```
'''
Created on 27 Oct 2014

@author: bob
'''

from workers.worker import Worker as worker
import scipy
from scipy.signal import argrelextrema



# Global parameters to tweak the algorithm
DIST = 1
GUESS_W = 10
UPPERZERO = 0.05
PRECISION = 400



class Worker2(worker):
    '''
    A class to represent a worker to find the eigenmodes as
    a function of K^2, sigma and g
    '''
    def __init__(self, Ksqr=[1],sigma=[1],g=[1],y0=[0.,1.],n=3,t0=0,tend=1
        '''
        The constructor to set up the right parameters and to create
        the ode's
        '''
        super(Worker2, self).__init__(Ksqr, sigma, g, y0, n, t0, tend, h)
        self.f = open(filename, 'w')
        self.filename = filename
        self.name = name

    def search(self,Ksqrnum,sigmanum,gnum,n):
        '''
        A method to search for the first n eigen modes, the hard way.
        This method will first guess the value of w. And based on the numb
        modes it will find the value for tune thise guess.
        If it finds a good value for w it will search for the neigbour val
        for w who are eigenmodes.
        '''
        self.tempKsqrnum = Ksqrnum
        self.tempsigmanum = sigmanum
        self.tempgnum = gnum
        N = 100
        #print w
```

```
49          fx = scipy.zeros(N)
50          nb_of_eigen = 0
51          count = 0
52          solutionW = []
53          while nb_of_eigen < n:
54              count = count + 1
55              w = scipy.logspace(0.1*count,10*count,N,base=0.5)
56              count = count+1
57              for x in xrange(N):
58                  fx[x] = self.endPoint(w[x])
59              index_local = argrelextrema(scipy.absolute(fx), scipy.less)[0]
60              solutW = scipy.zeros(len(index_local))
61              #matplotlib.pyplot.show(block=False)
62              for i in xrange(len(index_local)):
63                  solutW[i] = w[index_local[i]]
64              nb_of_eigen = len(index_local) + nb_of_eigen
65              solutionW = scipy.append(solutionW, solutW, 1)
66              #matplotlib.pyplot.draw()
67          #matplotlib.pyplot.show(block=False)
68          return solutW[0:n]
69
70      def getAnswer(self):
71          if (self.spectrum==None):
72              raise RuntimeError('No spectrum calculated')
73          return self.spectrum
74
75
76
77
78
```