

---

---

Assignment 7 - Harmonic Oscillator - III(Matching)

---

---

**SGTB Khalsa College, University of Delhi**  
**Preetpal Singh(2020PHY1140)(20068567043)**

**Unique Paper Code: 32221501**

**Paper Title: Quantum Mechanics and Applications**

**Submitted on: August 24, 2022**

**B.Sc(H) Physics Sem V**

**Submitted to: Dr. Mamta**

## Assignment - 7 (Harmonic Oscillator) (Matching)

Date: .../.../.....

(a) Explain why correct asymptotic solutions are not obtained when you integrate the Schrodinger from  $x_{\min}$  to  $x_{\max}$ .

Ans. Since, the solution of Schrodinger wave equation of Harmonic oscillator is of the form

$$\psi = Ae^{ikx} + Be^{-ikx}$$

So, when  $x$  approaches  $+\infty$  or  $-\infty$  then  $\psi$  approaches  $\infty$ , which is physically not feasible. We can ignore this divergence of approaching  $\infty$  theoretically by removing  $Ae^{ikx}$  or  $Be^{-ikx}$  when required.

However, when we solve for same equation numerically, it's not possible to do so. Therefore, the wave function approaches  $\infty$  infinity while taking bigger values of  $x$ .

(b) Discuss how the instability discussed above maybe controlled by matching the solution at a point in the allowed region.

Ans. To remove the instability, we divide the integration region ( $x_{\min}$  to  $x_{\max}$ ) into two parts by classical turning point (the point at which solution becomes unstable).

Therefore, we perform integration from 0 to  $x_{cp}$  with forward integration.

$\rightarrow$  Perform integration from  $x_{\max}$  to  $x_{cp}$  with backward integration.

Then, we have to do matching at classical turning point to ensure that we get same value at classical turning point with forward and backward integration.

To do matching

$\rightarrow$  match the value of integration at  $x_{cp}$  by using scaling factors

$\rightarrow$  match the derivative at  $x_{cp}$

# Programming

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import math
4 import scipy.integrate as integrate
5 from scipy import optimize, stats
6 from scipy.optimize import fsolve
7 import pandas as pd
8 from scipy.integrate import solve_ivp
9
10 def numerov(x, E_min, E_max):
11     c_i = []; u = []
12     h = x[1] - x[0]
13     Alpha = 2 * (((-x**2)/2) + (E_min + E_max)/2)
14     ddx_12 = (h**2)/12
15     for i in range(0, len(x)):
16         c_i_ = 1 + np.multiply(ddx_12, Alpha[i])
17         c_i.append(c_i_)
18     if (E_max - 0.5) % 2 == 0:
19         u_0 = 1
20         u_1 = (6 - 5 * c_i[0]) / c_i[1]
21     else:
22         u_0 = 0
23         u_1 = h
24     u.append(u_0); u.append(u_1)
25     for i in range(2, len(x)):
26         u_ = (1/c_i[i]) * (((12 - 10 * c_i[i-1]) * u[i-1]) - c_i[i-2] * u[i-2])
27         u.append(u_)
28     return u, x
29 def numerov_(x, E_min, E_max):
30     c_i = []; u = []
31     h = -(x[1] - x[0])
32     Alpha = 2 * (((-x**2)/2) + (E_min + E_max)/2)
33     ddx_12 = (h**2)/12
34
35     for i in range(0, len(x)):
36         c_i_ = 1 + np.multiply(ddx_12, Alpha[i])
37         c_i.append(c_i_)
38     u_0 = 0
39     u_1 = h
40     u.append(u_0); u.append(u_1)
41     for i in range(2, len(x)):
42         u_ = (1/c_i[i]) * (((12 - 10 * c_i[i-1]) * u[i-1]) - c_i[i-2] * u[i-2])
43         u.append(u_)
44     return u, x
45
46 def e_range(u, n_node, E_min, E_max) :
47     I = []
48     E = (E_min + E_max)/2
49     for i in range(len(u)):
50         if (u[i-1] * u[i]) < 0:
51             I.append(i)
52     N_node = len(I)
53     if N_node > int(n_node):
54         E_max = E
55     else:
56         E_min = E
57     return len(I), E_min, E_max
58
59 def E(n_node, E_min, E_max, tol):
60     for i in range(1000):
61         p = numerov(x, E_min, E_max)
62         p1 = numerov_(x1, E_min, E_max)
63         U, U_back_ = matching(p, p1, x, x1)
64         U = U[:-1]
65         U_back = U_back_[:-1]
```

```

66         for i in U_back:
67             U.append(i)
68         u_norm=U/np.sqrt(integrate.simps(np.power(U,2),np.linspace(xi_1,xf_2,len(U)
)))
69         I ,E_min_new,E_max_new = e_range(u_norm,n_node,E_min,E_max)
70         if abs(E_max_new - E_min_new)<tol:
71             break
72         else:
73             E_min = E_min_new
74             E_max = E_max_new
75         return E_min_new,E_max_new,U
76
77 def combine(list1, list2):
78     list1_ = np.delete(list1, len(list1)-1)
79     result_list = []
80     result_list = list(list1_)
81     for item in list2:
82         result_list.append(item)
83     return result_list
84
85
86 def parity(n_node,E_min,E_max,tol):
87     p = E(n_node,E_min,E_max,tol)
88     t = p[2][::-1]
89     # t_1 = p[2][::-1]
90     array = [];array_1=[]
91
92     if n_node%2 != 0:
93         # array_1 = combine((np.multiply(-1,t_1)),p[2])
94         array = combine((np.multiply(-1,t)),p[2])
95         array_1 = combine((np.multiply(-1,t)),p[2])
96     elif n_node%2 == 0:
97         # array_1 = combine(t_1, p[2])
98         array = combine(t, p[2])
99     return array, array
100 def cl_trn_pts(xf_1,N,n_node):
101     x = np.linspace(0,xf_1,N+1)
102     index=0;xf_=0
103     for i in x:
104         index+=1
105         if round(i,2) == round(np.sqrt(2*n_node+1),2):
106             xf_ = i
107
108             break
109     x_for,x_back=x[:index+1],x[index:]
110     return xf_,index,x_for,x_back[::-1]
111
112 def matching(p,p1,x,x1):
113     rescale_fac = p[0][-1]/p1[0][-1]
114     p2=[]
115     for i in p1[0]:
116         p2.append(i*rescale_fac)
117     return p[0], p2
118
119 def E_range(p1,p2,x,x1,E_min,E_max) :
120     E = (E_min+E_max)/2
121     if phi(E) > 0:
122         E_max = E
123     else:
124         E_min = E
125     return E_min,E_max
126
127 def phi(E):
128     array, array1 = parity(n_node,E,E_max,tol)
129     u_norm=array/np.sqrt(integrate.simps(np.power(array,2),np.linspace(-xf_2,xf_2,
len(array))))
130     c_i=[]

```

```

131     h = x[1]-x[0]
132     Alpha = 2*(((x**2)/2)+E)
133     ddx_12 = (h**2)/12
134     for i in range(0,len(x)):
135         c_i_ = 1 + np.multiply(ddx_12,Alpha[i])
136         c_i.append(c_i_)
137     t= int(len(u_norm)/2)
138     p=len(x)
139     G = (1/h)*(u_norm[t+p+1]+u_norm[t+p-1]-((12*c_i[-1])-10)*u_norm[t+p])
140     return abs(G)
141
142 def match_der(p1,p2,x,x1,E_min,E_max,tol):
143     E_min_new,E_max_new,U=E(n_node,E_min,E_max,tol)
144     root = fsolve(phi,E_min_new,xtol=1e-10)
145     return root[0]
146
147
148 '''
149
150 num_eig_val=[];n=[];anal_eig_val=[]
151 for i in range(0,6):
152     n_node=i
153     xf_1_1=10;N=1000;E_max=n_node+0.5;E_min=0;tol=0.4
154     if n_node %2 ==0:
155         u_0 = 1
156         u_prime=0
157     else:
158         u_0 = 0
159         u_prime = 1
160
161     xf_,index,x,x1 = cl_trn_pts(xf_1_,N,n_node)
162     xi_1=0;xf_1=xf_;xi_2=xf_1_;xf_2=xf_
163     p=numerov(x,E_min, E_max )
164     p1=numerov_(x1,E_min, E_max)
165     p1,p2=matching(p,p1,x,x1)
166     num_eig_val_ = match_der(p1,p2,x,x1,E_min,E_max,tol)
167     num_eig_val.append(num_eig_val_)
168     n.append(i)
169     anal_eig_val.append(i+0.5)
170 print("Table for Eigen Values for xmax = 10")
171 data = {
172     "N":n,
173     "Numerical Eigen Value":num_eig_val,
174     "Analytical Eigen Value":anal_eig_val,
175 }
176 print(pd.DataFrame(data))
177
178 '''
179 num_eig_val=[];n=[];anal_eig_val=[]
180 for i in range(0,2):
181     n_node=i
182     xf_1_2=2;N=1000;E_max=n_node+0.5;E_min=0;tol=0.4
183     if n_node %2 ==0:
184         u_0 = 1
185         u_prime=0
186     else:
187         u_0 = 0
188         u_prime = 1
189
190     xf_,index,x,x1 = cl_trn_pts(xf_1_,N,n_node)
191     xi_1=0;xf_1=xf_;xi_2=xf_1_;xf_2=xf_
192     p=numerov(x,E_min, E_max )
193     p1=numerov_(x1,E_min, E_max)
194     p1,p2=matching(p,p1,x,x1)
195     num_eig_val_ = match_der(p1,p2,x,x1,E_min,E_max,tol)
196     num_eig_val.append(num_eig_val_)

```

```
196     n.append(i)
197     anal_eig_val.append(i+0.5)
198 print("Table for Eigen Values for xmax = 2")
199 data = {
200     "N":n,
201     "Numerical Eigen Value":num_eig_val,
202     "Analytical Eigen Value":anal_eig_val,
203 }
204 print(pd.DataFrame(data))
```

## Result and Discussion

```

    N Numerical Eigen Value Analytical Eigen Value
0 0      0.506604          0.5
1 1      1.506816          1.5
2 2      2.526674          2.5
3 3      3.532667          3.5
4 4      4.519993          4.5
5 5      5.537053          5.5
C:\Users\adn19\anaconda3\lib\site-packages\scipy\optimize\minpack.py:175: RuntimeWarning: The iteration is not making good progress, as measured by the
improvement from the last ten iterations.
  warnings.warn(msg, RuntimeWarning)
Table for Eigen Values for xmax = 2
    N Numerical Eigen Value Analytical Eigen Value
0 0      0.496006          0.5
1 1      1.488351          1.5
2 2      2.480696          2.5
3 3      3.473041          3.5
4 4      4.465386          4.5
5 5      5.457731          5.5
6 6      6.450076          6.5
7 7      7.442421          7.5
8 8      8.434766          8.5
9 9      9.427111          9.5
10 10     10.419456         10.5
11 11     11.411801         11.5
12 12     12.404146         12.5
13 13     13.396491         13.5
14 14     14.388836         14.5
15 15     15.381181         15.5
16 16     16.373526         16.5
17 17     17.365871         17.5
18 18     18.358216         18.5
19 19     19.350561         19.5
20 20     20.342906         20.5
21 21     21.335251         21.5
22 22     22.327596         22.5
23 23     23.319941         23.5
24 24     24.312286         24.5
25 25     25.304631         25.5
26 26     26.296976         26.5
27 27     27.289321         27.5
28 28     28.281666         28.5
29 29     29.274011         29.5
30 30     30.266356         30.5
31 31     31.258701         31.5
32 32     32.251046         32.5
33 33     33.243391         33.5
34 34     34.235736         34.5
35 35     35.228081         35.5
36 36     36.220426         36.5
37 37     37.212771         37.5
38 38     38.205116         38.5
39 39     39.197461         39.5
40 40     40.189806         40.5
41 41     41.182151         41.5
42 42     42.174496         42.5
43 43     43.166841         43.5
44 44     44.159186         44.5
45 45     45.151531         45.5
46 46     46.143876         46.5
47 47     47.136221         47.5
48 48     48.128566         48.5
49 49     49.120911         49.5
50 50     50.113256         50.5
51 51     51.105601         51.5
52 52     52.097946         52.5
53 53     53.090291         53.5
54 54     54.082636         54.5
55 55     55.074981         55.5
56 56     56.067326         56.5
57 57     57.059671         57.5
58 58     58.052016         58.5
59 59     59.044361         59.5
60 60     60.036706         60.5
61 61     61.029051         61.5
62 62     62.021396         62.5
63 63     63.013741         63.5
64 64     64.006086         64.5
65 65     64.998431         65.5
66 66     65.990776         66.5
67 67     66.983121         67.5
68 68     67.975466         68.5
69 69     68.967811         69.5
70 70     69.960156         70.5
71 71     70.952501         71.5
72 72     71.944846         72.5
73 73     72.937191         73.5
74 74     73.929536         74.5
75 75     74.921881         75.5
76 76     75.914226         76.5
77 77     76.906571         77.5
78 78     77.898916         78.5
79 79     78.891261         79.5
80 80     79.883606         80.5
81 81     80.875951         81.5
82 82     81.868296         82.5
83 83     82.860641         83.5
84 84     83.852986         84.5
85 85     84.845331         85.5
86 86     85.837676         86.5
87 87     86.830021         87.5
88 88     87.822366         88.5
89 89     88.814711         89.5
90 90     89.807056         90.5
91 91     90.799401         91.5
92 92     91.791746         92.5
93 93     92.784091         93.5
94 94     93.776436         94.5
95 95     94.768781         95.5
96 96     95.761126         96.5
97 97     96.753471         97.5
98 98     97.745816         98.5
99 99     98.738161         99.5
100 100    99.730506        100.5
101 101    100.722851       101.5
102 102    101.715196       102.5
103 103    102.707541       103.5
104 104    103.699886       104.5
105 105    104.692231       105.5
106 106    105.684576       106.5
107 107    106.676921       107.5
108 108    107.669266       108.5
109 109    108.661611       109.5
110 110    109.653956       110.5
111 111    110.646301       111.5
112 112    111.638646       112.5
113 113    112.630991       113.5
114 114    113.623336       114.5
115 115    114.615681       115.5
116 116    115.608026       116.5
117 117    116.600371       117.5
118 118    117.592716       118.5
119 119    118.585061       119.5
120 120    119.577406       120.5
121 121    120.569751       121.5
122 122    121.562096       122.5
123 123    122.554441       123.5
124 124    123.546786       124.5
125 125    124.539131       125.5
126 126    125.531476       126.5
127 127    126.523821       127.5
128 128    127.516166       128.5
129 129    128.508511       129.5
130 130    129.500856       130.5
131 131    130.493201       131.5
132 132    131.485546       132.5
133 133    132.477891       133.5
134 134    133.470236       134.5
135 135    134.462581       135.5
136 136    135.454926       136.5
137 137    136.447271       137.5
138 138    137.439616       138.5
139 139    138.431961       139.5
140 140    139.424306       140.5
141 141    140.416651       141.5
142 142    141.408996       142.5
143 143    142.401341       143.5
144 144    143.393686       144.5
145 145    144.386031       145.5
146 146    145.378376       146.5
147 147    146.370721       147.5
148 148    147.363066       148.5
149 149    148.355411       149.5
150 150    149.347756       150.5
151 151    150.340101       151.5
152 152    151.332446       152.5
153 153    152.324791       153.5
154 154    153.317136       154.5
155 155    154.309481       155.5
156 156    155.301826       156.5
157 157    156.294171       157.5
158 158    157.286516       158.5
159 159    158.278861       159.5
160 160    159.271206       160.5
161 161    160.263551       161.5
162 162    161.255896       162.5
163 163    162.248241       163.5
164 164    163.240586       164.5
165 165    164.232931       165.5
166 166    165.225276       166.5
167 167    166.217621       167.5
168 168    167.209966       168.5
169 169    168.202311       169.5
170 170    169.194656       170.5
171 171    170.187001       171.5
172 172    171.179346       172.5
173 173    172.171691       173.5
174 174    173.164036       174.5
175 175    174.156381       175.5
176 176    175.148726       176.5
177 177    176.141071       177.5
178 178    177.133416       178.5
179 179    178.125761       179.5
180 180    179.118106       180.5
181 181    180.110451       181.5
182 182    181.102796       182.5
183 183    182.095141       183.5
184 184    183.087486       184.5
185 185    184.079831       185.5
186 186    185.072176       186.5
187 187    186.064521       187.5
188 188    187.056866       188.5
189 189    188.049211       189.5
190 190    189.041556       190.5
191 191    190.033901       191.5
192 192    191.026246       192.5
193 193    192.018591       193.5
194 194    193.010936       194.5
195 195    194.003281       195.5
196 196    195.095626       196.5
197 197    196.087971       197.5
198 198    197.080316       198.5
199 199    198.072661       199.5
200 200    199.065006       200.5
201 201    200.057351       201.5
202 202    201.049696       202.5
203 203    202.042041       203.5
204 204    203.034386       204.5
205 205    204.026731       205.5
206 206    205.019076       206.5
207 207    206.011421       207.5
208 208    207.003766       208.5
209 209    208.096111       209.5
210 210    209.088456       210.5
211 211    210.080801       211.5
212 212    211.073146       212.5
213 213    212.065491       213.5
214 214    213.057836       214.5
215 215    214.050181       215.5
216 216    215.042526       216.5
217 217    216.034871       217.5
218 218    217.027216       218.5
219 219    218.019561       219.5
220 220    219.011906       220.5
221 221    220.004251       221.5
222 222    221.096596       222.5
223 223    222.088941       223.5
224 224    223.081286       224.5
225 225    224.073631       225.5
226 226    225.065976       226.5
227 227    226.058321       227.5
228 228    227.050666       228.5
229 229    228.043011       229.5
230 230    229.035356       230.5
231 231    230.027701       231.5
232 232    231.020046       232.5
233 233    232.012391       233.5
234 234    233.004736       234.5
235 235    234.097081       235.5
236 236    235.089426       236.5
237 237    236.081771       237.5
238 238    237.074116       238.5
239 239    238.066461       239.5
240 240    239.058806       240.5
241 241    240.051151       241.5
242 242    241.043496       242.5
243 243    242.035841       243.5
244 244    243.028186       244.5
245 245    244.020531       245.5
246 246    245.012876       246.5
247 247    246.005221       247.5
248 248    247.097566       248.5
249 249    248.089911       249.5
250 250    249.082256       250.5
251 251    250.074601       251.5
252 252    251.066946       252.5
253 253    252.059291       253.5
254 254    253.051636       254.5
255 255    254.043981       255.5
256 256    255.036326       256.5
257 257    256.028671       257.5
258 258    257.021016       258.5
259 259    258.013361       259.5
260 260    259.005706       260.5
261 261    260.098051       261.5
262 262    261.090396       262.5
263 263    262.082741       263.5
264 264    263.075086       264.5
265 265    264.067431       265.5
266 266    265.059776       266.5
267 267    266.052121       267.5
268 268    267.044466       268.5
269 269    268.036811       269.5
270 270    269.029156       270.5
271 271    270.021501       271.5
272 272    271.013846       272.5
273 273    272.006191       273.5
274 274    273.098536       274.5
275 275    274.090881       275.5
276 276    275.083226       276.5
277 277    276.075571       277.5
278 278    277.067916       278.5
279 279    278.060261       279.5
280 280    279.052606       280.5
281 281    280.044951       281.5
282 282    281.037296       282.5
283 283    282.029641       283.5
284 284    283.021986       284.5
285 285    284.014331       285.5
286 286    285.006676       286.5
287 287    286.099021       287.5
288 288    287.091366       288.5
289 289    288.083711       289.5
290 290    289.076056       290.5
291 291    290.068401       291.5
292 292    291.060746       292.5
293 293    292.053091       293.5
294 294    293.045436       294.5
295 295    294.037781       295.5
296 296    295.030126       296.5
297 297    296.022471       297.5
298 298    297.014816       298.5
299 299    298.007161       299.5
300 300    299.099506       300.5
301 301    300.091851       301.5
302 302    301.084196       302.5
303 303    302.076541       303.5
304 304    303.068886       304.5
305 305    304.061231       305.5
306 306    305.053576       306.5
307 307    306.045921       307.5
308 308    307.038266       308.5
309 309    308.030611       309.5
310 310    309.022956       310.5
311 311    310.015301       311.5
312 312    311.007646       312.5
313 313    312.099991       313.5
314 314    313.092336       314.5
315 315    314.084681       315.5
316 316    315.077026       316.5
317 317    316.069371       317.5
318 318    317.061716       318.5
319 319    318.054061       319.5
320 320    319.046406       320.5
321 321    320.038751       321.5
322 322    321.031096       322.5
323 323    322.023441       323.5
324 324    323.015786       324.5
325 325    324.008131       325.5
326 326    325.000476       326.5
327 327    326.092821       327.5
328 328    327.085166       328.5
329 329    328.077511       329.5
330 330    329.069856       330.5
331 331    330.062201       331.5
332 332    331.054546       332.5
333 333    332.046891       333.5
334 334    333.039236       334.5
335 335    334.031581       335.5
336 336    335.023926       336.5
337 337    336.016271       337.5
338 338    337.008616       338.5
339 339    338.000961       339.5
340 340    339.093306       340.5
341 341    340.085651       341.5
342 342    341.077996       342.5
343 343    342.070341       343.5
344 344    343.062686       344.5
345 345    344.055031       345.5
346 346    345.047376       346.5
347 347    346.039721       347.5
348 348    347.032066       348.5
349 349    348.024411       349.5
350 350    349.016756       350.5
351 351    350.009101       351.5
352 352    351.001446       352.5
353 353    352.093791       353.5
354 354    353.086136       354.5
355 355    354.078481       355.5
356 356    355.070826       356.5
357 357    356.063171       357.5
358 358    357.055516       358.5
359 359    358.047861       359.5
360 360    359.040206       360.5
361 361    360.032551       361.5
362 362    361.024896       362.5
363 363    362.017241       363.5
364 364    363.009586       364.5
365 365    364.001931       365.5
366 366    365.094276       366.5
367 367    366.086621       367.5
368 368    367.078966       368.5
369 369    368.071311       369.5
370 370    369.063656       370.5
371 371    370.056001       371.5
372 372    371.048346       372.5
373 373    372.040691       373.5
374 374    373.033036       374.5
375 375    374.025381       375.5
376 376    375.017726       376.5
377 377    376.010071       377.5
378 378    377.002416       378.5
379 379    378.094761       379.5
380 380    379.087106       380.5
381 381    380.079451       381.5
382 382    381.071796       382.5
383 383    382.064141       383.5
384 384    383.056486       384.5
385 385    384.048831       385.5
386 386    385.041176       386.5
387 387    386.033521       387.5
388 388    387.025866       388.5
389 389    388.018211       389.5
390 390    389.010556       390.5
391 391    390.002901       391.5
392 392    391.095246       392.5
393 393    392.087591       393.5
394 394    393.079936       394.5
395 395    394.072281       395.5
396 396    395.064626       396.5
397 397    396.056971       397.5
398 398    397.049316       398.5
399 399    398.041661       399.5
400 400    399.034006       400.5
401 401    400.026351       401.5
402 402    401.018696       402.5
403 403    402.011041       403.5
404 404    403.003386       404.5
405 405    404.095731       405.5
406 406    405.088076       406.5
407 407    406.080421       407.5
408 408    407.072766       408.5
409 409    408.065111       409.5
410 410    409.057456       410.5
411 411    410.049801       411.5
412 412    411.042146       412.5
413 413    412.034491       413.5
414 414    413.026836       414.5
415 415    414.019181       415.5
416 416    415.011526       416.5
417 417    416.003871       417.5
418 418    417.096216       418.5
419 419    418.088561       419.5
420 420    419.080906       420.5
421 421    420.073251       421.5
422 422    421.065596       422.5
423 423    422.057941       423.5
424 424    423.050286       424.5
425 425    424.042631       425.5
426 426    425.034976       426.5
427 427    426.027321       427.5
428 428    427.019666       428.5
429 429    428.012011       429.5
430 430    429.004356       430.5
431 431    430.096701       431.5
432 432    431.089046       432.5
433 433    432.081391       433.5
434 434    433.073736       434.5
435 435    434.066081       435.5
436 436    435.058426       436.5
437 437    436.050771       437.5
438 438    437.043116       438.5
439 439    438.035461       439.5
440 440    439.027806       440.5
441 441    440.020151       441.5
442 442    441.012496       442.5
443 443    442.004841       443.5
444 444    443.097186       444.5
445 445    444.089531       445.5
446 446    445.081876       446.5
447 447    446.074221       447.5
448 448    447.066566       448.5
449 449    448.058911       449.5
450 450    449.051256       450.5
451 451    450.043601       451.5
452 452    451.035946       452.5
453 453    452.028291       453.5
454 454    453.020636       454.5
455 455    454.012981       455.5
456 456    455.005326       456.5
457 457    456.097671       457.5
458 458    457.089916       458.5
459 459    458.082261       459.5
460 460    459.074606       460.5
461 461    460.066951       461.5
462 462    461.059296       462.5
463 463    462.051641       463.5
464 464    463.043
```