```
'''
Created on 08 Nov 2014

@author: bob
'''
from workers.worker import Worker as worker
import scipy
from scipy.optimize import anderson
#from scipy.optimize import newton_krylov

GUESS_W = 1

class WorkerNLS(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(WorkerNLS, 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):
        '''
        Search for the first n roots.
        '''
        guess = GUESS_W
        self.tempKsqrnum = Ksqrnum
        self.tempsigmanum = sigmanum
        self.tempgnum = gnum
        if n ==1:
            orde = 1
            while(orde!=0):
                oneOnGuess = 1./guess
                print 'Root guess: %s'%(1./oneOnGuess)
                root = (1./scipy.absolute(anderson(self.aid_f, oneOnGuess,
                info = self.zero_point_info(Ksqrnum,sigmanum,gnum,root)
                orde = info[0]
                print 'Root and order: %s , %s'%(root,orde)
                guess = root*(orde + 1)
        else:
            pass


```

```
49          return [root]
50      def aid_f(self,oneonguess):
51          return self.endPoint(scipy.absolute(1/oneonguess))
52
53      def _foundAll(self,Nroots):
54          for root in Nroots:
55              if (root==0):
56                  return False
57          return True
58      def _nextGuess(self,Nroots,guess,previous=None):
59          '''
60          A method to do an educated guess for the next omega value
61          '''
62          # Find index of lowest 0
63          index = 0
64          for root in Nroots:
65              if (root==0):
66                  break
67              index = index +1
68          print '-'*40
69          print previous
70          print index
71          print '-'*40
72          if (previous!=0 and previous==index):
73              return (guess/2,index)
74          newguess = 0
75          count = 1
76          nonzero = 0
77          for root in Nroots:
78              newguess = newguess + root/count
79              count = 1 + count
80              if (root!=0):
81                  nonzero = nonzero +1
82          if nonzero!=0:
83              newguess  = (newguess/nonzero)/(index+1)
84              return (newguess,index)
85          else:
86              return (guess/2,index)
87
88
89
90
91
92
93
94
95
96
```

```
97
98
99
100
101
```