IsingGapClass.py

```python
import bootstrap
import matplotlib.pyplot as plt
import time
import datetime
import numpy as np

#We define a class which imposes a gap in the Z_2-even operator sector.
#The continuum starts at a specified value, and we add an operator between the unitarity bound and this value.
class IsingGap(object):
    bootstrap.cutoff=1e-10
    def __init__(self, gap, sig_values, eps_values):
        #Initialize default input parameters and the gap in the Z2-even operator spectrum.
        self.inputs={'dim': 3, 'kmax': 7, 'lmax': 7, 'nmax': 4, 'mmax': 2}
        self.gap=gap
        self.sig_values=sig_values
        self.eps_values=eps_values

    def plot_grid(self, parameter, table):
        start_time=time.time()
        start_cpu=time.clock()
        allowed_sig=[]
        allowed_eps=[]
        disallowed_sig=[]
        disallowed_eps=[]
        for sig in self.sig_values:
            for eps in self.eps_values:
                sdp=bootstrap.SDP(sig,table)
                sdp.set_bound(0,float(self.gap))
                sdp.add_point(0,eps)
                result=sdp.iterate()
                if result:
                    allowed_sig.append(sig)
                    allowed_eps.append(eps)
                else:
                    disallowed_sig.append(sig)
                    disallowed_eps.append(eps)
        end_time=time.time()
        end_cpu=time.clock()
        run_time=time.strftime("%H:%M:%S",time.gmtime(end_time-start_time))
        cpu_time=time.strftime("%H:%M:%S",time.gmtime(end_cpu-start_cpu))
        plt.plot(allowed_sig,allowed_eps,'r+')
        plt.plot(disallowed_sig,disallowed_eps,'b+')
        plt.title("n_max="+str(parameter)+". Time Taken: "+run_time+". CPU Time: "+cpu_time)
        plt.show()

    def iterate_parameter(self, par, par_range):
        if type(par_range)==int:
            par_range=[par_range]
        start_time=time.time()
        start_cpu=time.clock()
#        sig_set=np.arange(0.5,0.85,0.05)
#        eps_set=np.arange(1.0,2.2,0.2)
#        bootstrap.cutoff=1e-10
```

```
        for x in par_range:
            self.inputs[par]=x
            tab1=bootstrap.ConformalBlockTable(self.inputs[’dim’],self.inputs[’kmax’],self.inputs[’lmax’],self.inputs[’mmax’],self.inputs[’nmax’])
            tab2=bootstrap.ConvolvedBlockTable(tab1)
            self.plot_grid(x,tab2)
        end_time=time.time()
        end_cpu=time.clock()
        run_time=time.strftime("%H:%M:%S",time.gmtime(end_time-start_time))
        cpu_time=time.strftime("%H:%M:%S",time.gmtime(end_cpu-start_cpu))
        print("Run time "+run_time, "CPU time "+cpu_time)

#Instantiate an IsingGap object and use iterate_paramter to plot grids.
sig_set=np.arange(0.5,0.85,0.05)
eps_set=np.arange(1.0,2.2,0.2)
ising_gap=IsingGap(3.0, sig_set, eps_set)
n_range=np.arange(1,4,1)
ising_gap.iterate_parameter(’nmax’,n_range)
```