



University of Colorado  
Boulder

---

## HW7

---

Jesse Hettleman  
APPM 4600

COLLEGE OF ENGINEERING AND APPLIED SCIENCE

DEPARTMENT OF APPLIED MATH

October 18, 2024

# 1 Problem 1

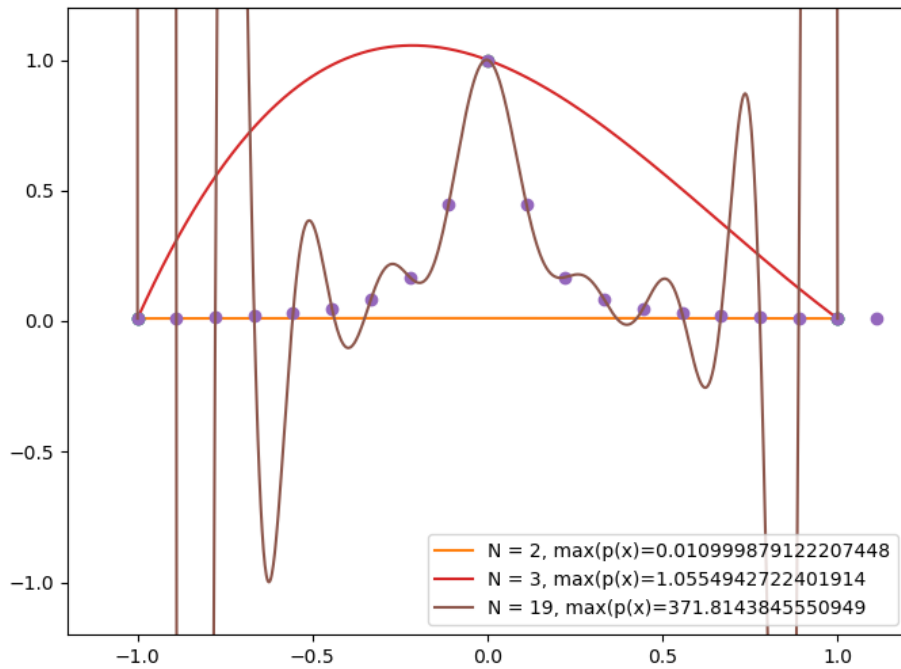
## 1.a

In order to find the coefficients  $c$  of the monomial interpolation, I solved the system  $V\mathbf{c} = \mathbf{y}$ . To do this, I first created a function that creates the Vandermonde matrix  $V$  by filling the  $i^{th}$  row with  $x_i$  to the power of the  $j^{th}$  column. Then, I computed the inverse of this Vandermonde matrix, and multiplied it by  $\mathbf{y}$  to find  $\mathbf{c}$  in the equation  $\mathbf{c} = V^{-1}\mathbf{y}$ . The resulting vector  $\mathbf{c}$  contains the coefficients for the monomial interpolation. See 4 for code conducting these calculations.

## 1.b

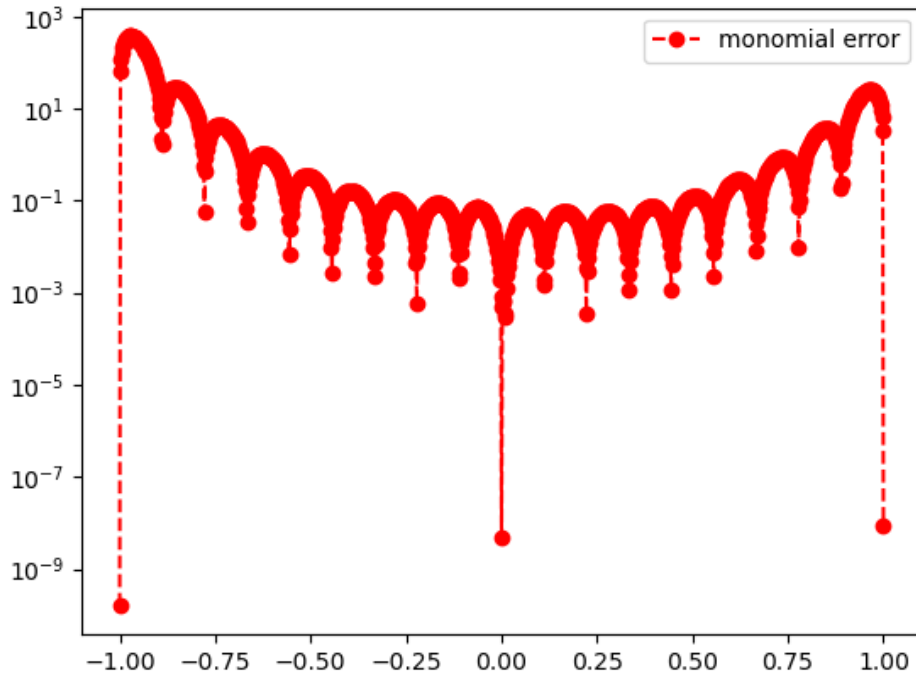
The following plot displays the monomial interpolation of the function  $f(x) = \frac{1}{1+(10x)^2}$  for the interpolation points  $x_i = -1 + (i-1)\frac{2}{N-1}$ . Interpolations are plotted for  $N = 2, 3, 19$ . Once  $N = 19$ , the maximum value of  $p(x) > 100$ . I tested other values of  $N$  to discover this fact, but I have removed these values from the plot because it was too cluttered otherwise.

**Figure 1: Monomial Interpolations for  $N = 2, 3, 19$**



Evidently, as  $N$  increases, the polynomial  $p(x)$  approximates  $f(x)$  more closely for the interpolation points. However, this comes at a tradeoff: the endpoints of polynomials with high  $N$  are very oscillatory due to their high degree. Thus, there is large error near the end points. This is exemplified in the plot below of the error for the polynomial estimation using  $N = 19$ :

**Figure 2:**  $p(x)$  Log Error for  $N = 19$



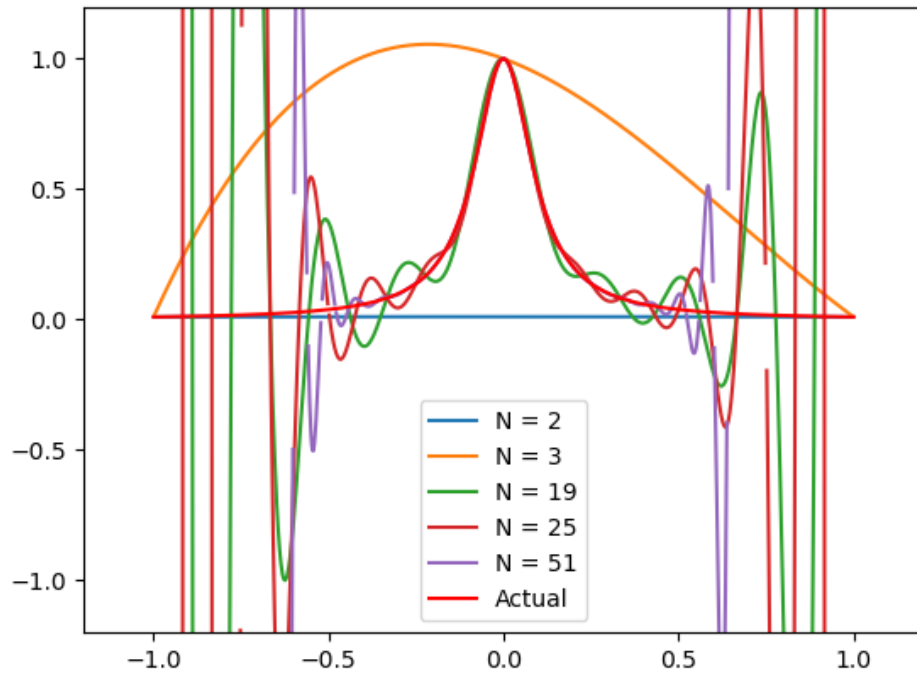
We see that the end points have very large errors, whereas this error decreases for values of  $x$  closer to zero. This is the behavior we expected.

## 2 Problem 2

### 2.a

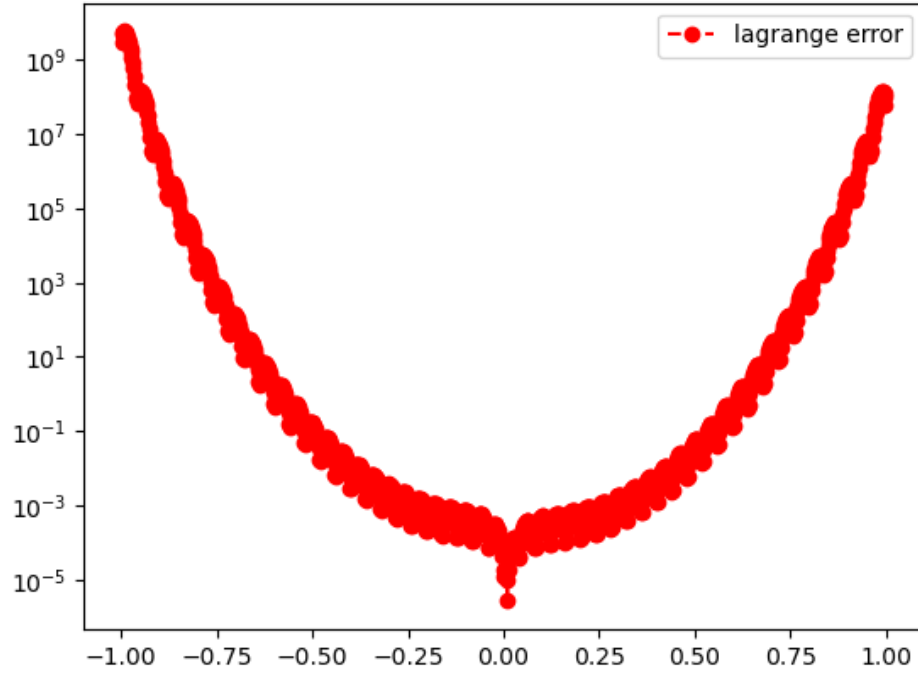
Next we will plot the interpolations of the same function  $f(x)$  but using the Barycentric Lagrange method this time. Below are the plots of various  $p(x)$  approximations for  $N = 2, 3, 19, 25, 51$ .

**Figure 3: Barycentric Lagrange Interpolations for  $N = 2, 3, 19, 25, 51$**



The polynomials this time appear to be very strong for the points closer to zero, but wildly erroneous toward the endpoints. This is especially true for higher degree polynomials. Take a look at the log error plot for  $N = 51$ :

**Figure 4:**  $p(x)$  Log Error for  $N = 51$



The endpoints here have massive errors on the scale of  $10e9$ . However, the values of  $x$  closer to zero in the center perform much better, with absolute errors of around  $10e5$ . Again, this is the behavior we expected from the Barycentric Lagrange method.

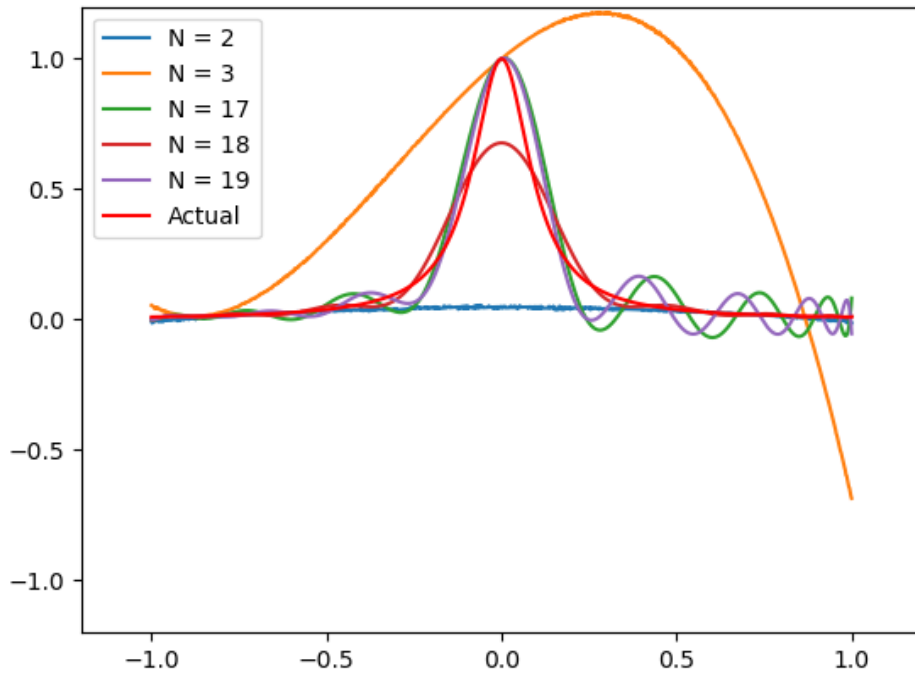
### 3 Problem 3

Next, we use the interpolation nodes  $x_j = \cos\frac{(2j-1)\pi}{2N}$ . These are known as the Chebyshev points.

The first method of monomial interpolation fails for these points due to the fact that the Vandermonde matrix is non-invertible. Therefore, we conclude that the monomial interpolation method is incompatible with the Chebyshev interpolation nodes in this instance.

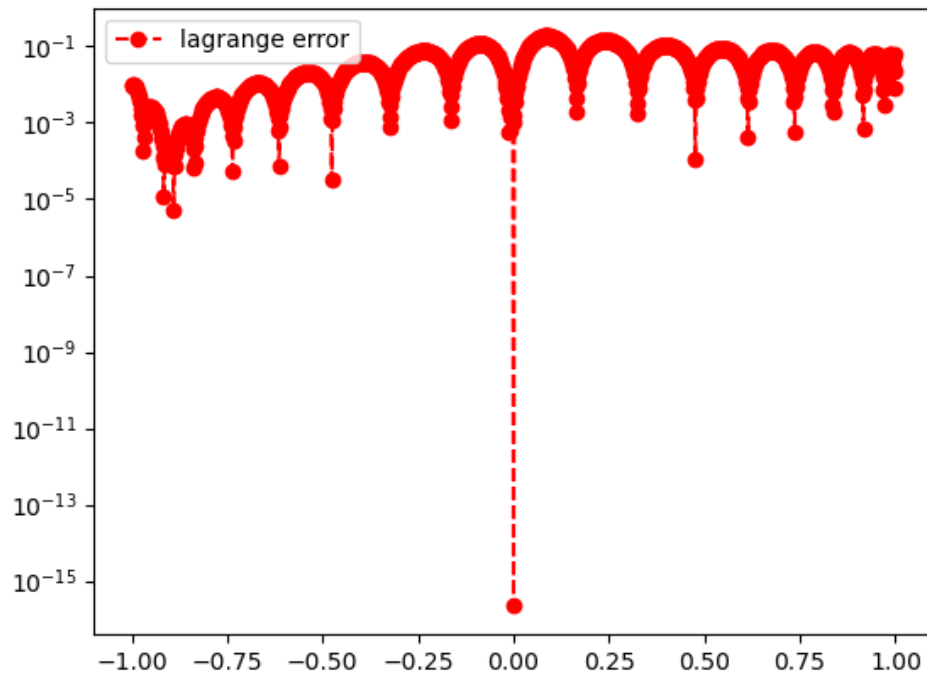
Next, we find that the Barycentric Lagrange method for interpolating the Chebyshev points works. Below is a plot of various  $p(x)$  interpolations for  $N = 2, 3, 17, 18, 19$ :

**Figure 5: Barycentric Lagrange Interpolations for  $N = 2, 3, 17, 18, 19$**



Evidently, the behavior near the endpoints is much better now! Although high degree polynomials still oscillate near the endpoints, these are much more controlled, and result in lower errors. Take a look at the log error plot for  $N = 19$ , for example:

**Figure 6:**  $p(x)$  Log Error for  $N = 19$



A noticeable difference here is that the error is more even throughout the interval, as opposed to the "smile" shaped error curves we observed before. This is a known effect of the Chebyshev points. Furthermore, the overall error is lower on average. Therefore, the Chebyshev points do not fail, and in fact appear to work better for Barycentric Lagrange interpolation.

## 4 Appendix

### 4.1 Code

```
1 import numpy as np
2 import numpy.linalg as la
3 import matplotlib.pyplot as plt
4 from numpy.linalg import inv
5 from numpy.linalg import norm
6 import math
7
8 def eval_monomial(xeval, coef, N, Neval):
9
10     yeval = coef[0]*np.ones(Neval+1)
11
12     # print('yeval = ', yeval)
13
14     for j in range(1, N+1):
15         for i in range(Neval+1):
```

```

16 #         print('yeval[i] = ', yeval[i])
17 #         print('a[j] = ', a[j])
18 #         print('i = ', i)
19 #         print('xeval[i] = ', xeval[i])
20         yeval[i] = yeval[i] + coef[j]*xeval[i]**j
21
22     return yeval
23
24
25 def Vandermonde(xint,N):
26
27     V = np.zeros((N+1,N+1))
28
29     ''' fill the first column'''
30     for j in range(N+1):
31         V[j][0] = 1.0
32
33     for i in range(1,N+1):
34         for j in range(N+1):
35             V[j][i] = xint[j]**i
36
37     return V
38
39
40 def eval_lagrange_bary(xeval,xint,yint,N,f):
41
42     # lj = np.ones(N+1)
43
44     # for count in range(N+1):
45     #     for jj in range(N+1):
46     #         if (jj != count):
47     #             lj[count] = lj[count]*(xeval - ...
48     #                 xint[jj])/(xint[count]-xint[jj])
49
50     # yeval = 0
51
52     # for jj in range(N+1):
53     #     yeval = yeval + yint[jj]*lj[jj]
54
55     phi = 1
56
57     for xi in xint:
58         if xi != xeval:
59             phi *= (xeval - xi)
60
61     summation = 0
62
63     for j in range(N+1):
64
65         wj_denom = 1
66
67         for i in range(N+1):
68             if xint[j] != xint[i]:
69                 wj_denom *= (xint[j] - xint[i])
70
71         wj = 1/wj_denom
72
73         summation += (wj/(xeval-xint[j]))*f(xint[j])

```



```

73
74     yeval = phi*summation
75
76
77     return(yeval)
78
79
80 def question1():
81
82     # Monomial
83
84     f = lambda x: 1 / (1 + ((10*x)**2))
85
86     N = 10
87     a = -1
88     b = 1
89
90     ''' Create interpolation nodes'''
91     xint = np.array([-1 + (j-1)*(2/(N-1)) for j in range(1,N+2)])
92
93     '''Create interpolation data'''
94     yint = f(xint)
95
96     ''' Create the Vandermonde matrix'''
97     V = Vandermonde(xint,N)
98
99     ''' Invert the Vandermonde matrix'''
100    Vinv = inv(V)
101
102    # -----1.a-----
103
104    c = Vinv@yint
105
106    # -----1.b-----
107
108    plt.figure(figsize=(8,6))
109
110    Ns = [2,3,19]
111
112    for N in Ns:
113        a = -1
114        b = 1
115
116        ''' Create interpolation nodes'''
117        xint = np.array([-1 + (j-1)*(2/(N-1)) for j in range(1,N+2)])
118        yint = f(xint)
119
120        ''' Create the Vandermonde matrix'''
121        V = Vandermonde(xint,N)
122
123        ''' Invert the Vandermonde matrix'''
124        Vinv = inv(V)
125
126        c = Vinv@yint
127
128        xeval = np.linspace(-1,1,1001)
129        # yeval = f(xeval)
130        yeval = eval_monomial(xeval,c,N,1000)

```

```

131         ymax = np.max(yeval)
132
133
134
135         plt.plot(xint,yint,'o')
136         plt.plot(xeval,yeval,label=f'N = {N}, max(p(x))={ymax}')
137
138
139
140     plt.legend()
141     plt.xlim(-1.2,1.2)
142     plt.ylim(-1.2,1.2)
143     plt.savefig("HW7.1.b.png")
144
145     fex = f(xeval)
146     plt.figure()
147     err_m = abs(yeval-fex)
148     plt.semilogy(xeval,err_m,'ro--',label='monomial error')
149     plt.legend()
150     plt.savefig("HW7.1.c.png")
151
152     return
153
154 question1()
155
156 def question2():
157
158     # Lagrange Barycentric
159
160     f = lambda x: 1 / (1 + ((10*x)**2))
161
162     plt.figure()
163
164     Ns = [2,3,19,25,51]
165
166     a = -1
167     b = 1
168
169     for N in Ns:
170
171         ''' Create interpolation nodes'''
172         xint = np.array([-1 + (j-1)*(2/(N-1)) for j in range(1,N+2)])
173
174         '''Create interpolation data'''
175         yint = f(xint)
176
177
178         ''' create points for evaluating the Lagrange interpolating ...
            polynomial'''
179         Neval = 1000
180         xeval = np.linspace(a,b,Neval+1)
181         yeval_l= np.zeros(Neval+1)
182
183         ''' evaluate lagrange poly '''
184         for kk in range(Neval+1):
185             yeval_l[kk] = eval_lagrange_bary(xeval[kk],xint,yint,N,f)
186
187

```

```

188
189     plt.plot(xeval,yeval_l,label=f'N = {N}')
190
191     ''' create vector with exact values'''
192     Neval = 1000
193     xeval = np.linspace(a,b,Neval+1)
194     fex = f(xeval)
195
196     plt.plot(xeval,fex,'r-',label='Actual')
197
198     plt.legend()
199     plt.xlim(-1.2,1.2)
200     plt.ylim(-1.2,1.2)
201     plt.savefig("HW7.2.a.png")
202
203     plt.figure()
204     err_l = abs(yeval_l-fex)
205     plt.semilogy(xeval,err_l,'ro--',label='lagrange error')
206     plt.legend()
207     plt.savefig("HW7.2.b.png")
208
209     return
210
211 question2()
212
213 def question3a():
214
215     # Monomial
216
217     f = lambda x: 1 / (1 + ((10*x)**2))
218
219
220
221     plt.figure(figsize=(8,6))
222
223     Ns = [2,3,19]
224
225     a = -1
226     b = 1
227
228     for N in Ns:
229
230         ''' Create interpolation nodes'''
231         xint = np.array([math.cos(((2*j -1)*math.pi)/(2*N)) for j in ...
232                         range(1,N+2)])
233         yint = f(xint)
234
235         ''' Create the Vandermonde matrix'''
236         V = Vandermonde(xint,N)
237
238         ''' Invert the Vandermonde matrix'''
239         Vinv = inv(V)
240
241         c = Vinv@yint
242
243         xeval = np.linspace(-1,1,1001)
244         yeval = f(xeval)
245         yeval = eval_monomial(xeval,c,N,1000)

```

```

245     fex = f(xeval)
246     ymax = np.max(yeval)
247
248
249
250     plt.plot(xint,yint, 'o')
251     plt.plot(xeval,yeval,label=f'N = {N}, max(p(x))={ymax}')
252
253
254
255     plt.legend()
256     plt.xlim(-1.2,1.2)
257     plt.ylim(-1.2,1.2)
258     plt.savefig("HW7.3.a.i.png")
259
260     plt.figure()
261     err = abs(yeval-fex)
262     plt.semilogy(xeval,err, 'ro--',label='monomial error')
263     plt.legend()
264     plt.savefig("HW7.3.a.ii.png")
265
266 # 3a fails due to singular matrix being non-invertible
267
268 def question3b():
269
270     # Lagrange Barycentric
271
272     f = lambda x: 1 / (1 + ((10*x)**2))
273
274     plt.figure()
275
276     Ns = [2,3,17,18,19]
277
278     a = -1
279     b = 1
280
281     for N in Ns:
282
283         ''' Create interpolation nodes'''
284         xint = np.array([math.cos(((2*j -1)*math.pi)/(2*N)) for j in ...
285                         range(1,N+2)])
286
287         '''Create interpolation data'''
288         yint = f(xint)
289
290         ''' create points for evaluating the Lagrange interpolating ...
291             polynomial'''
292         Neval = 1000
293         xeval = np.linspace(a,b,Neval+1)
294         yeval_l= np.zeros(Neval+1)
295
296         ''' evaluate lagrange poly '''
297         for kk in range(Neval+1):
298             yeval_l[kk] = eval_lagrange_bary(xeval[kk],xint,yint,N,f)
299
300     plt.plot(xeval,yeval_l,label=f'N = {N}')

```

```

301
302     ''' create vector with exact values'''
303     Neval = 1000
304     xeval = np.linspace(a,b,Neval+1)
305     fex = f(xeval)
306
307     plt.plot(xeval,fex,'r-',label='Actual')
308
309     plt.legend()
310     plt.xlim(-1.2,1.2)
311     plt.ylim(-1.2,1.2)
312     plt.savefig("HW7.3.b.i.png")
313
314     plt.figure()
315     err = abs(yeval_l-fex)
316     plt.semilogy(xeval,err,'ro--',label='lagrange error')
317     plt.legend()
318     plt.savefig("HW7.3.b.ii.png")
319
320 question3b()

```

NOTE: ALL CODE FOR THIS ASSIGNMENT IS ORIGINAL, WRITTEN BY JESSE HETTLERMAN