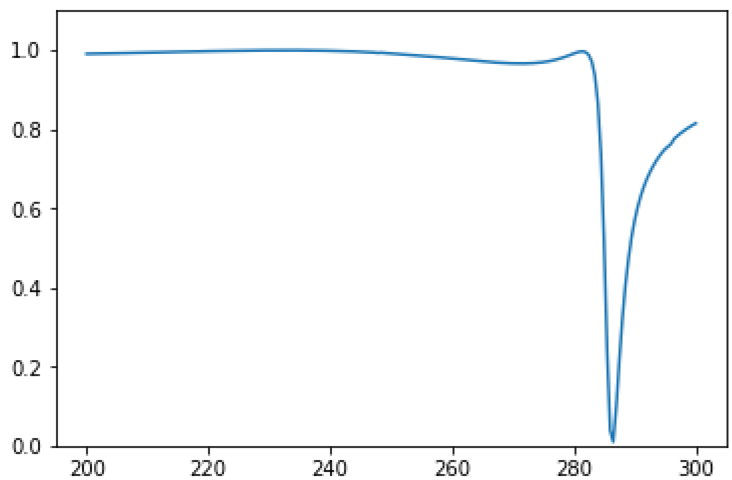

```
Test 40
True spectrum:
```



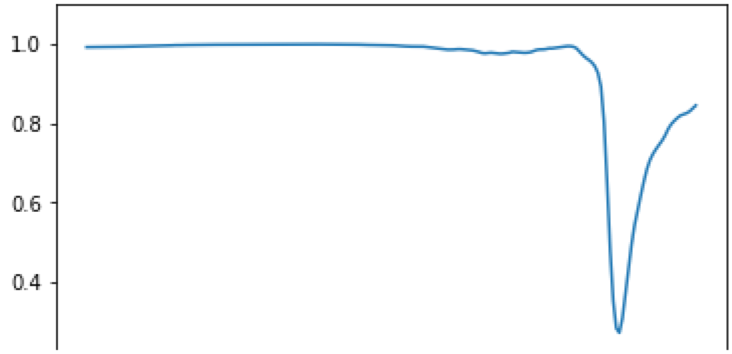
Predicted spectrum:



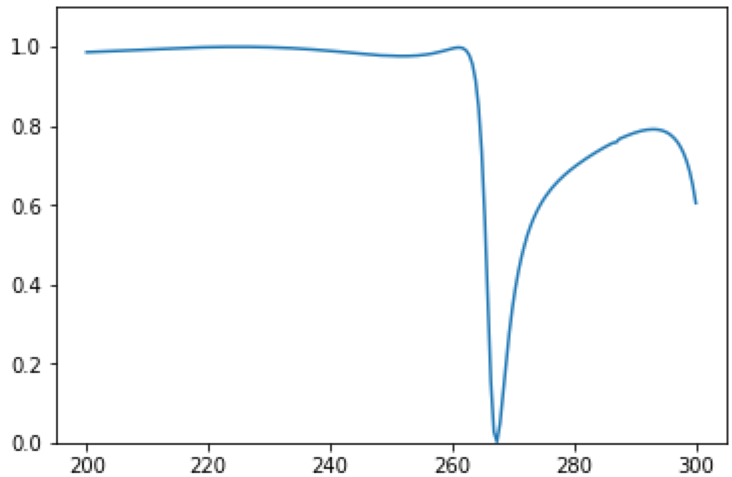
Test 41
True spectrum:



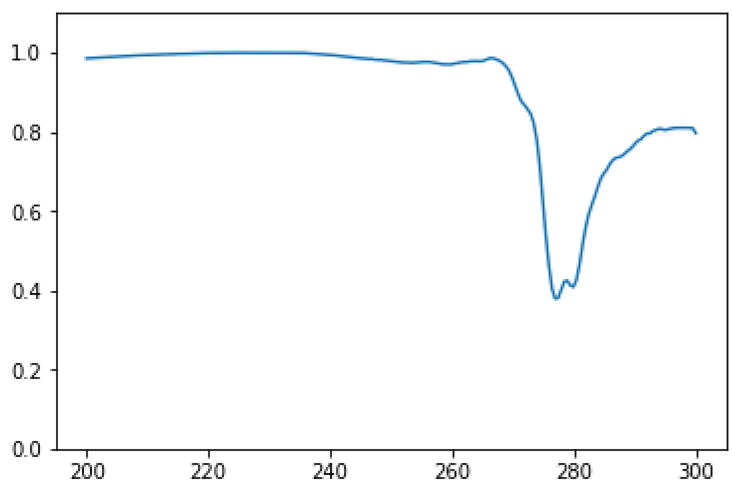
Predicted spectrum:



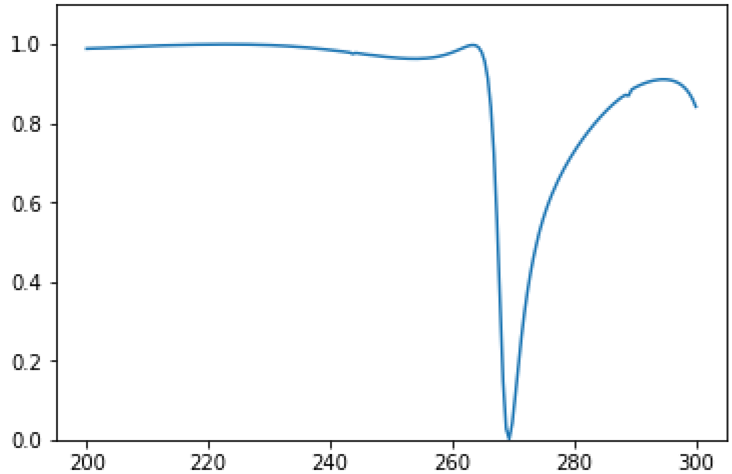

```
Test 42
True spectrum:
```



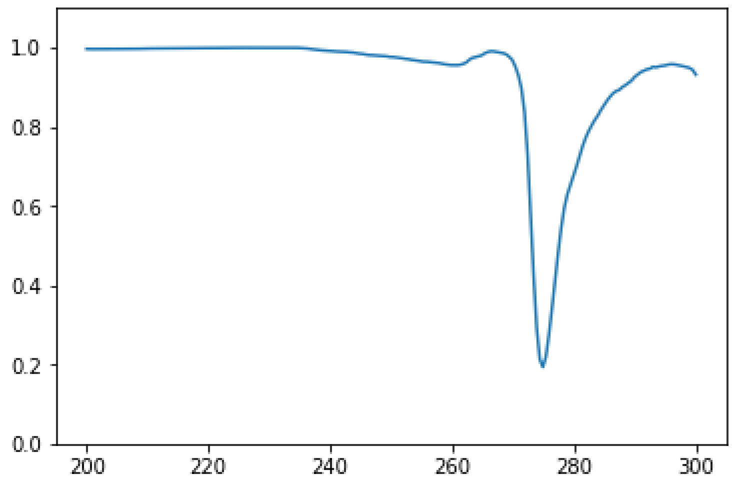

```
Predicted spectrum:
```



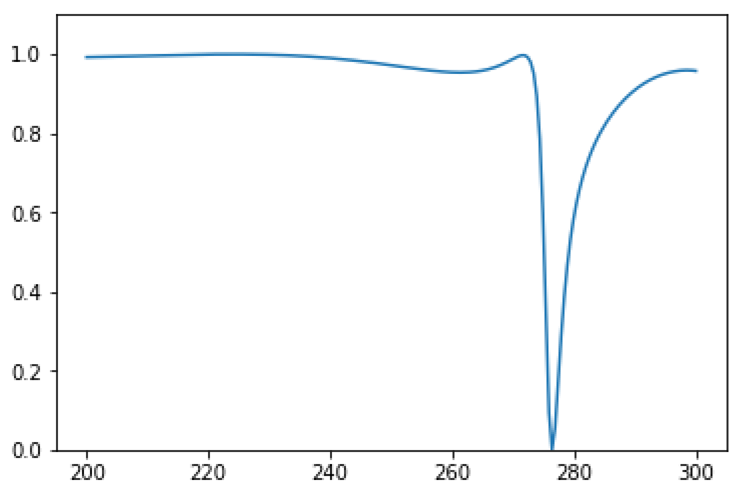

```
Test 43
True spectrum:
```



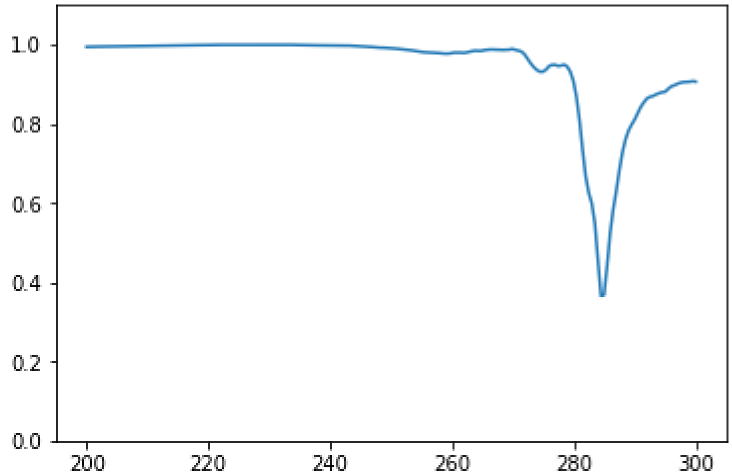

```
Predicted spectrum:
```



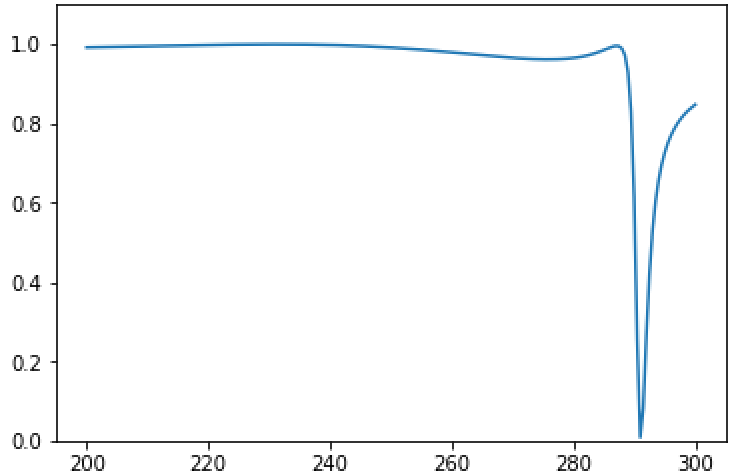

```
Test 44
True spectrum:
```



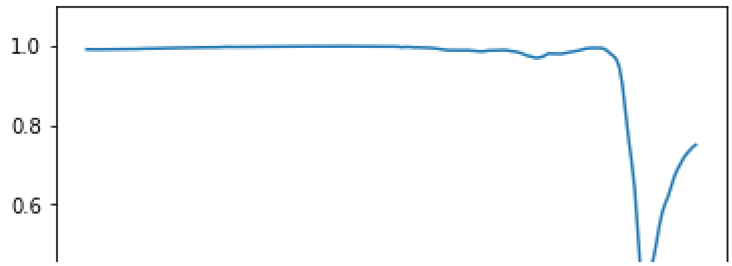
```
Predicted spectrum:
```



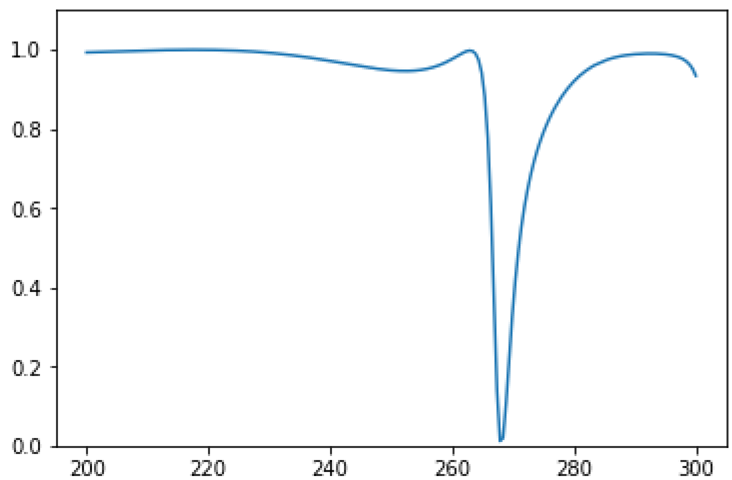

```
Test 45
True spectrum:
```



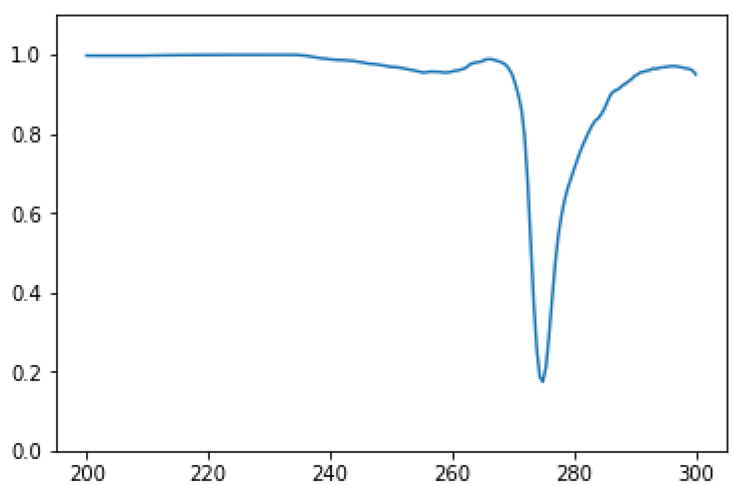

```
Predicted spectrum:
```



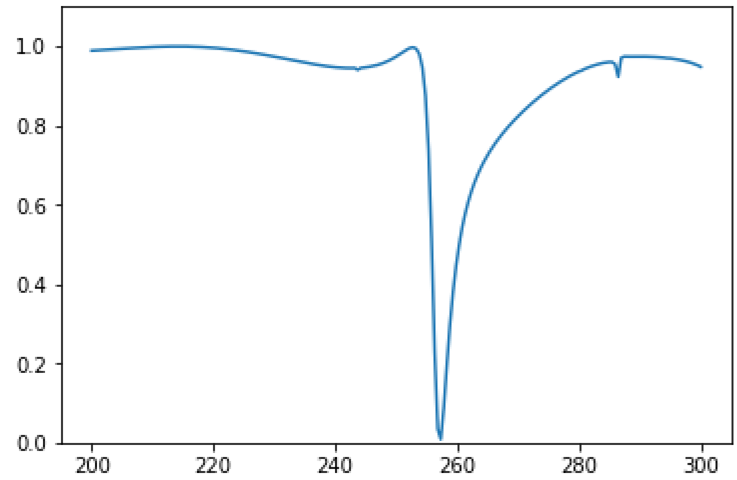

```
Test 46
True spectrum:
```



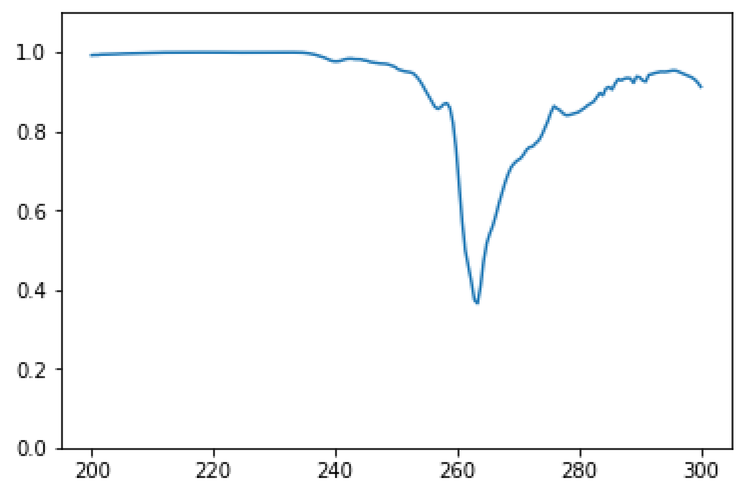

```
Predicted spectrum:
```



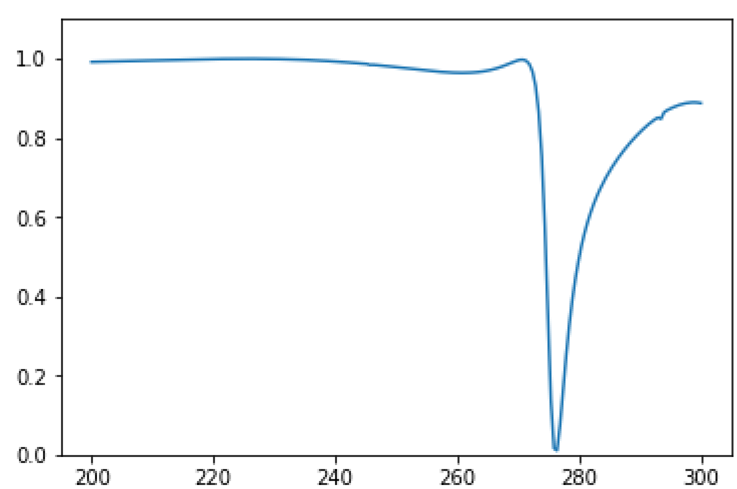

```
Test 47
True spectrum:
```



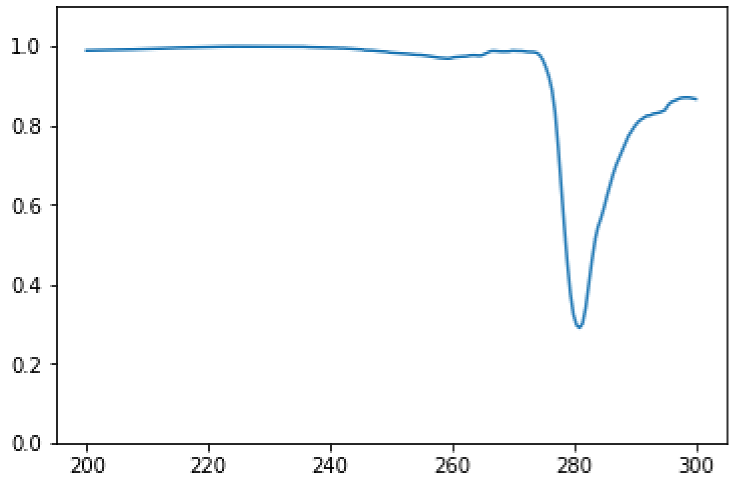
Predicted spectrum:



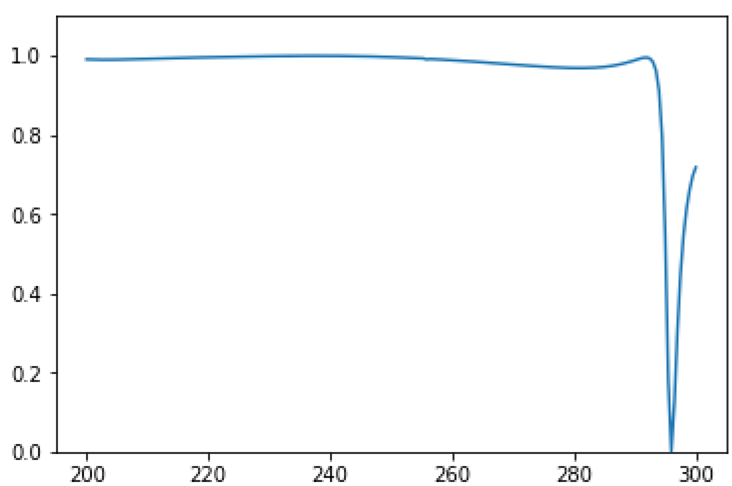
Test 48
True spectrum:



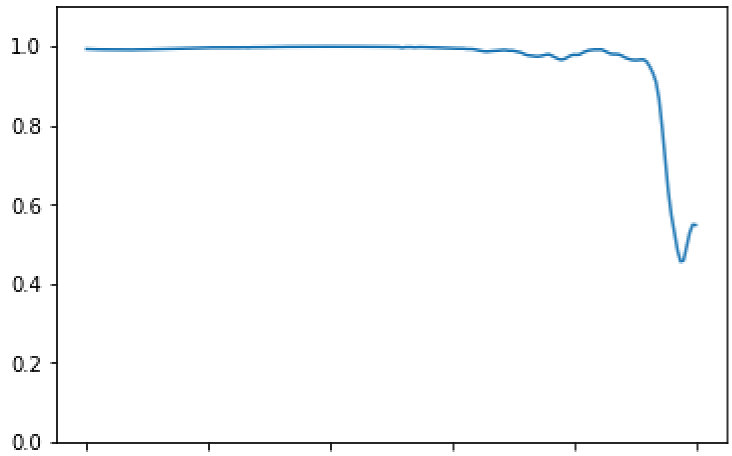
Predicted spectrum:



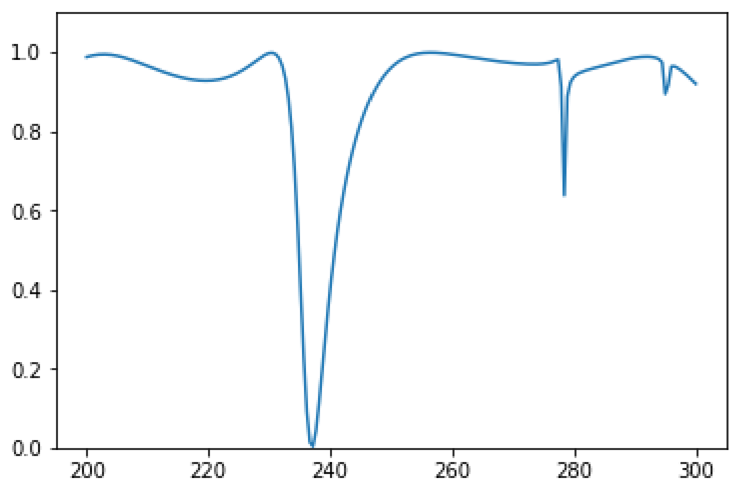
Test 49
True spectrum:



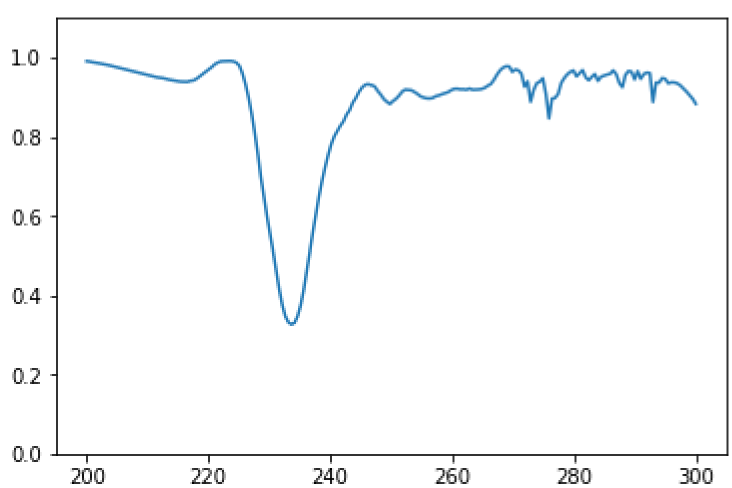
Predicted spectrum:



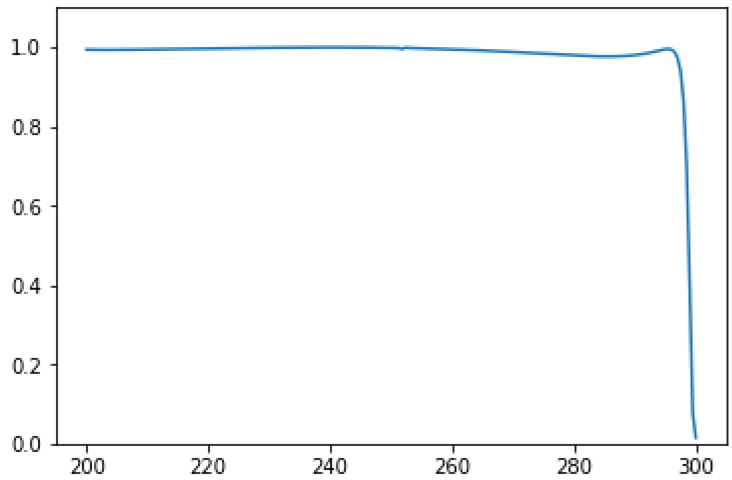

```
Test 50
True spectrum:
```



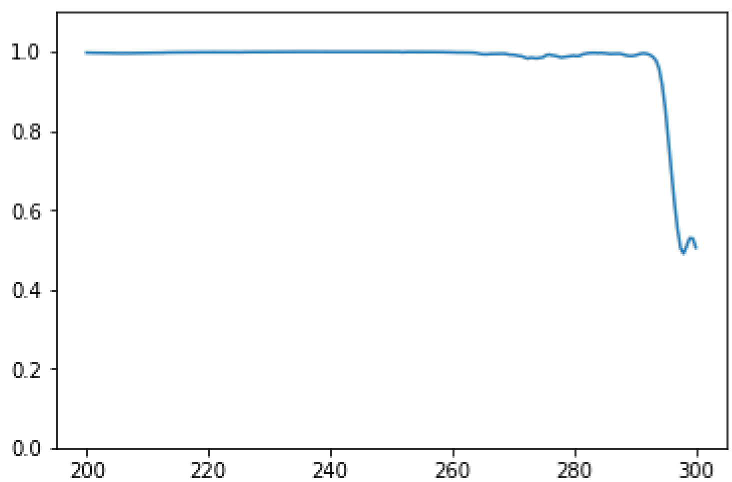

```
Predicted spectrum:
```



```
Test 51
True spectrum:
```

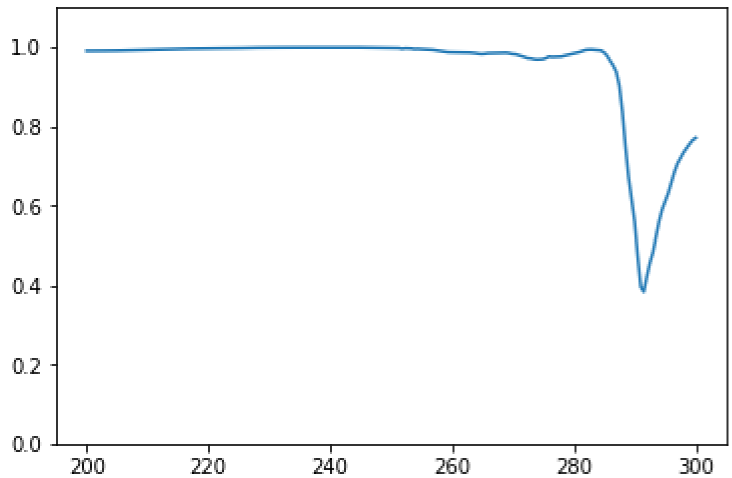



```
Predicted spectrum:
```

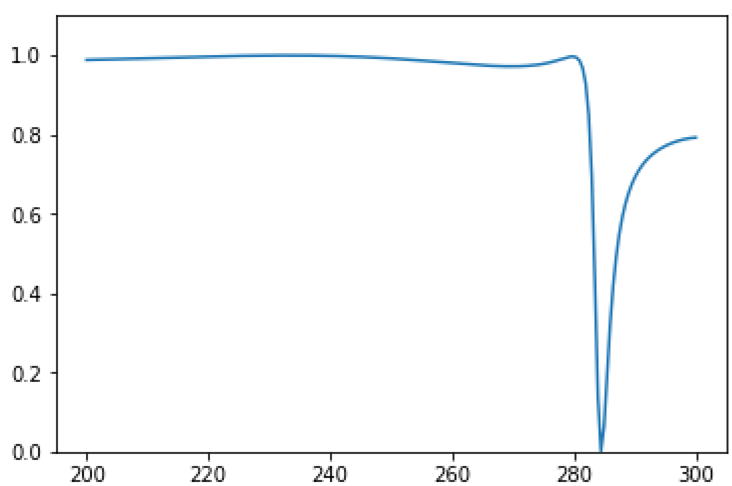



```
Test 52
True spectrum:
```

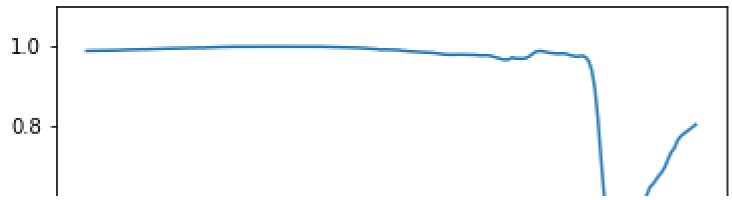
Predicted spectrum:



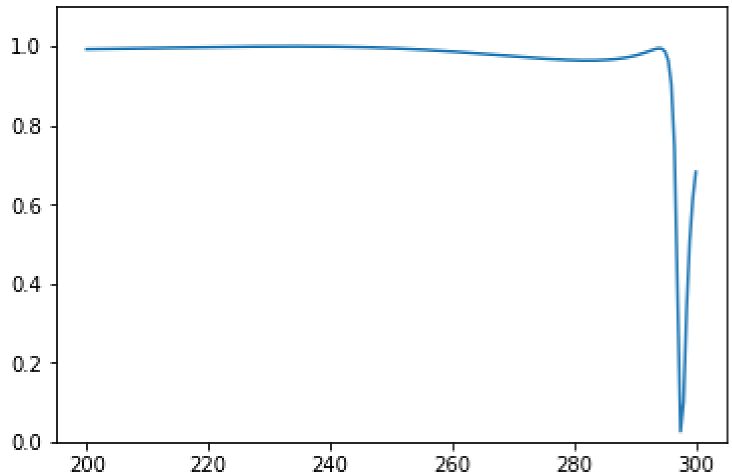
Test 53
True spectrum:



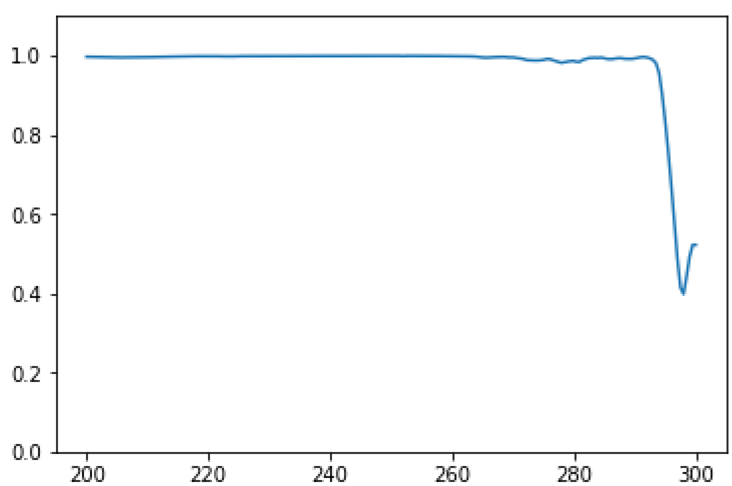
Predicted spectrum:



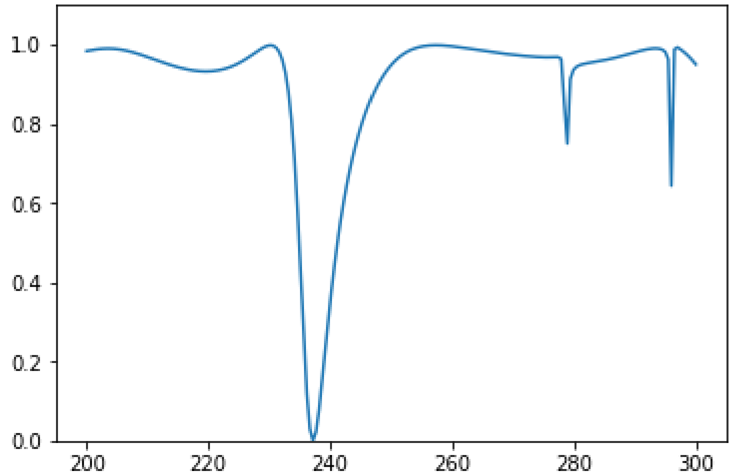

```
Test 54
True spectrum:
```



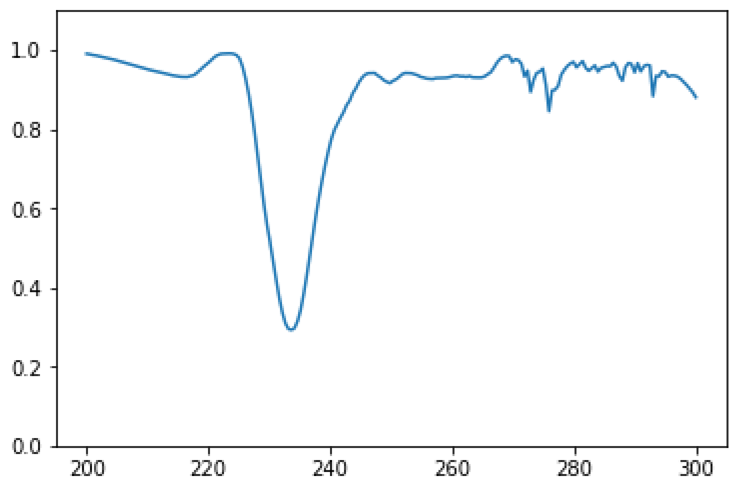

```
Predicted spectrum:
```



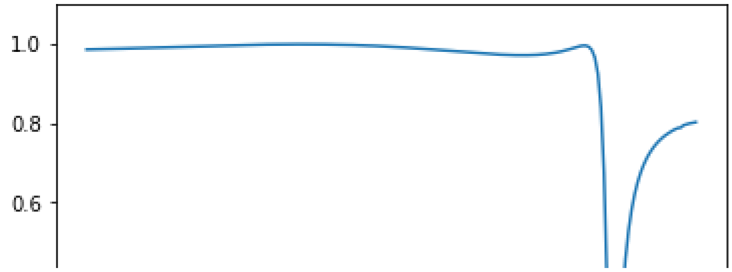
```
Test 55
True spectrum:
```



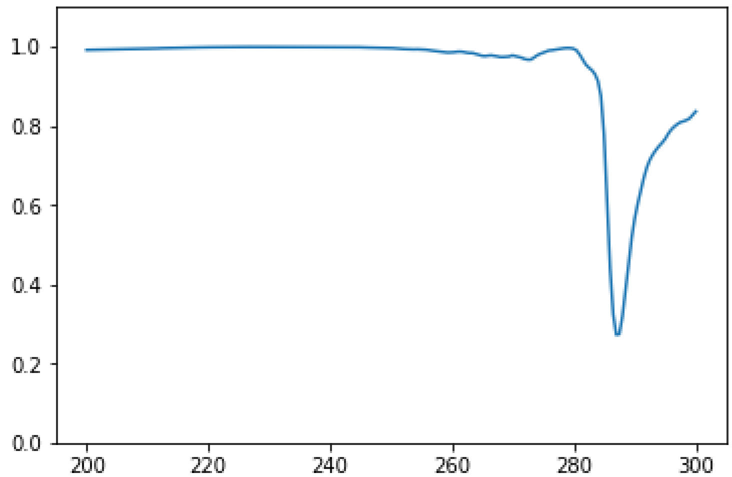

```
Predicted spectrum:
```



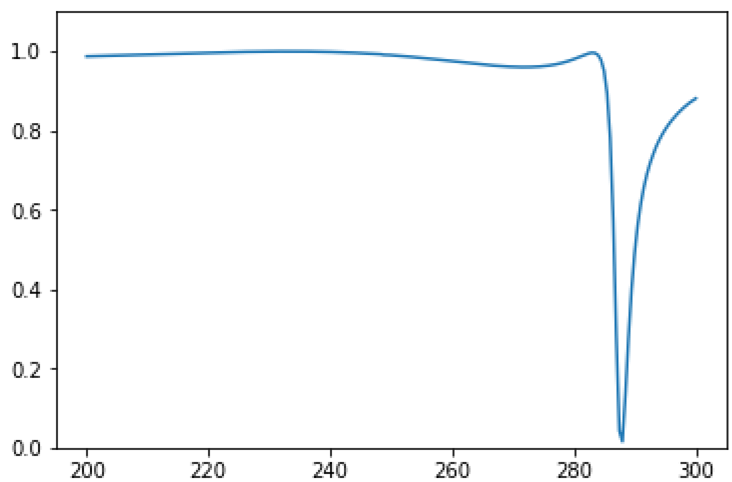

```
Test 56
True spectrum:
```



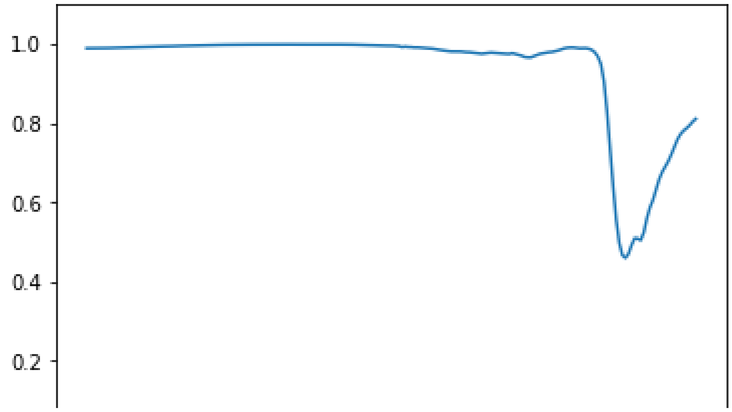
Predicted spectrum:



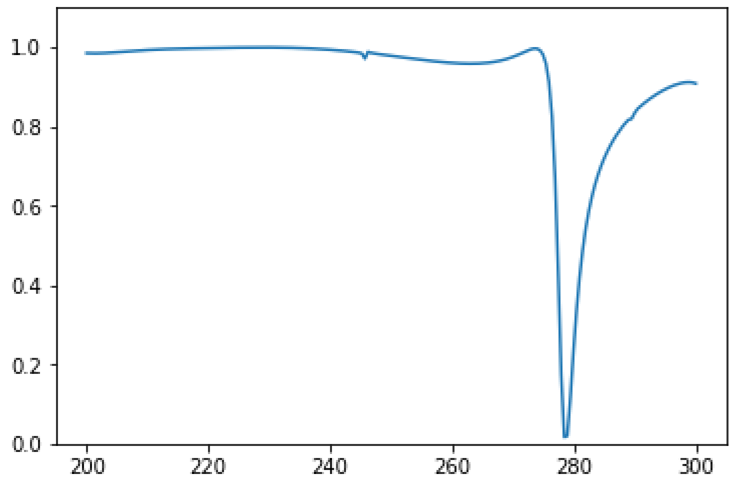
Test 57
True spectrum:



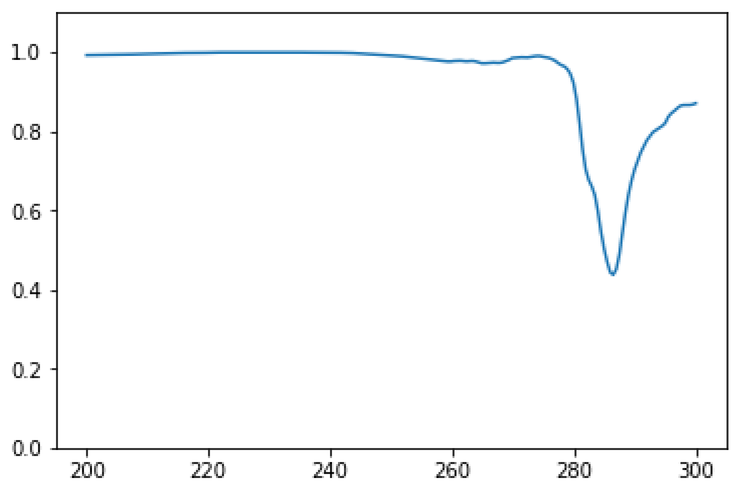
Predicted spectrum:



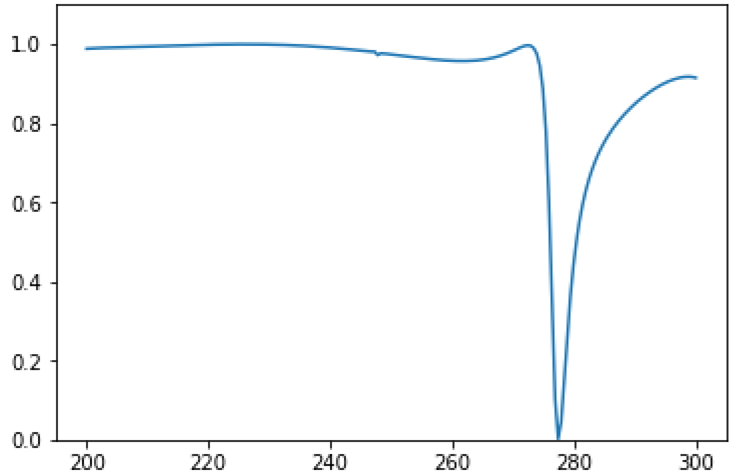

```
Test 58
True spectrum:
```



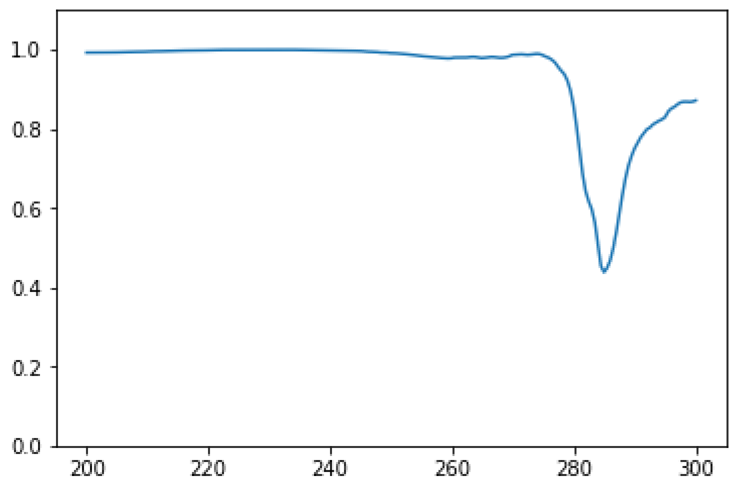

```
Predicted spectrum:
```



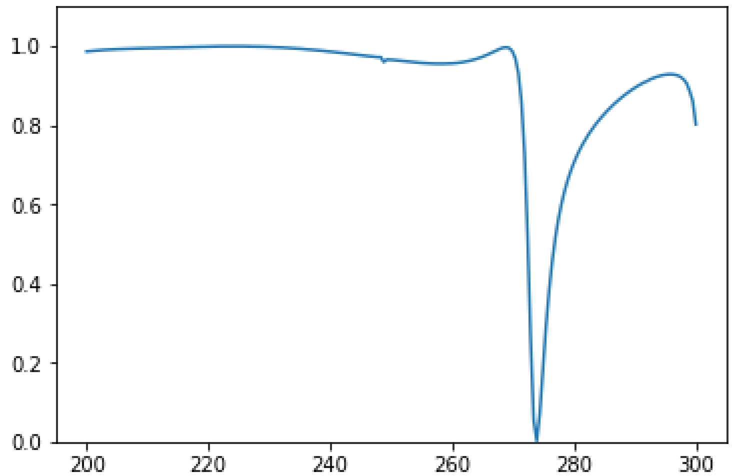
```
Test 59
True spectrum:
```



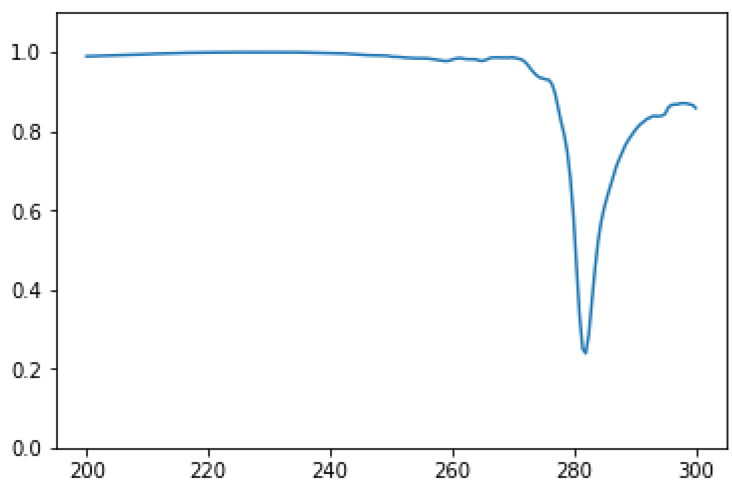

```
Predicted spectrum:
```



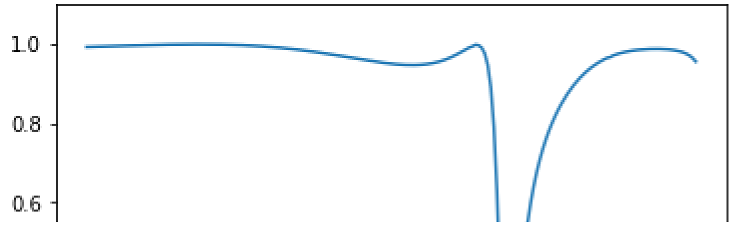

```
Test 60
True spectrum:
```



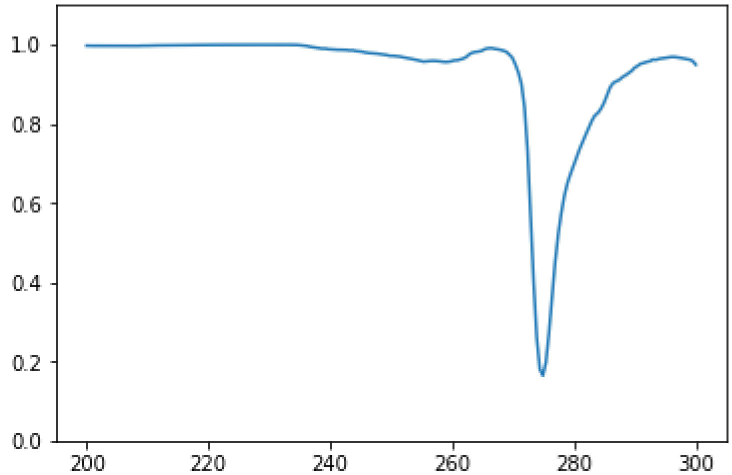
Predicted spectrum:



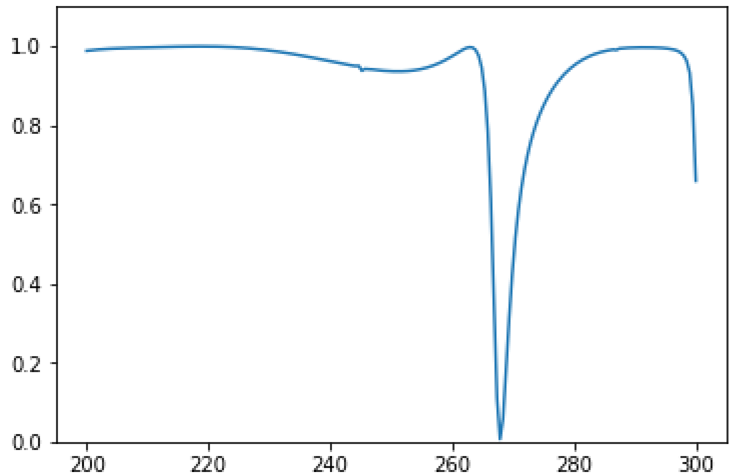
Test 61
True spectrum:



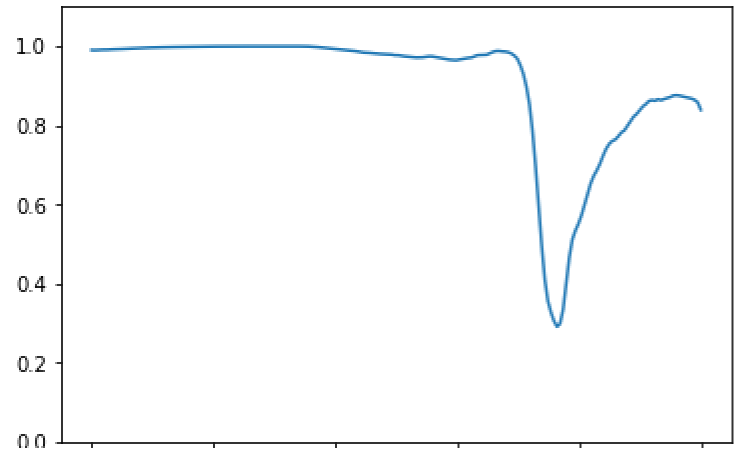

```
Predicted spectrum:
```



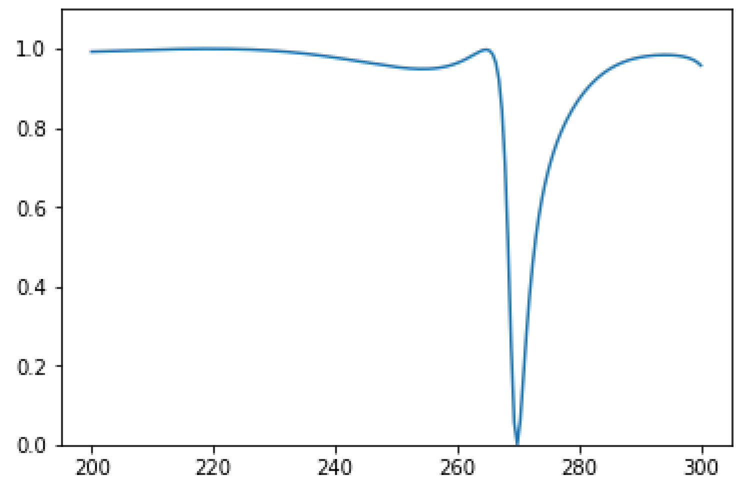

```
Test 62
True spectrum:
```




```
Predicted spectrum:
```

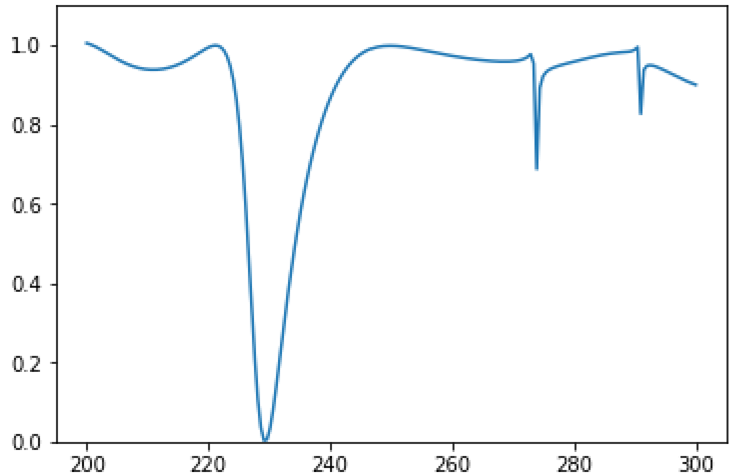



```
Test 63
True spectrum:
```

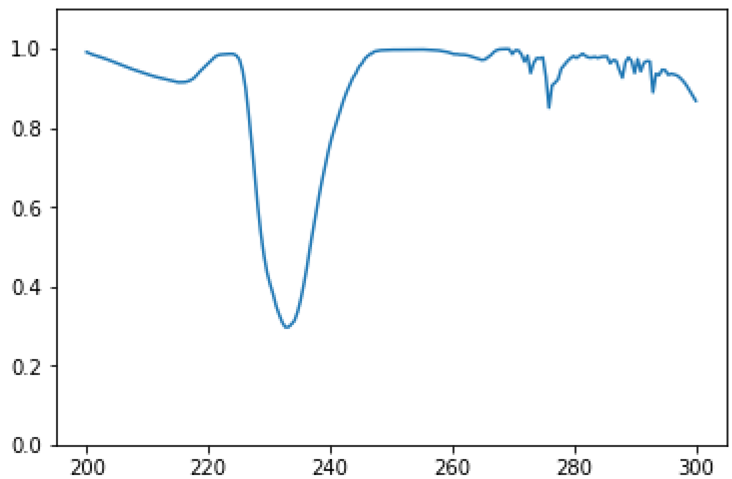



```
Predicted spectrum:
```

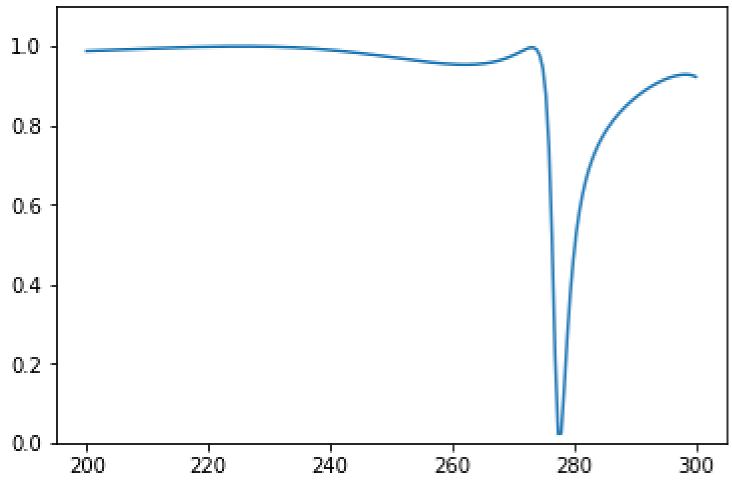
Test 64
True spectrum:



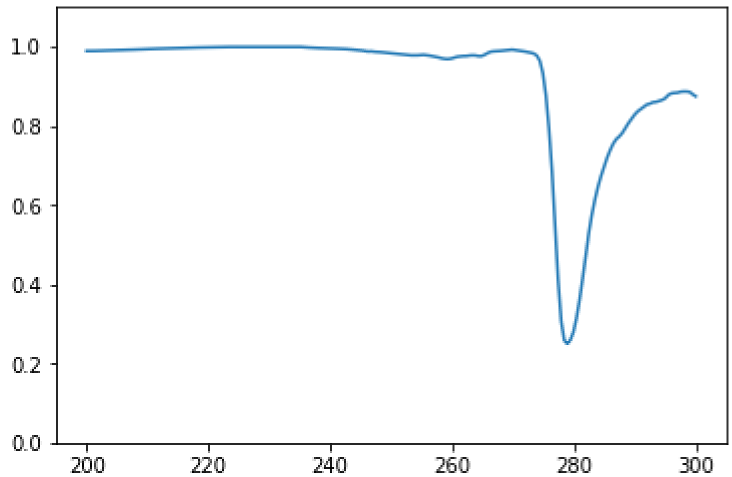
Predicted spectrum:



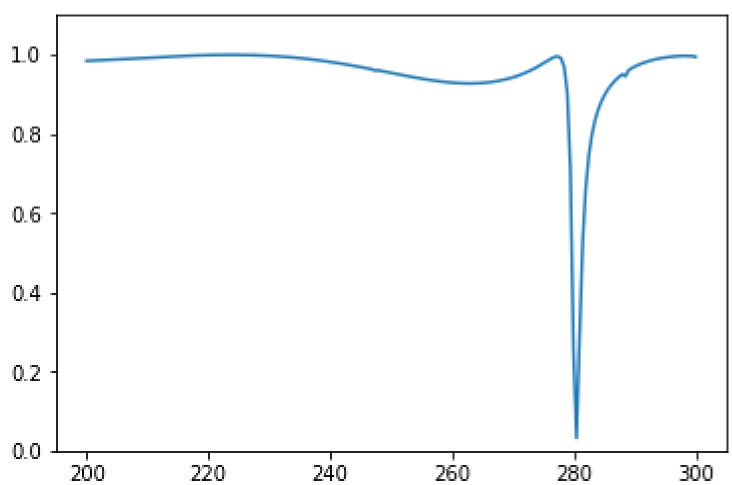
Test 65
True spectrum:



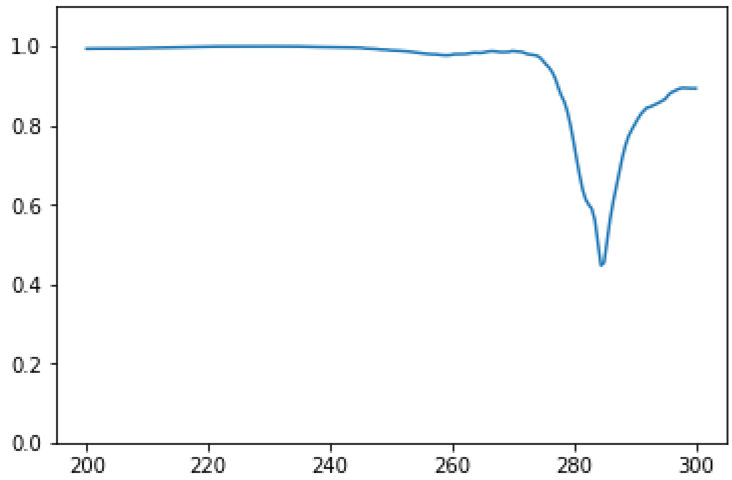
Predicted spectrum:



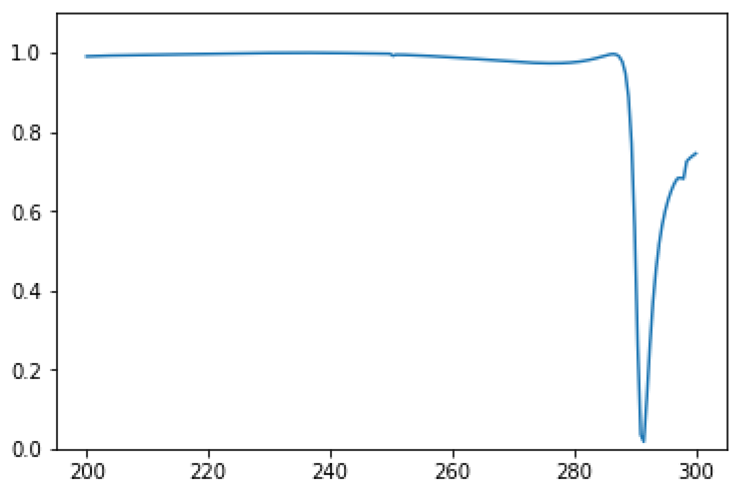
Test 66
True spectrum:



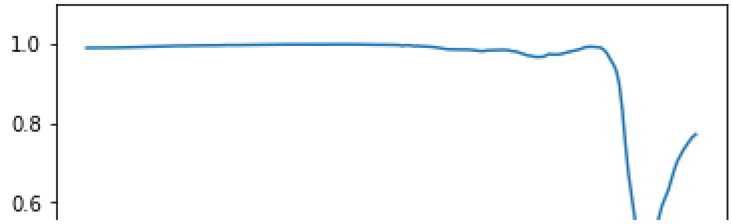
Predicted spectrum:



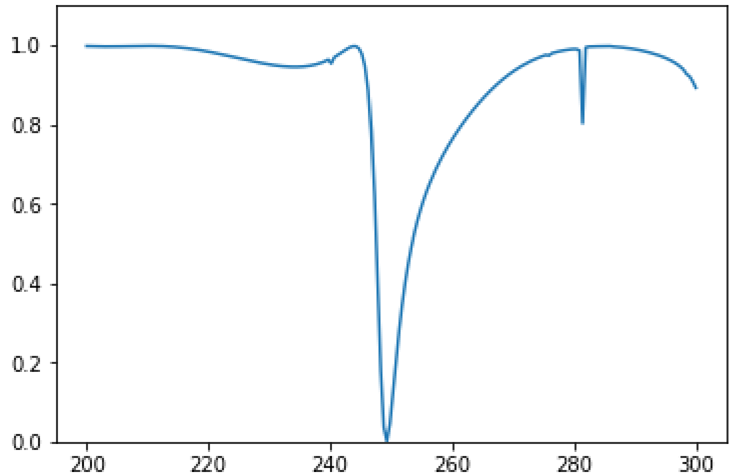

```
Test 67
True spectrum:
```



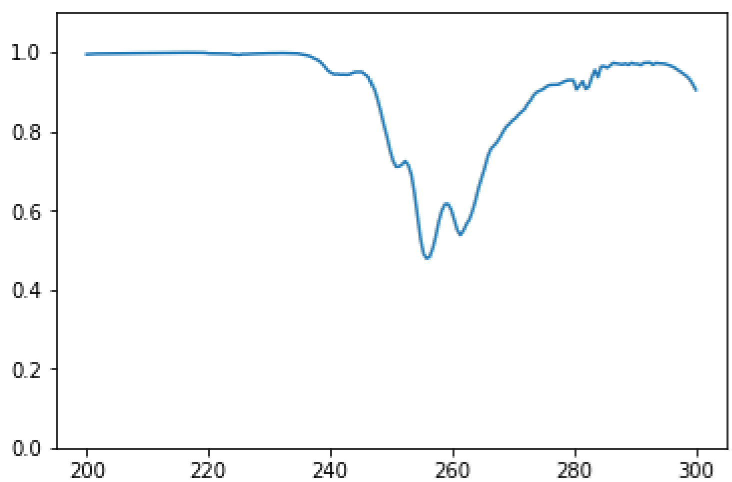

```
Predicted spectrum:
```



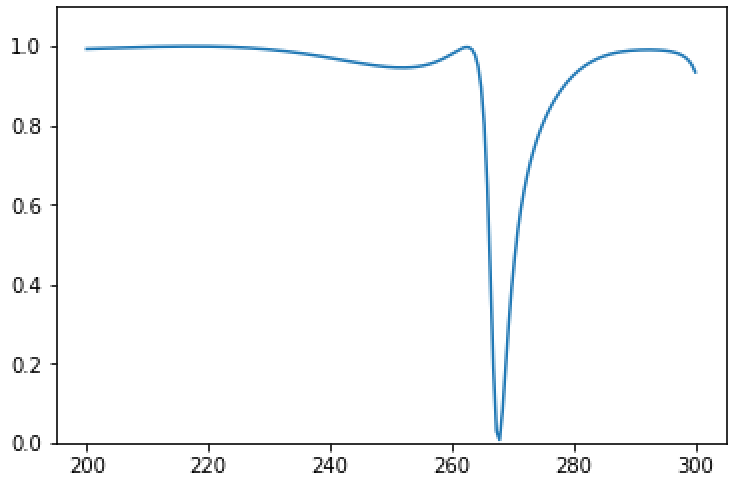
Test 68
True spectrum:



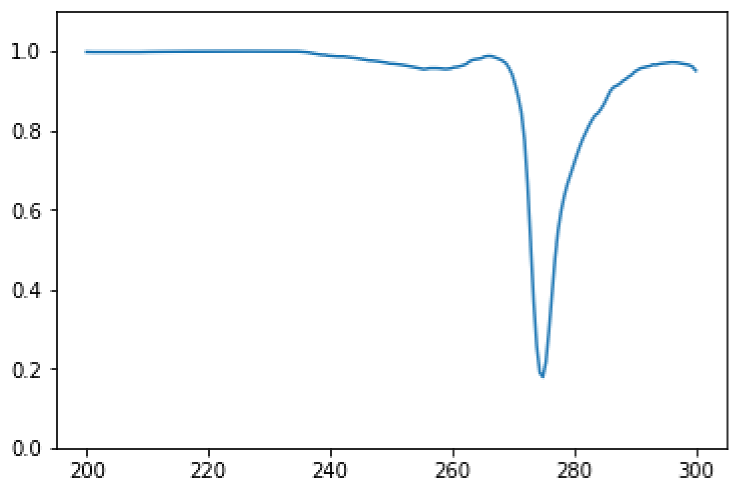
Predicted spectrum:



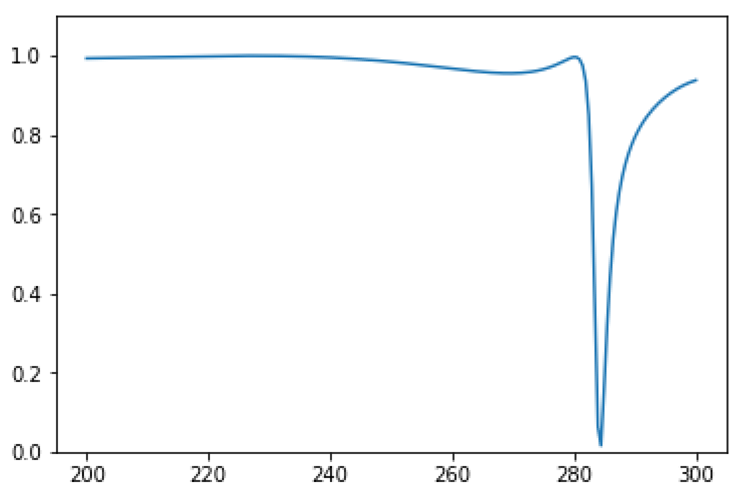
Test 69
True spectrum:



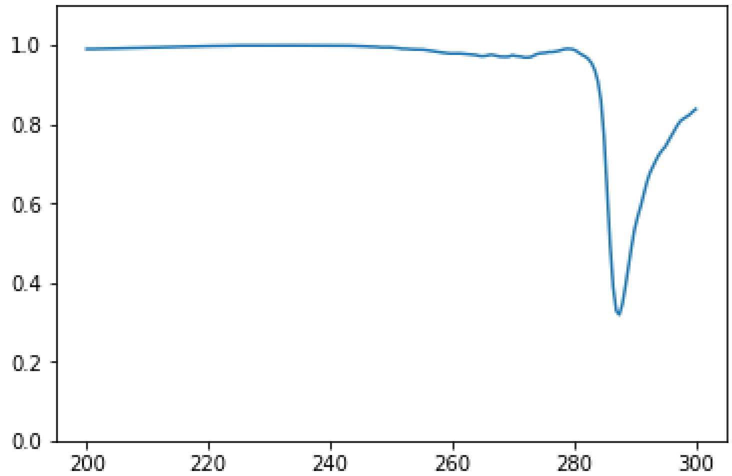
Predicted spectrum:



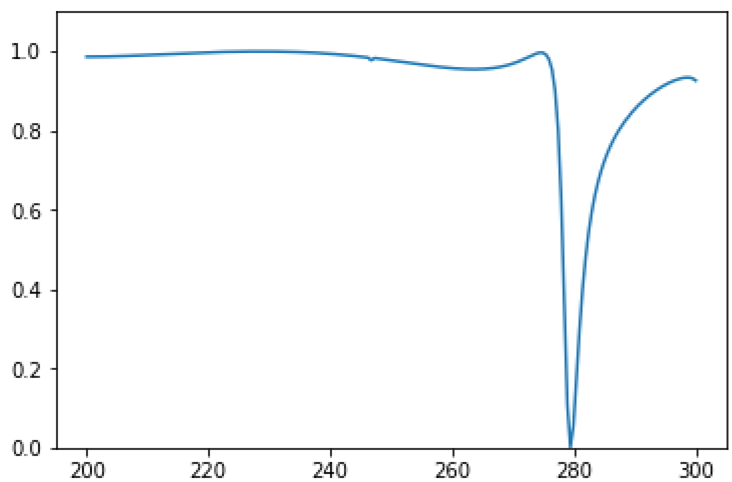
Test 70
True spectrum:



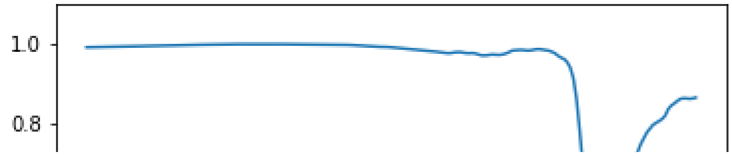
Predicted spectrum:



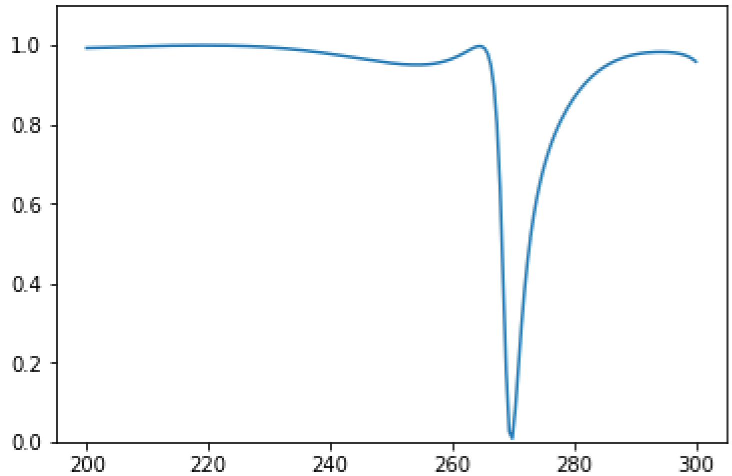

```
Test 71
True spectrum:
```



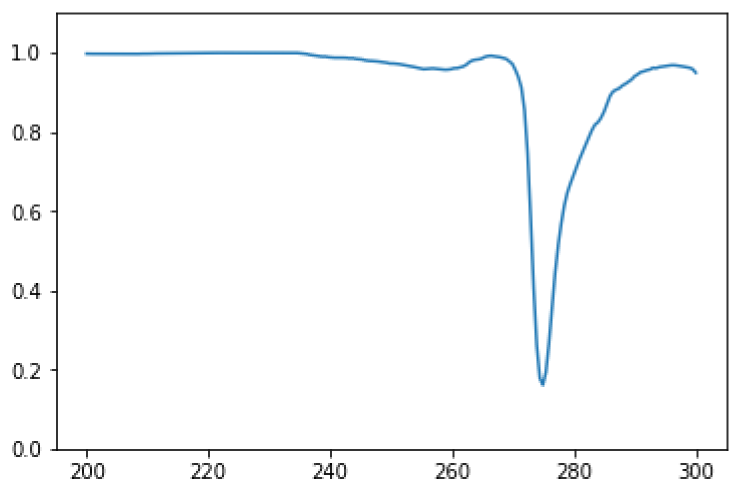

```
Predicted spectrum:
```



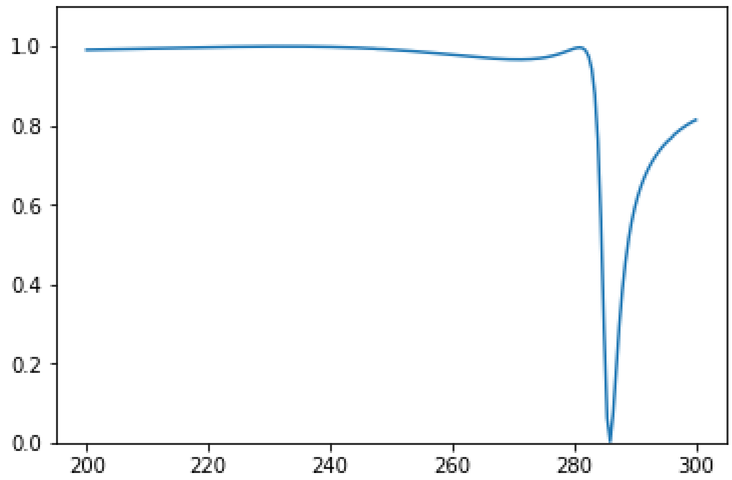

```
Test 72
True spectrum:
```



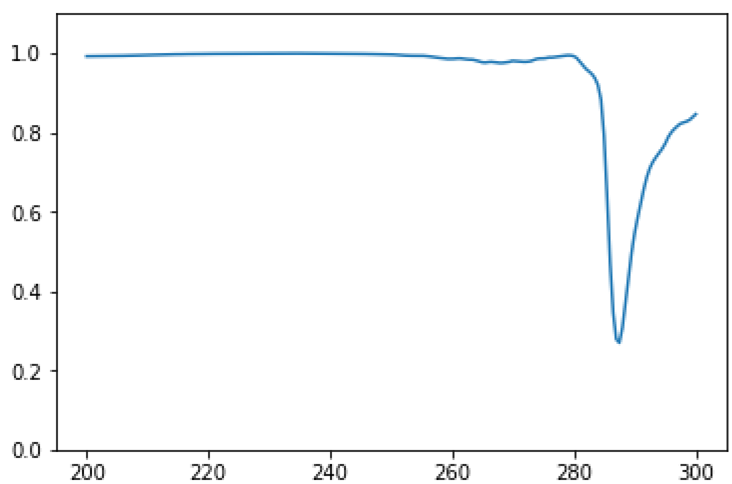

```
Predicted spectrum:
```



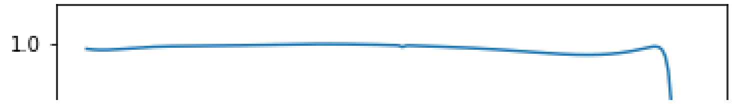

```
Test 73
True spectrum:
```



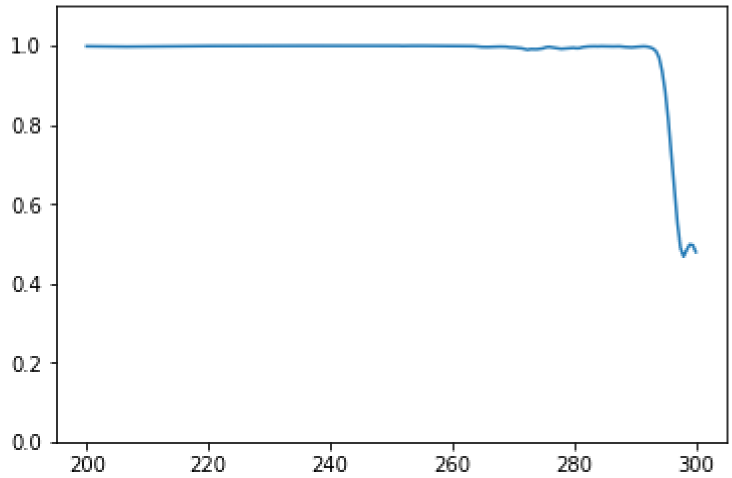

```
Predicted spectrum:
```



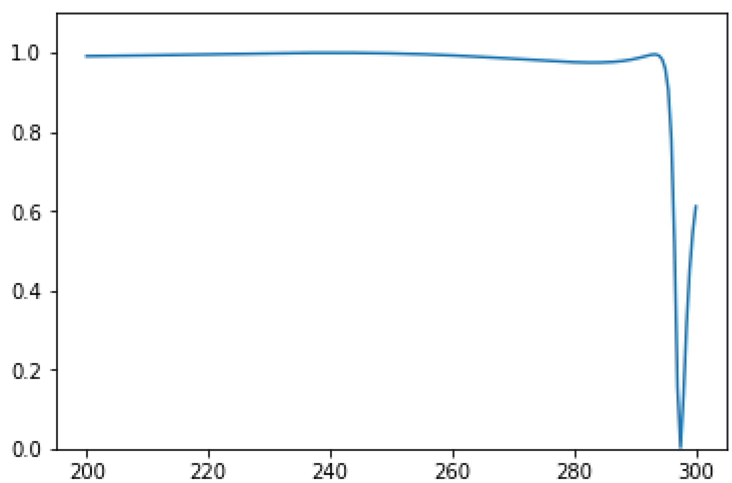

```
Test 74
True spectrum:
```



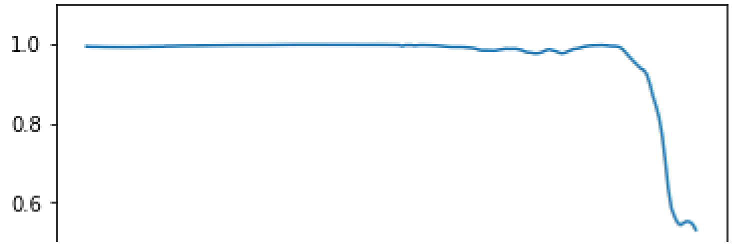

```
Predicted spectrum:
```



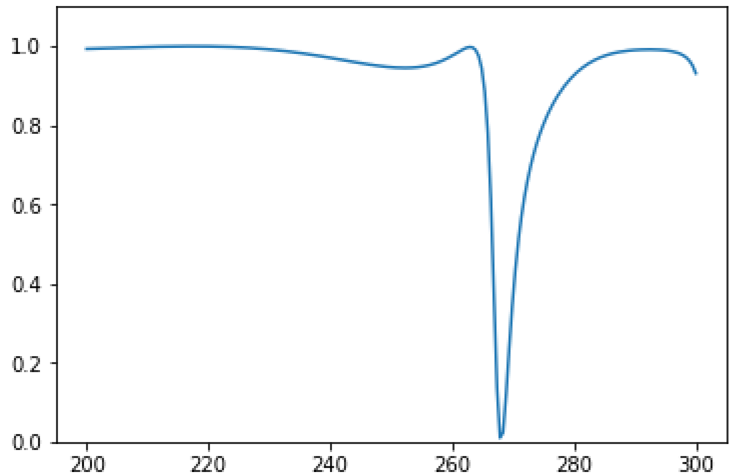

```
Test 75
True spectrum:
```



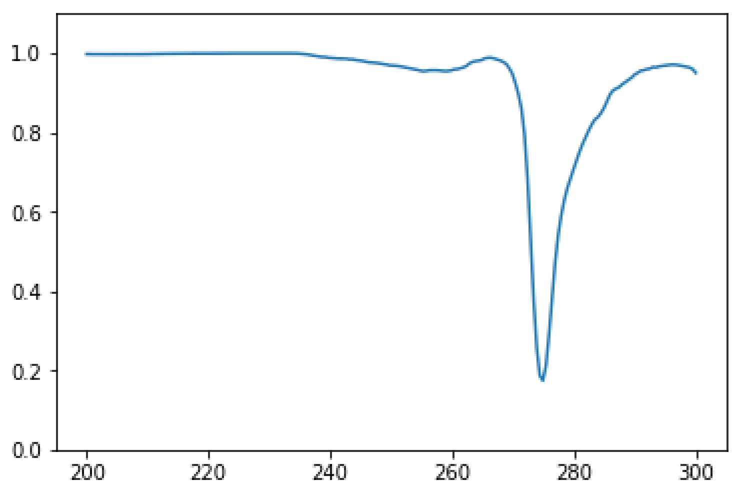

```
Predicted spectrum:
```



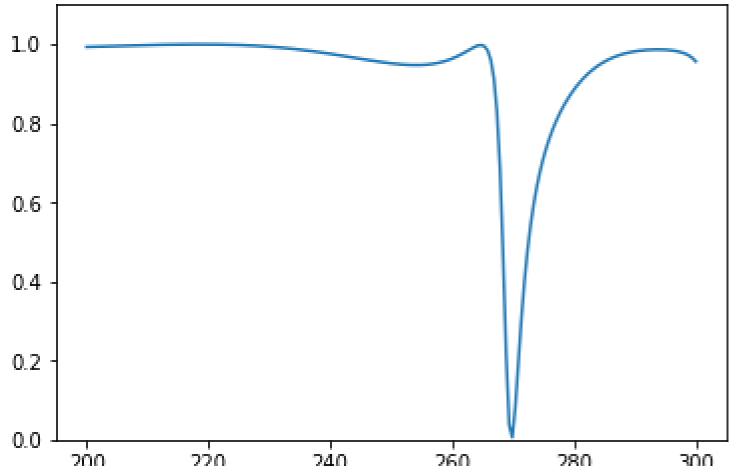
Test 76
True spectrum:



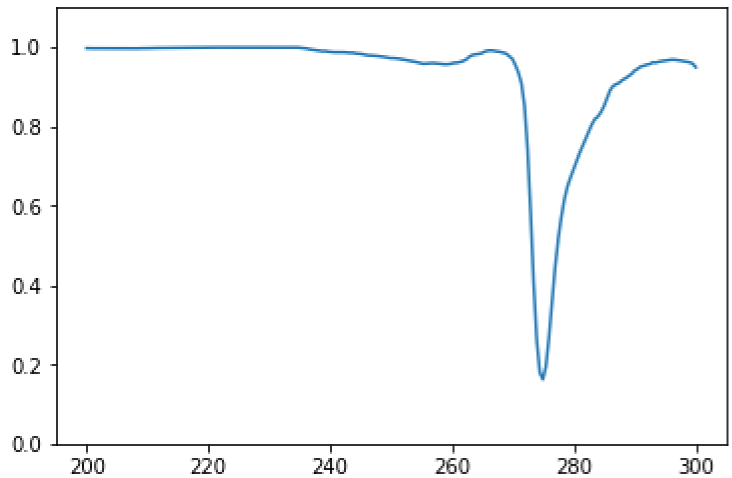
Predicted spectrum:



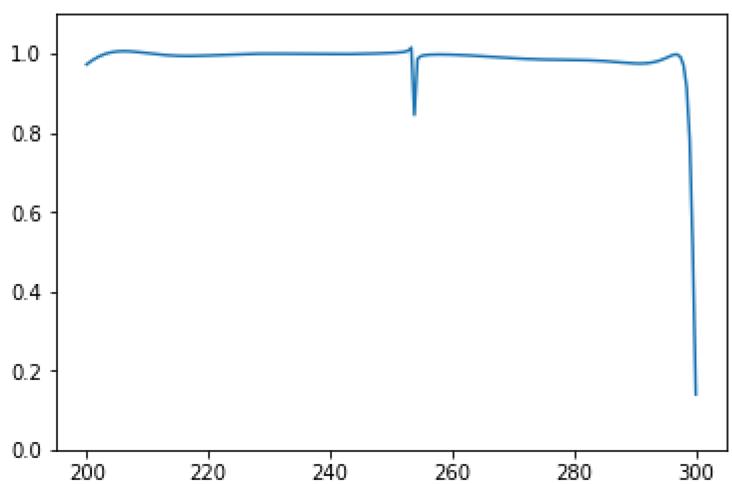
Test 77
True spectrum:



Predicted spectrum:

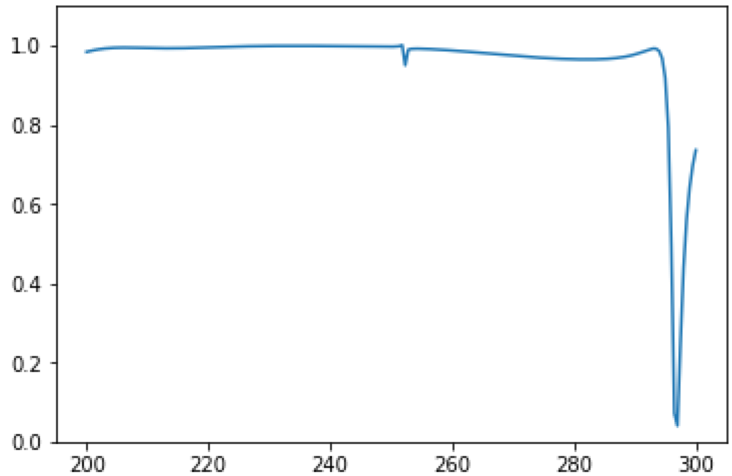


Test 78
True spectrum:

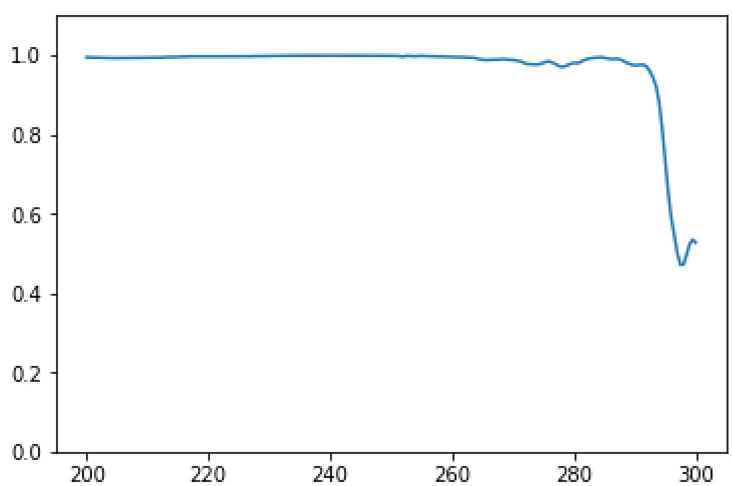


Predicted spectrum:

Test 79
True spectrum:

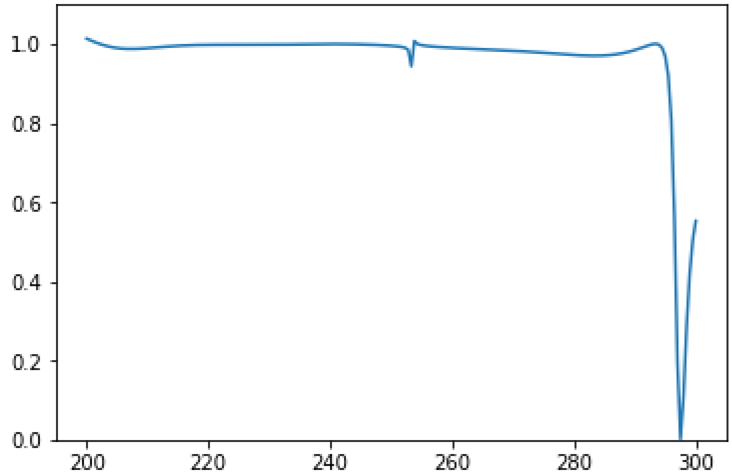


Predicted spectrum:

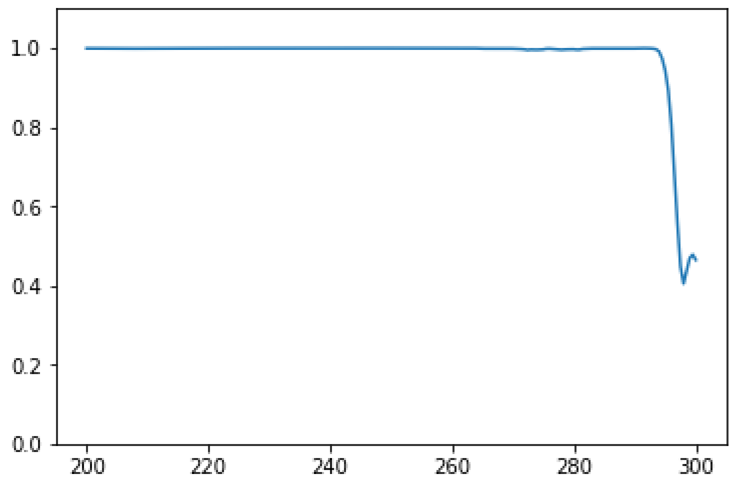


Test 80

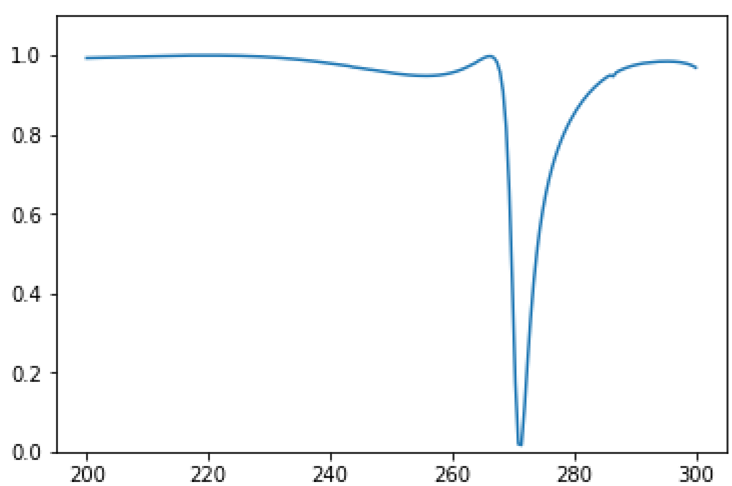
```
True spectrum:
```



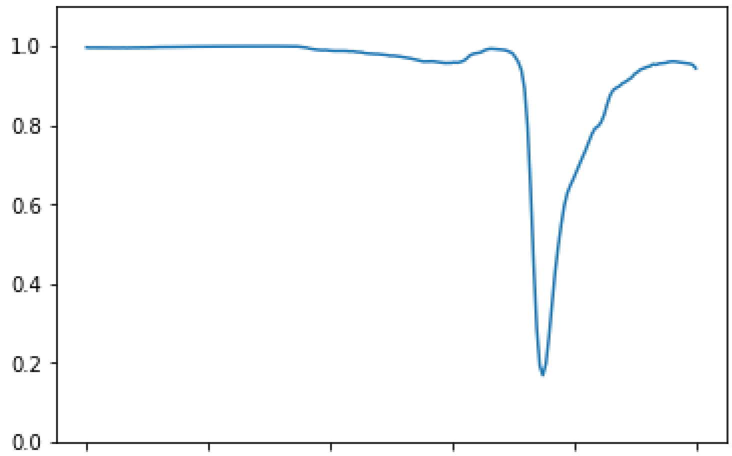

```
Predicted spectrum:
```



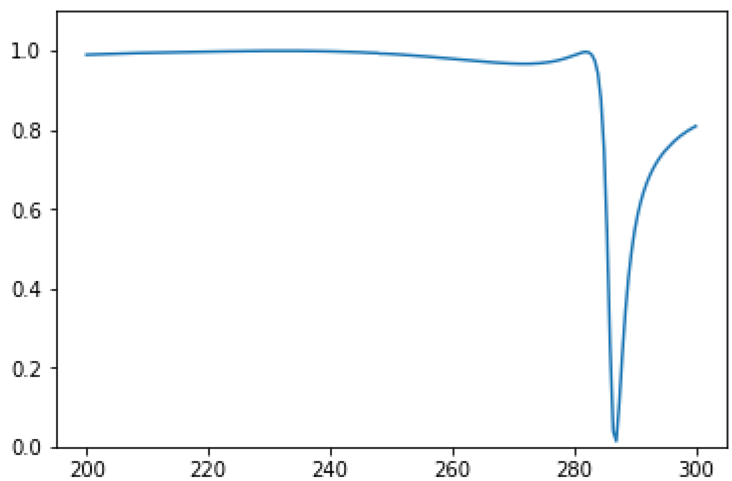

```
Test 81
True spectrum:
```



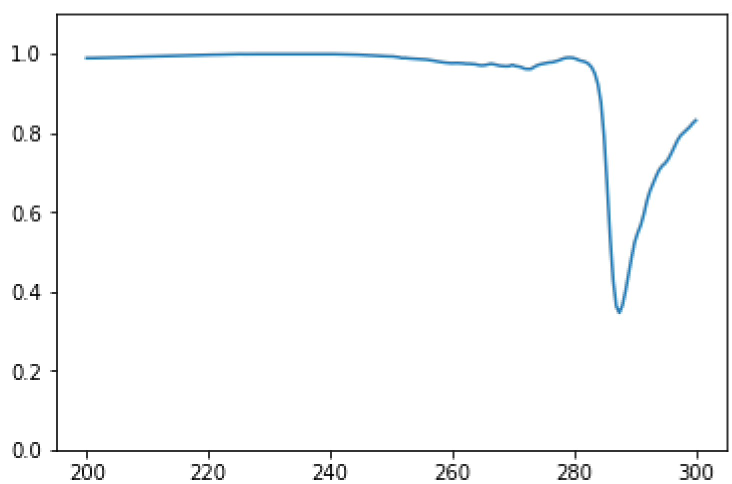

```
Predicted spectrum:
```



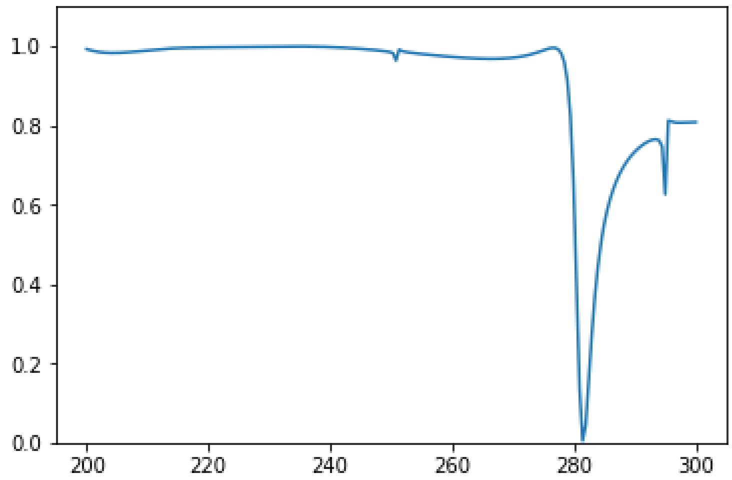

```
Test 82
True spectrum:
```



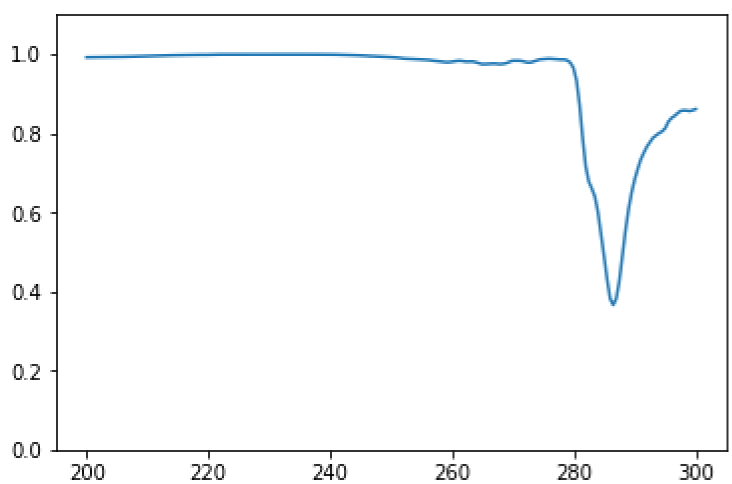

```
Predicted spectrum:
```



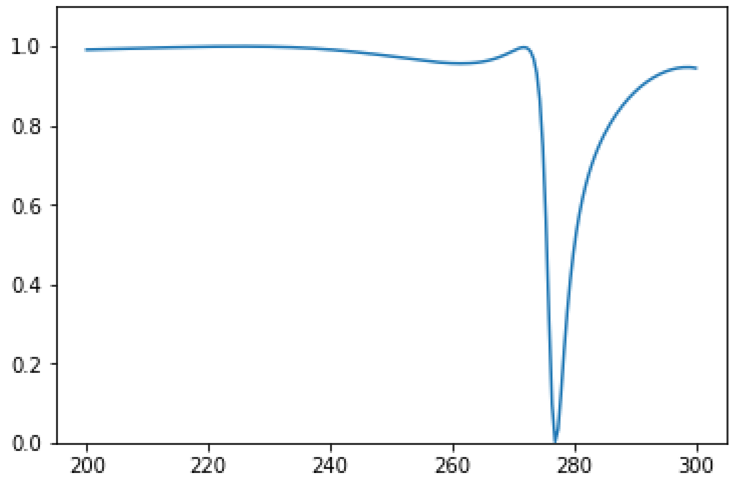

```
Test 83
True spectrum:
```



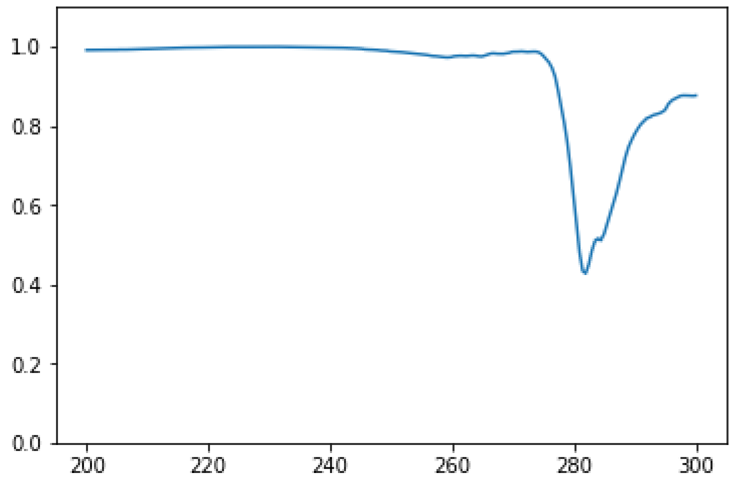
Predicted spectrum:



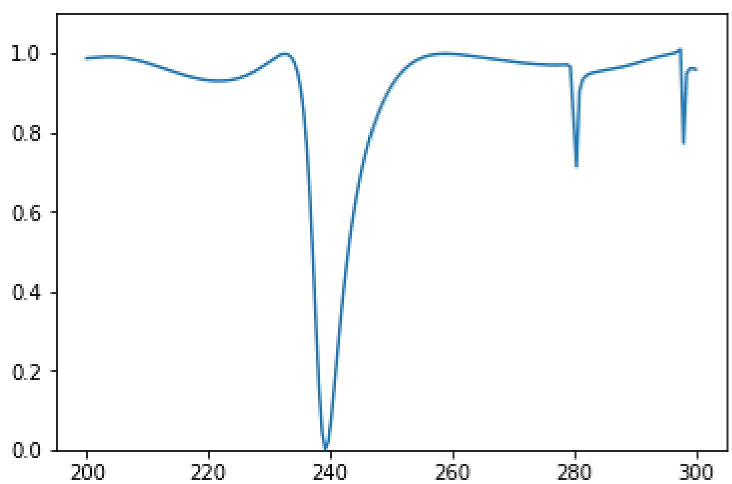
Test 84
True spectrum:



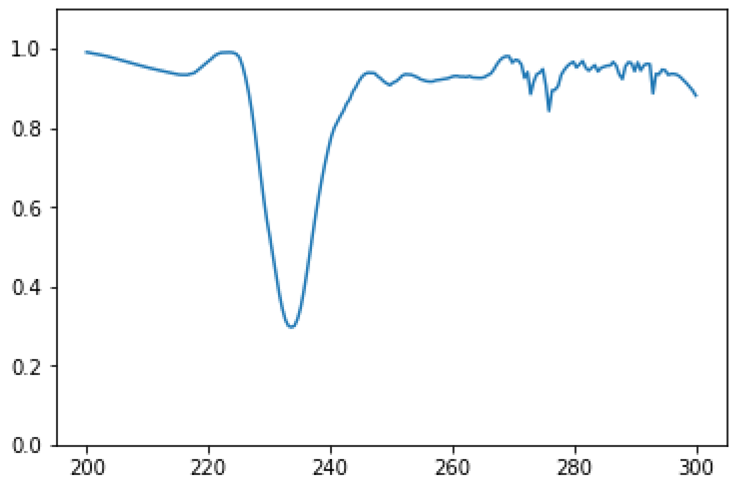
```
Predicted spectrum:
```



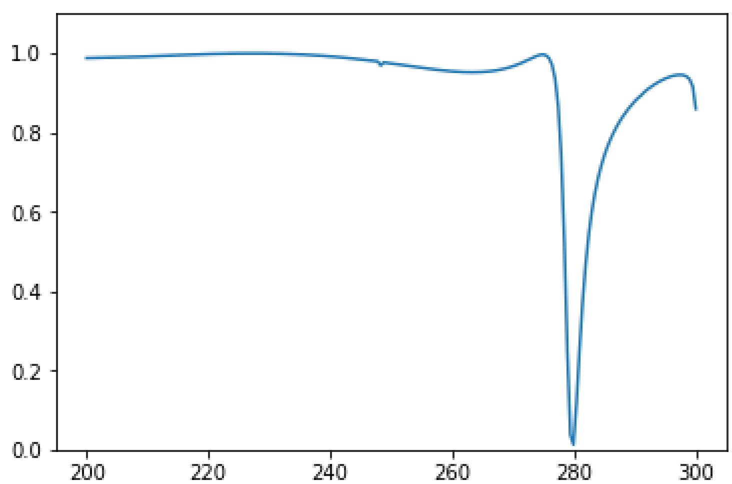
```
Test 85
True spectrum:
```



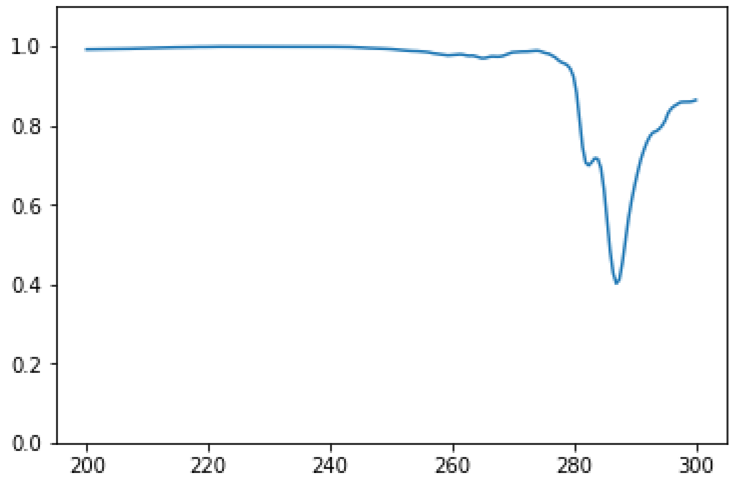
```
Predicted spectrum:
```



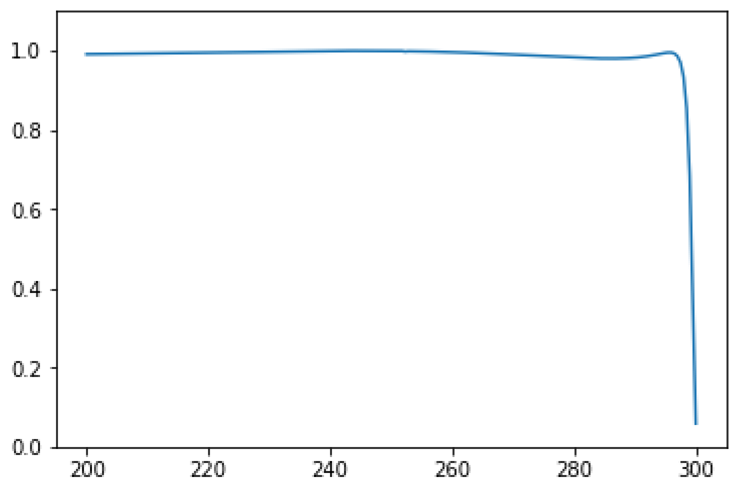

```
Test 86
True spectrum:
```



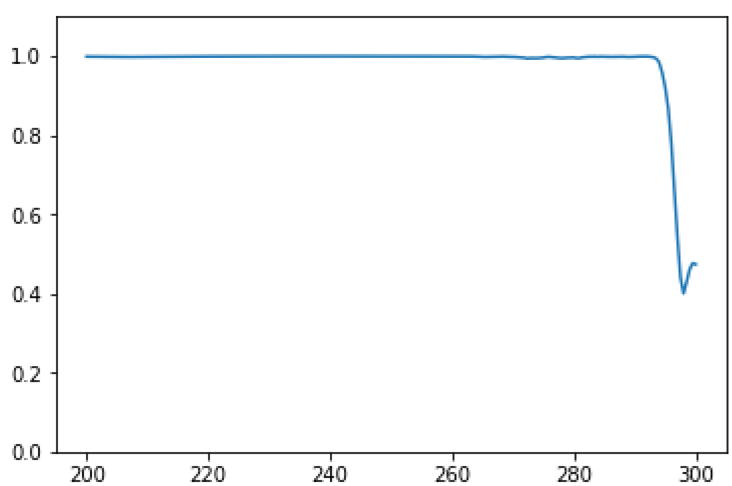

```
Predicted spectrum:
```



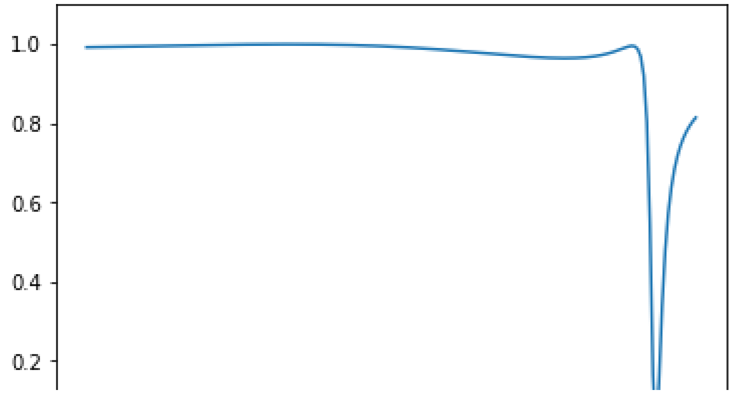

```
Test 87
True spectrum:
```



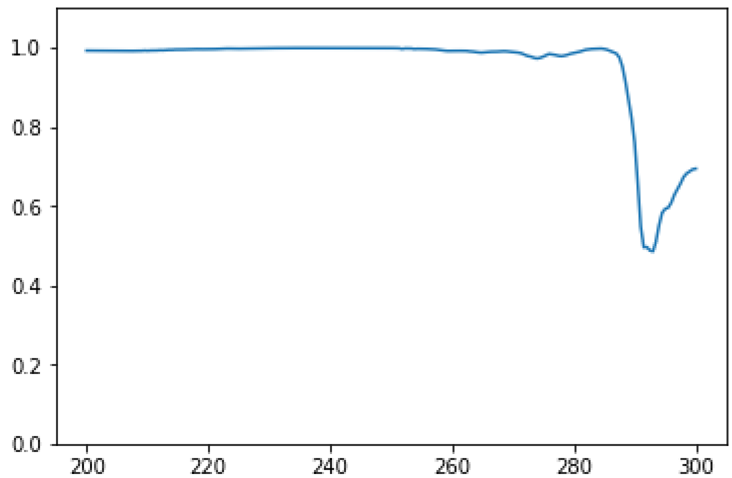

```
Predicted spectrum:
```



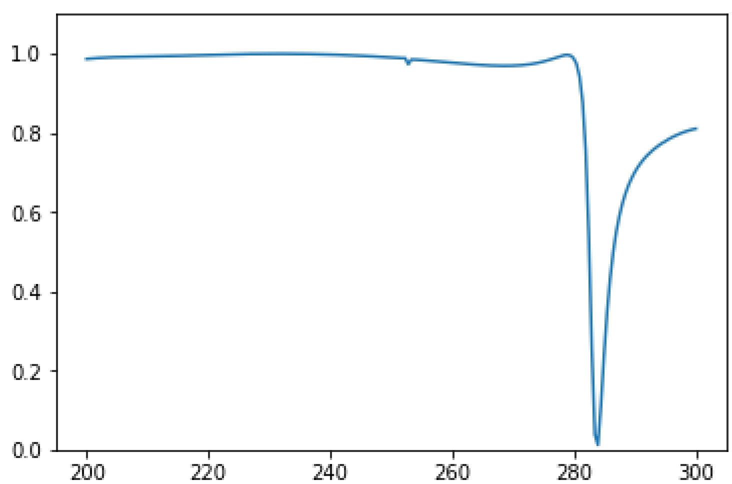

```
Test 88
True spectrum:
```



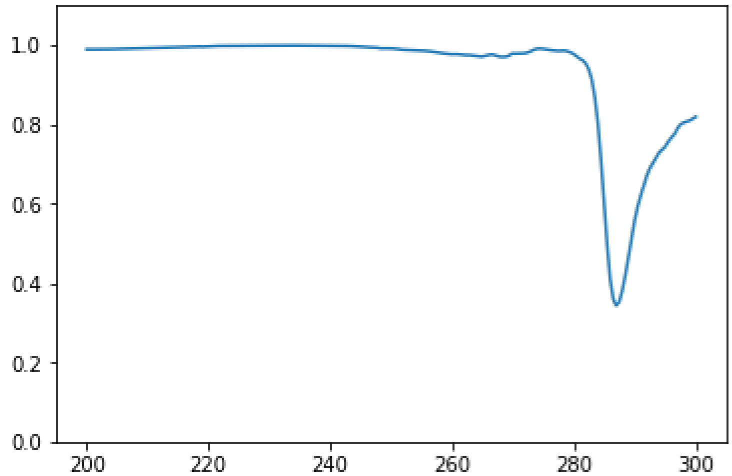
Predicted spectrum:



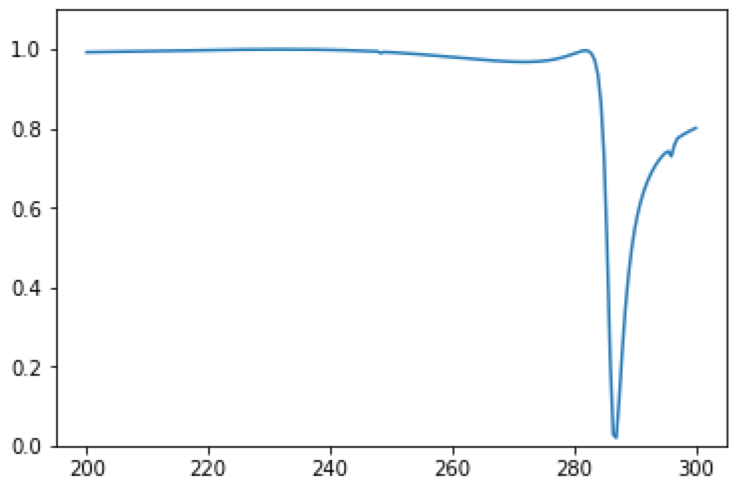
Test 89
True spectrum:



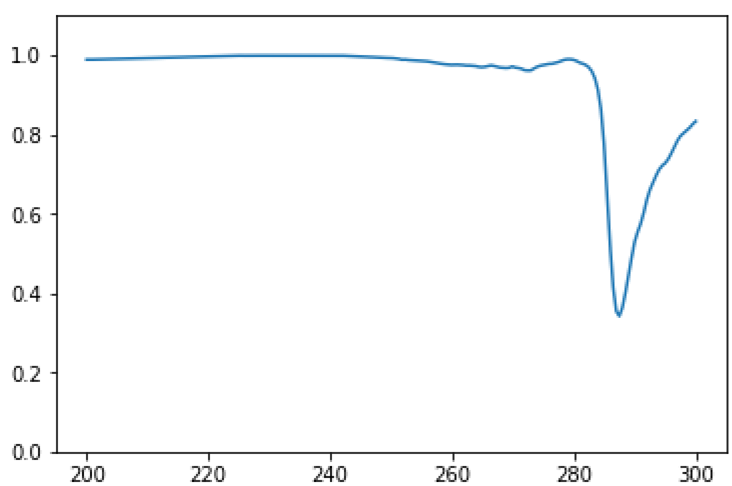
Predicted spectrum:



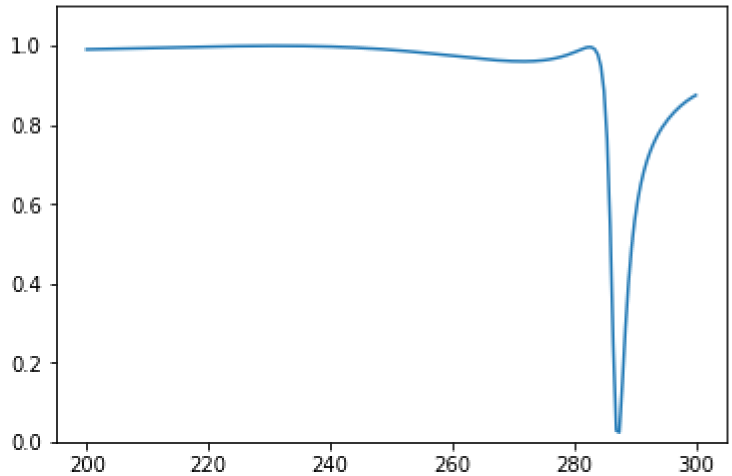
Test 90
True spectrum:



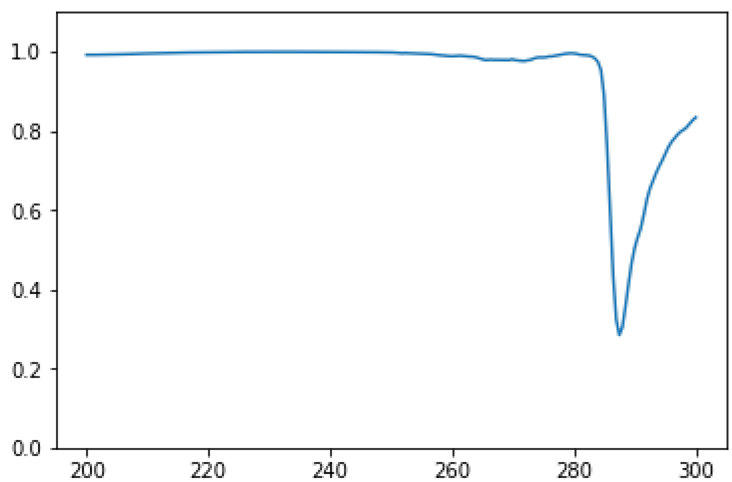
Predicted spectrum:



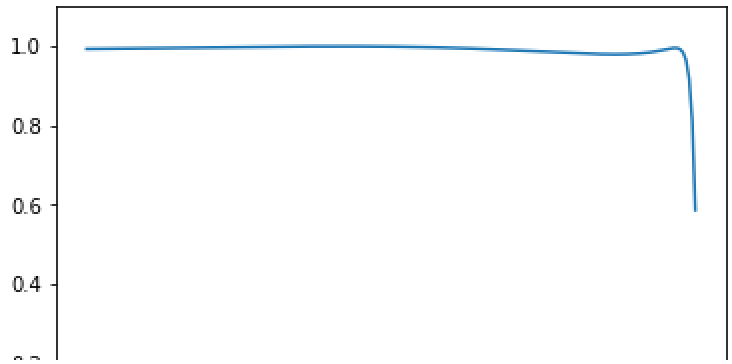
Test 91
True spectrum:



Predicted spectrum:



Test 92
True spectrum:




```
Predicted spectrum:
```