```
'''
Created on 15 Oct 2014

@author: bob


A test file for the worker class.

'''

from workers.workerSimple import workerSimple as workerSimple
import system.waveSystem as wave
import function.function as func
import integrators.rungeKutta as rK
import matplotlib.pyplot as plt

Ksqr = [1]
sigma = [1]
g = [1]
n = 10
y0 = 0
t0 = 1
tend = 1
h = 0.01

#wsqrnum = 0.3-0.285
wsqrnum = 0.015*(1+23./50)
wsqrnum = wsqrnum*(1+7./50)
wsqrnum = 0.001

class rho0(func.Function):
    def __init__(self,Ksqr,sigma,g,wsqr):
        func.Function.__init__(self)
        self.Ksqr = Ksqr
        self.sigma = sigma
        self.g = g
        self.wsqr = wsqr
    def evaluate(self, x):
        return (1+self.sigma*x)
# Create the two objects to represent the functions P and Q
class P(func.Function):
    def __init__(self,Ksqr,sigma,g,wsqr):
        func.Function.__init__(self)
        self.Ksqr = Ksqr
        self.sigma = sigma
        self.g = g
        self.wsqr = wsqr
    def evaluate(self, x):
```

```
49          return self.wsqr*rho0(self.Ksqr,self.sigma,self.g,self.wsqr).evalu
50  class Q(func.Function):
51      def __init__(self,Ksqr,sigma,g,wsqr):
52          func.Function.__init__(self)
53          self.Ksqr = Ksqr
54          self.sigma = sigma
55          self.g = g
56          self.wsqr = wsqr
57      def evaluate(self, x):
58          return -self.Ksqr*(rho0(self.Ksqr,self.sigma,self.g,self.wsqr).eva
59                             rho0(self.Ksqr,self.sigma,self.g,self.wsqr).der
60
61
62  def plot_ode(Ksqr, sigma, g, wsqr):
63      funcP = P(Ksqr,sigma,g,wsqr)
64      funcQ = Q(Ksqr,sigma,g,wsqr)
65      vgl = wave.WaveSystem(funcP,funcQ)
66      fe = rK.RungeKutta(vgl)
67      t_runge,soln_runge = fe.integrate(y0,t0,tend,h)
68      solution_runge = [soln_runge[i][0] for i in range(len(soln_runge))]
69      plt.plot(t_runge,solution_runge)
70      plt.show()
71      pass
72
73
74  def nbOfZerosnumber_of_zeros():
75      eigenSys = workerSimple(Ksqr,sigma,g,y0,n,t0,tend,h)
76      print eigenSys.zero_point_info(Ksqrnum=1, sigmanum=1, gnum=1, wsqrnum=
77
78  def find_eigen_mode():
79      w = wsqrnum
80      eigenSys = workerSimple(Ksqr,sigma,g,y0,n,t0,tend,h)
81      nb_of_zero , index_last_min , end_point , length_Set = eigenSys.zero_p
82      newW = w
83      while (nb_of_zero==0):
84          newW = (newW+0.0)/2
85          nb_of_zero , index_last_min , end_point , length_Set = eigenSys.ze
86      print nb_of_zero , index_last_min , end_point , length_Set
87
88      print newW
89      previousW = newW
90      counter = 0
91      while(abs(end_point)>0.001):
92          counter = counter +1
93          nb_of_zero_new , index_last_min_new , end_point_new , length_Set_n
94          print counter, newW , nb_of_zero_new , index_last_min_new , end_po
95          if (abs(end_point_new)<0.001):
96              break
```

```
97          if (nb_of_zero_new>=nb_of_zero):
98              # Vergroot w
99              previousW = newW
100             newW = newW*(1+((length_Set-index_last_min_new)+0.0)/index_las
101         if (nb_of_zero_new < nb_of_zero):
102             # Nu weten we dat het vorige punt wel nog achter het nulpunt l
103             # gemmidelde tussen het slechte punt en het vorige goede punt.
104             newW = ((newW+previousW)+0.0)/2
105     print newW
106     plot_ode(Ksqr=1, sigma=1, g=1, wsqr=newW)
107 
108 def task():
109     eigenSys = workerSimple(Ksqr,sigma,g,y0,n,t0,tend,h)
110     eigenSys.task()
111     pass
112 
113 if __name__ == '__main__':
114     Ksqr = [1]
115     sigma = [1]
116     g = [1]
117     n = 10
118     y0 = 0
119     t0 = 1
120     tend = 1
121     h = 0.1
122     Ksqr=[1];sigma=[1];g=[1];y0=[0.,1.];n=10;t0=0;tend=1;h=0.01
123     #nbOfZerosnumber_of_zeros()
124     #find_eigen_mode()
125     #plot_ode(Ksqr=1, sigma=1, g=1, wsqr=wsqrnum)
126     task()
127 
128 
129 
130 
```