Expression d'une assertion impliquant la conjecture de Goldbach dans $\mathbb{C}$

L'assertion mathématique ci-dessous exprime le fait que n, un nombre pair supérieur ou égal à 6, possède un décomposant de Goldbach x (i.e. un nombre premier tel que $n - x$ est premier également).

C'est une assertion plus forte que la conjecture de Goldbach (elle implique la conjecture) dans la mesure où le nombre premier x que définit l'assertion est compris entre $\sqrt{n}$ et $n/2$. Les nombres premiers inférieurs à $\sqrt{n}$ dont le complémentaire est premier ne vérifient pas l'assertion (a).

L'assertion (a) exprime le fait que x est un nombre premier compris entre $\sqrt{n}$ et $n/2$ et l'assertion b exprime le fait que $n - x$ est un nombre premier également.

ε doit être "suffisamment petit". Une valeur de 10^{-10} convient pour détecter les décomposants de Goldbach des nombres 6 à 100 par exemple.

$$\forall n \geq 6, \exists x \geq 3 \,/$$

$$\left| \prod_{\substack{2 \leq x \leq \frac{n}{2} \\ 2 \leq p \leq \sqrt{n}}} \left(1 - \exp\left(\frac{2i\pi x}{p}\right)\right) \right| > \varepsilon \qquad \text{(a)}$$

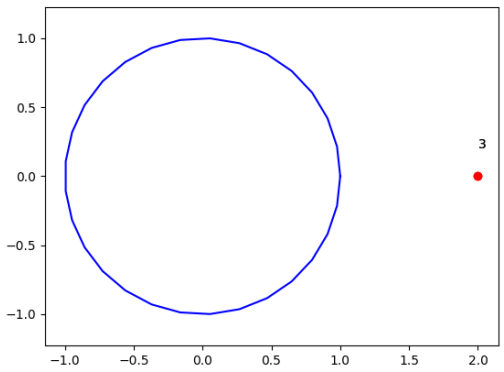
et

$$\left| \prod_{\substack{2 \leq x \leq \frac{n}{2} \\ 2 \leq p \leq \sqrt{n}}} \left(1 - \exp\left(\frac{2i\pi (n-x)}{p}\right)\right) \right| > \varepsilon \qquad \text{(b)}$$

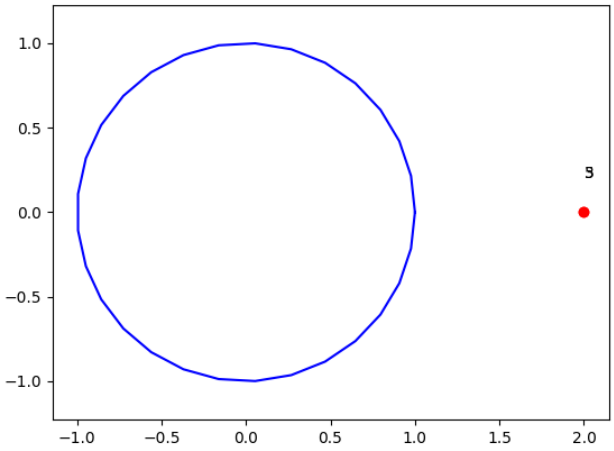
Le programme qui trouve les décompositions de Goldbach des nombres pairs successifs de 6 à 100 est très court. On peut le lire ci-dessous.

On fournit après le programme les images qui montrent où se placent les nombres premiers, dont certains fournissent des décompositions de Goldbach, en dehors du cercle-unité. La position d'un nombre relativement à la position du cercle-unité est modifiée à chaque traversée d'un carré parfait.

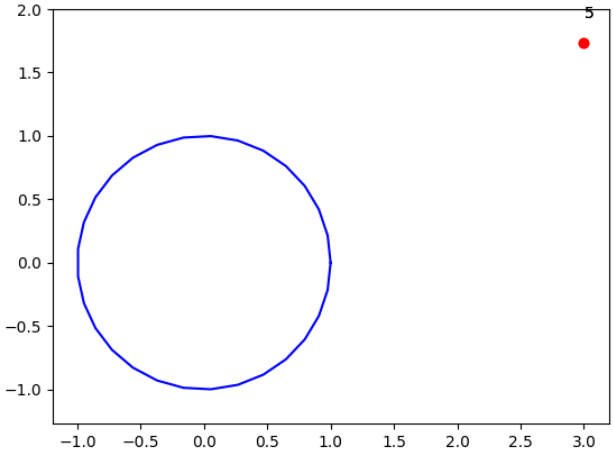
```
from cmath import *
import numpy as np
from numpy import exp, pi, sqrt, linspace, cos, sin
import matplotlib.pyplot as plt

for n in range(6,103,2):
      theta = linspace(0,2*pi,30)
      r = 1
      a = r*cos(theta)
      b = r*sin(theta)
      z = a+1j*b
      plt.plot(a,b, color='blue')
      racine = int(sqrt(n))
      for x in range(2,n//2+1):
            prodbas = 1
            prodhaut = 1
            for p in range(2,racine+1):
                  xx = 2*pi*x/p
                  letout = exp(2*pi*1j*n/p)
                  lun = exp(xx*1j)
                  prodbas = prodbas*(1-lun)
                  produit = letout/lun
                  prodhaut = prodhaut*(1-produit)
            if abs(prodbas) > pow(10,-10) and abs(prodhaut) > pow(10,-10):
                  print(x, ' dg de ',n)
                  plt.scatter(prodbas.real, prodbas.imag, color='red')
                  plt.annotate(x,xy=(prodbas.real, prodbas.imag+0.2))
                  plt.scatter(prodhaut.real, prodhaut.imag, color='red')
                  plt.annotate(n-x,xy=(prodhaut.real, prodhaut.imag+0.2))
      plt.axis('equal')
      plt.show()
```



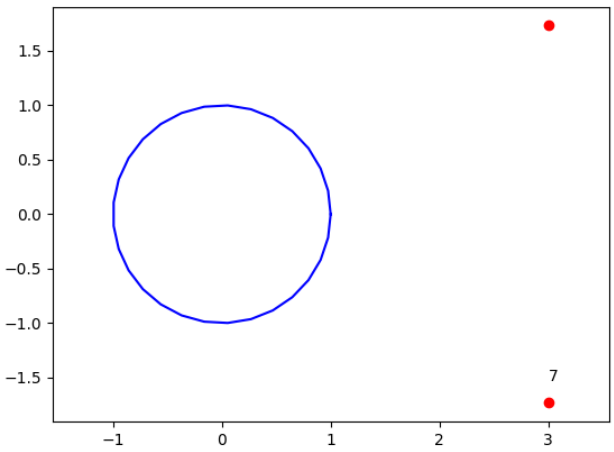
n=6


1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

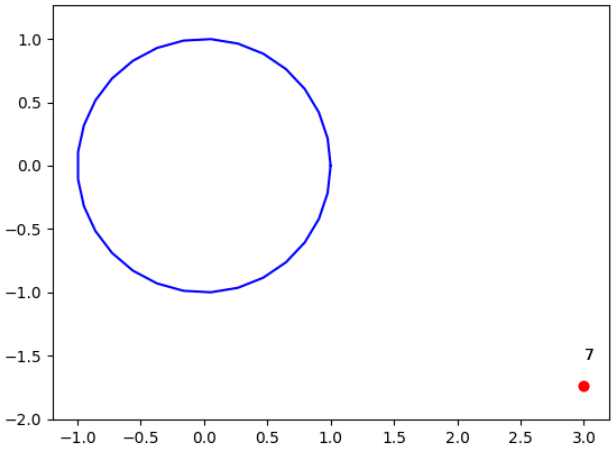
n=8


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0

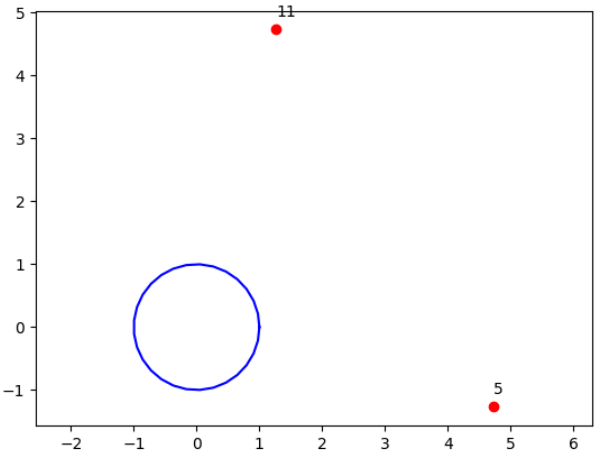
n=10


1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1
0
1
2
3
7

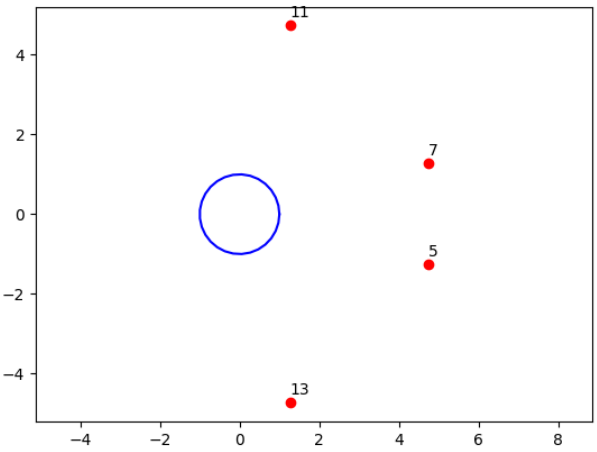
n=12



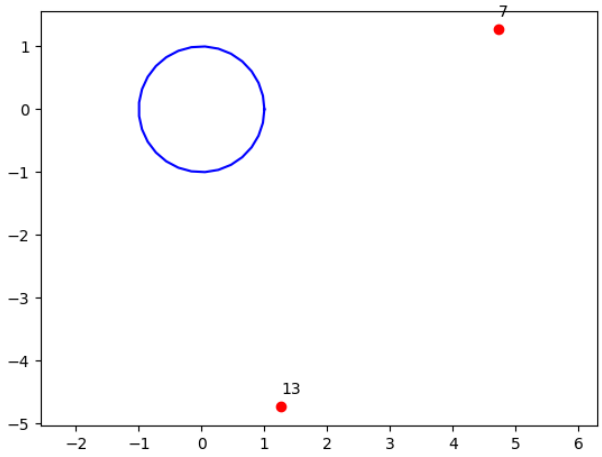
n=14



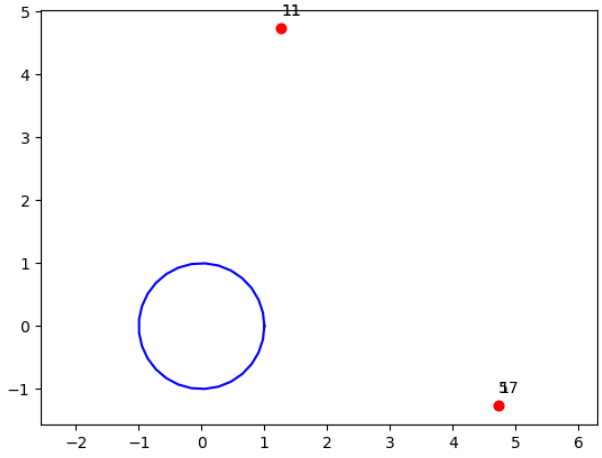
n=16



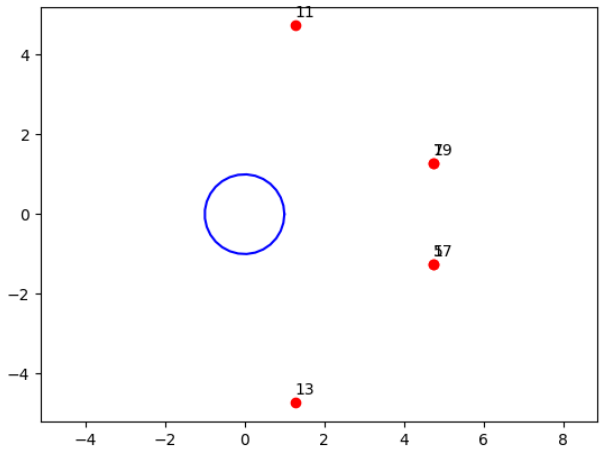
n=18



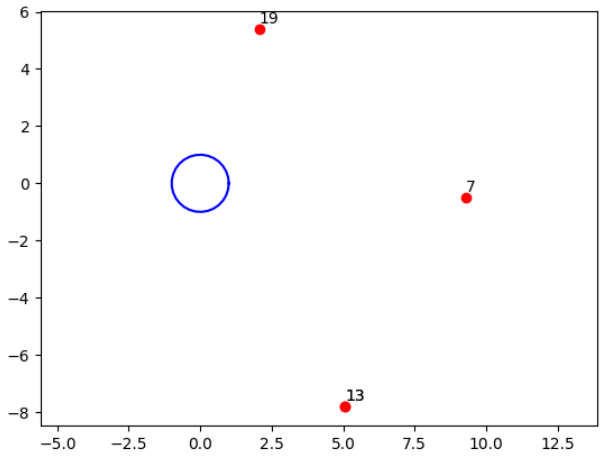
n=20



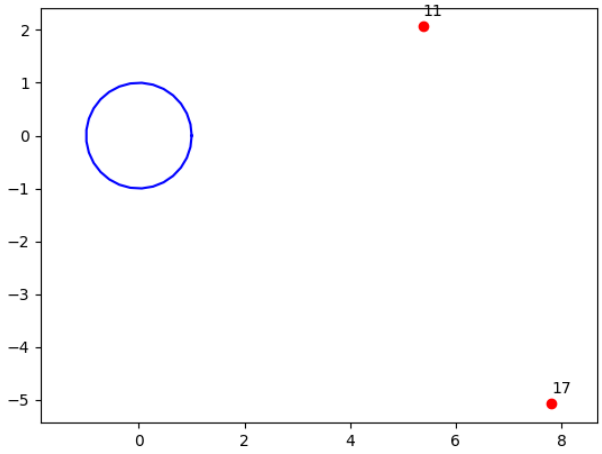
n=22



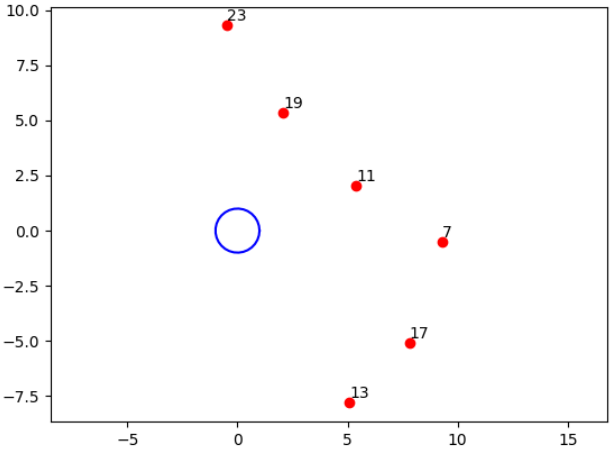
n=24



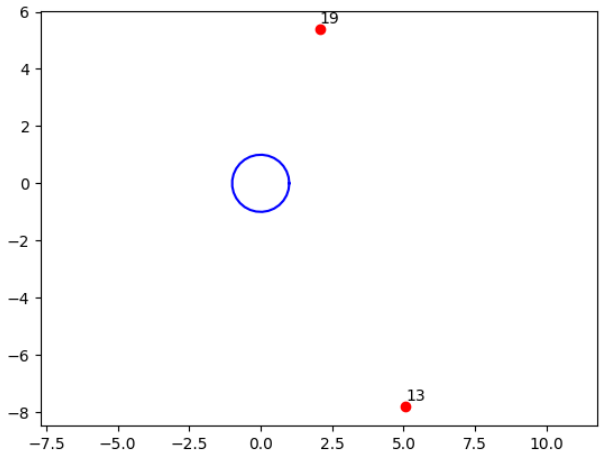
n=26



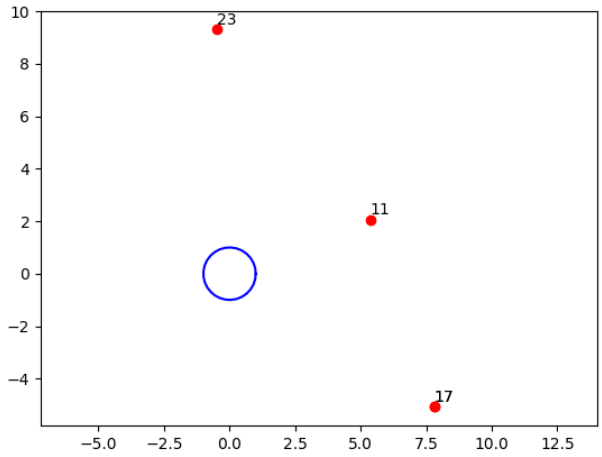
n=28



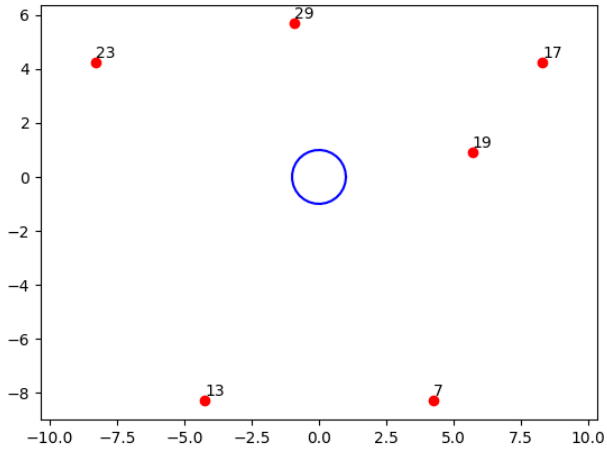
n=30


19
13
6
4
2
0
−2
−4
−6
−8
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

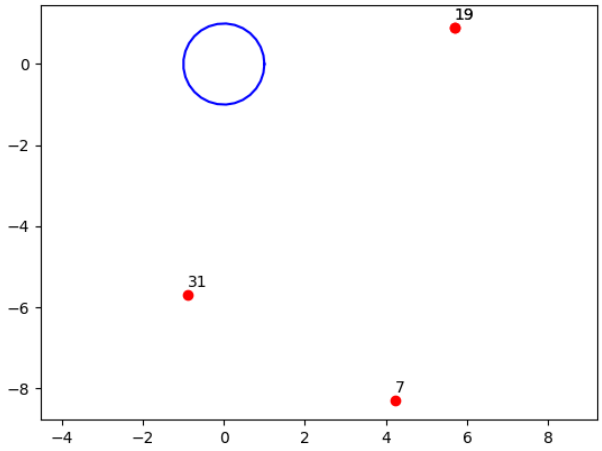
n=32


23
11
17
10
8
6
4
2
0
−2
−4
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5

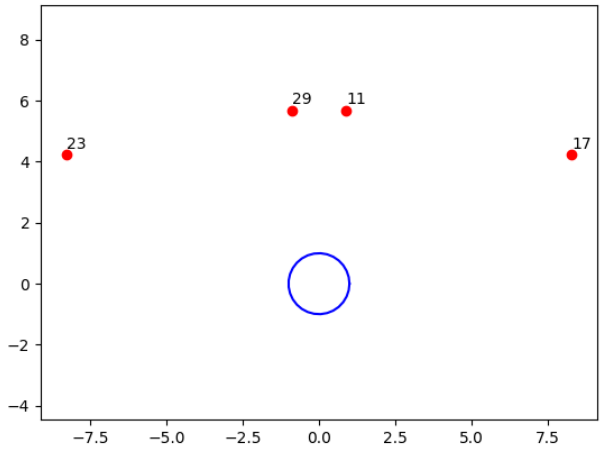
n=34


29
23
17
19
13
7
6
4
2
0
−2
−4
−6
−8
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

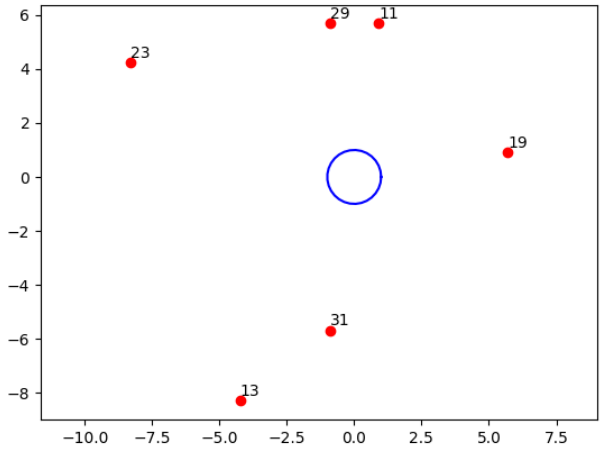
n=36



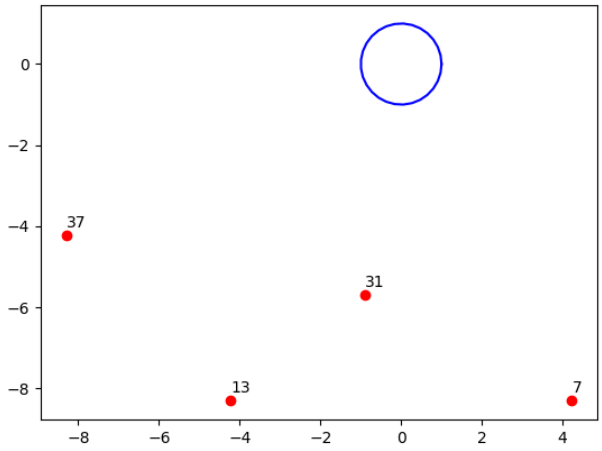
n=38



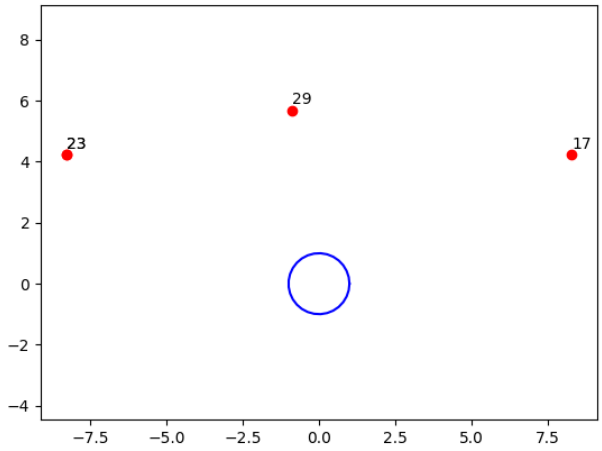
n=40



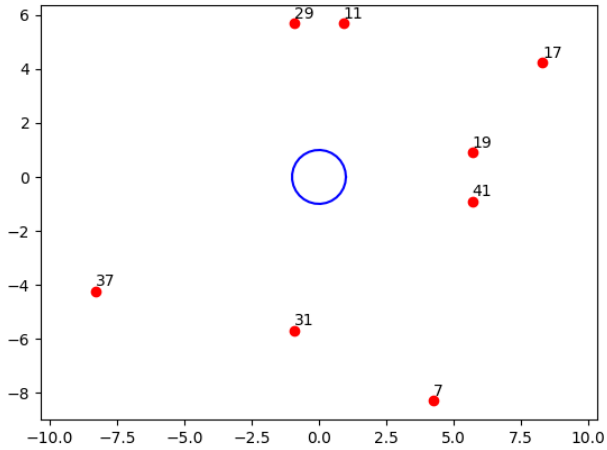
n=42



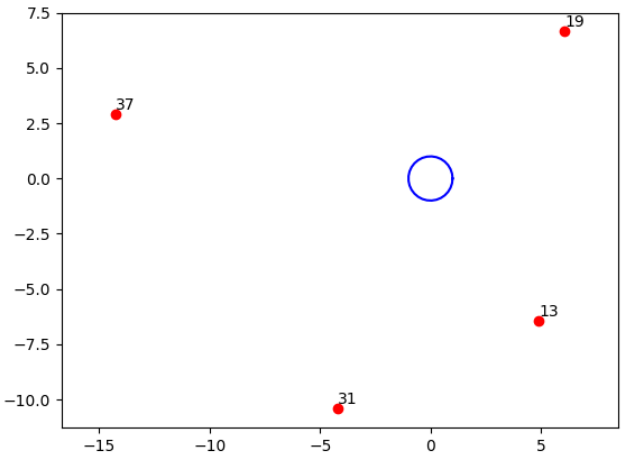
n=44



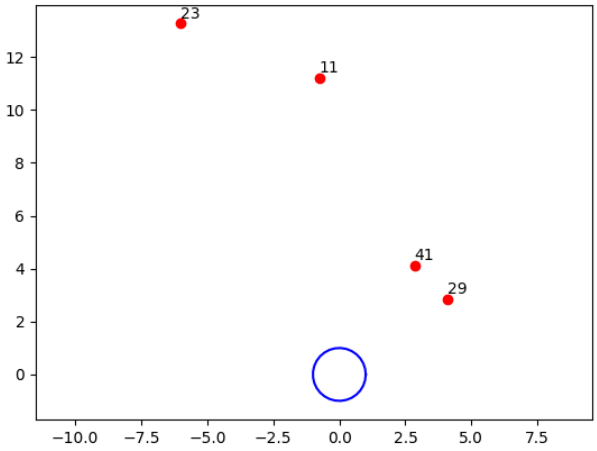
n=46



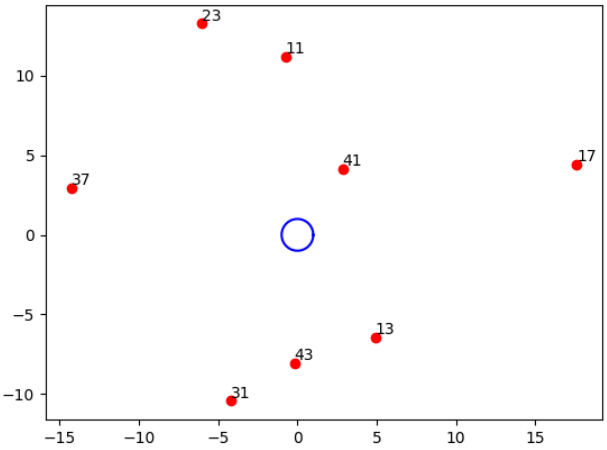
n=48



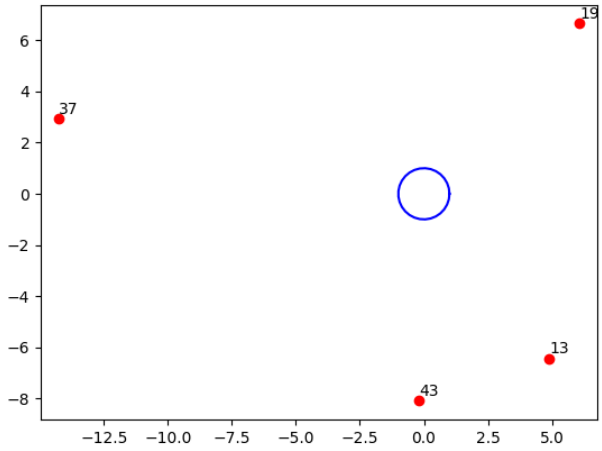
n=50



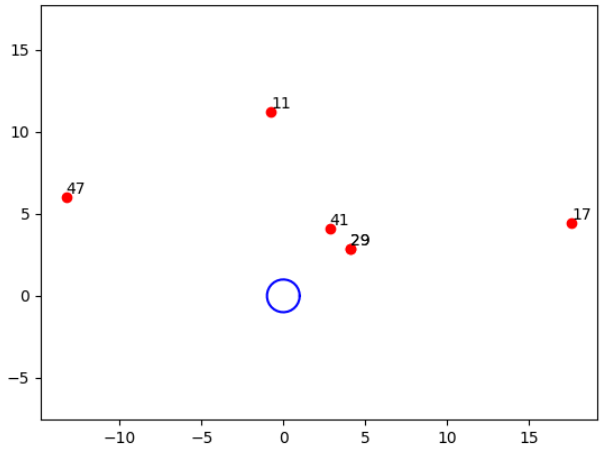
n=52



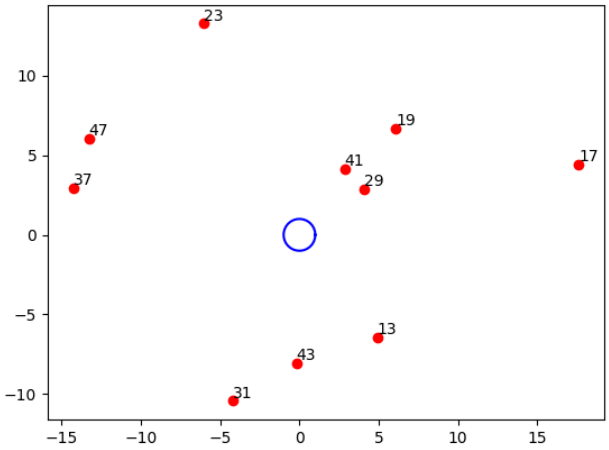
n=54


19
37
13
43
6
4
2
0
−2
−4
−6
−8
−12.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0

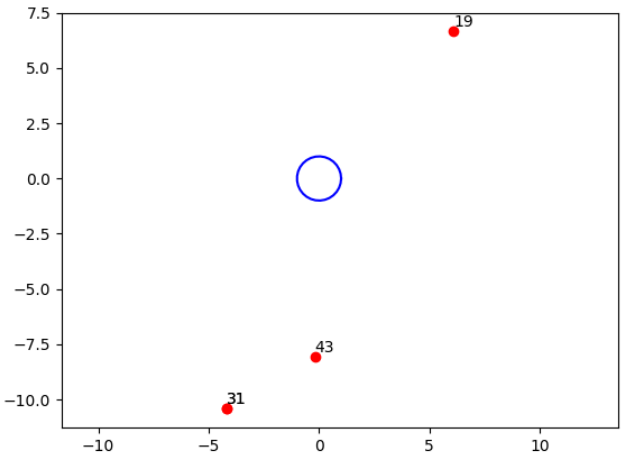
n=56


11
47
41
29
17
15
10
5
0
−5
−10
−5
0
5
10
15

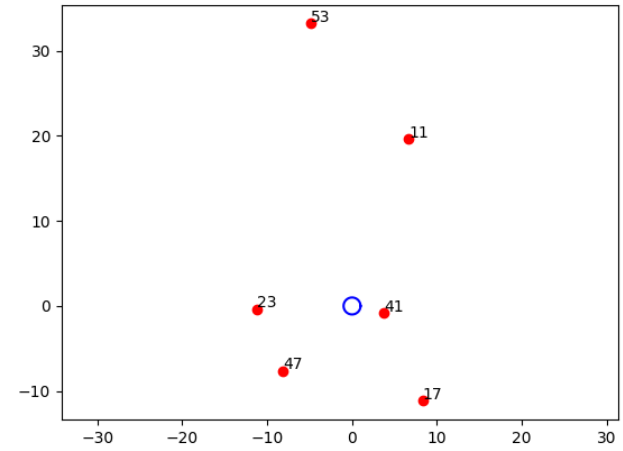
n=58


23
19
47
41
17
37
29
13
43
31
10
5
0
−5
−10
−15
−10
−5
0
5
10
15

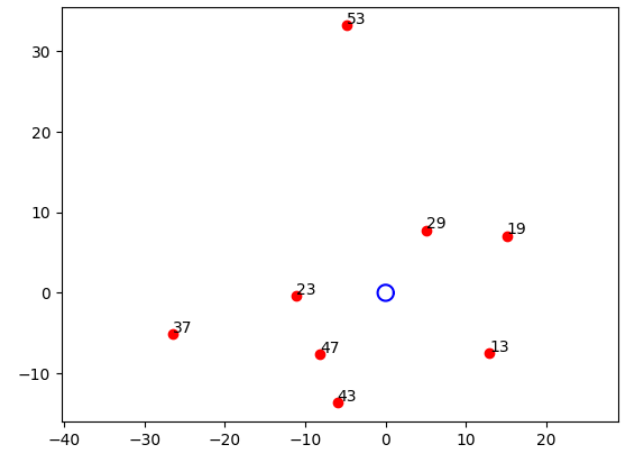
n=60



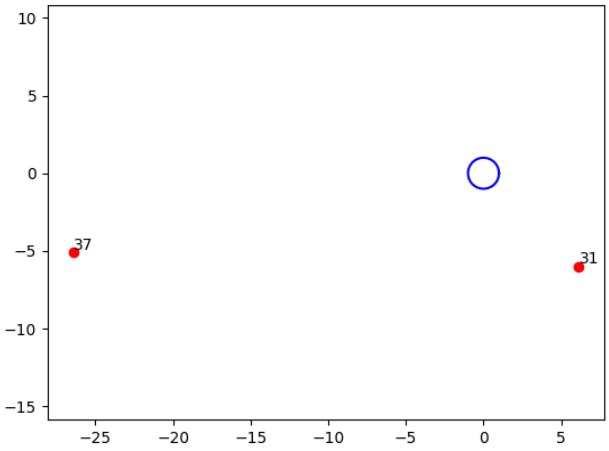
n=62



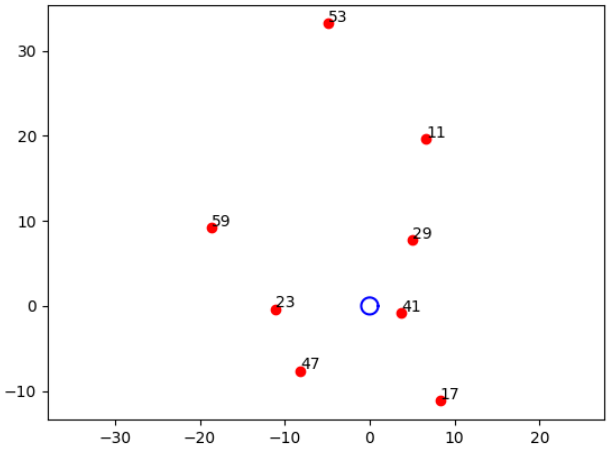
n=64



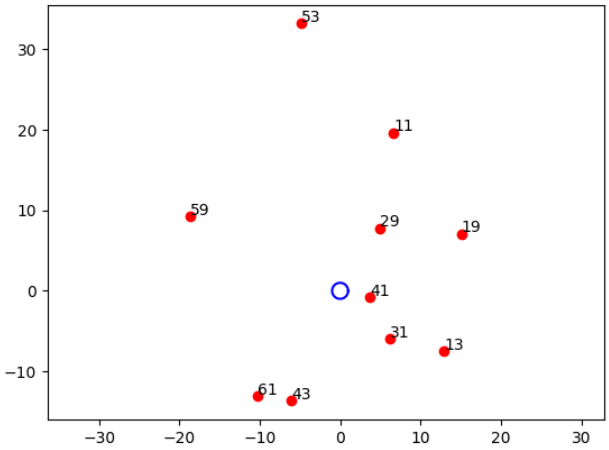
n=66


10
5
0
−5
−10
−15
−25
−20
−15
−10
−5
0
5
37
31

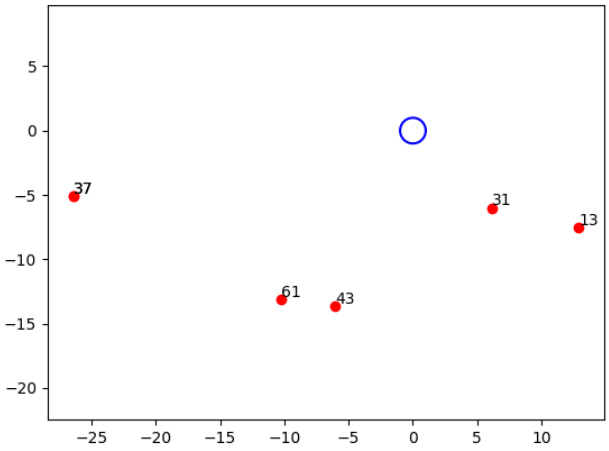
n=68


30
20
10
0
−10
−30
−20
−10
0
10
20
53
11
59
29
23
41
47
17

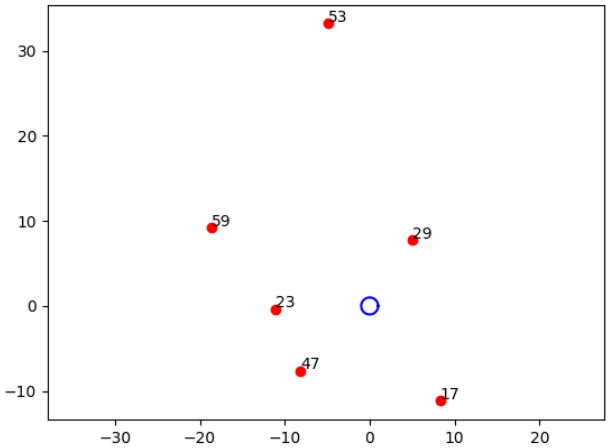
n=70


30
20
10
0
−10
−30
−20
−10
0
10
20
30
53
11
59
29
19
41
31
13
61
43

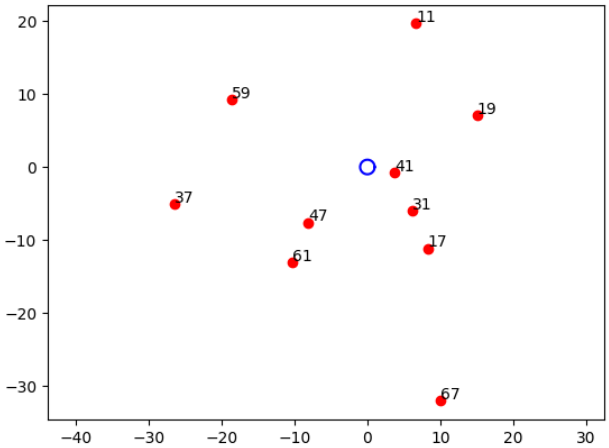
n=72



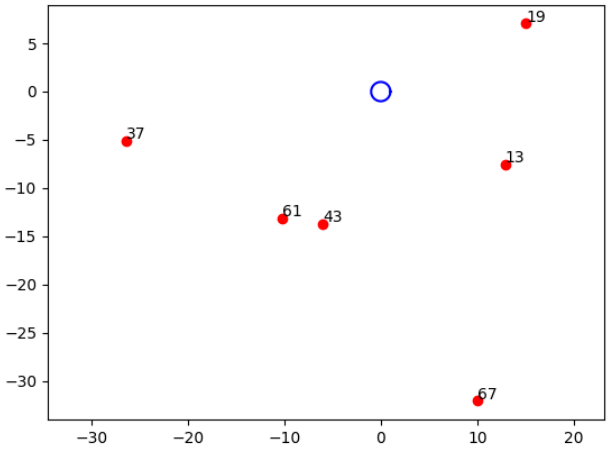
n=74



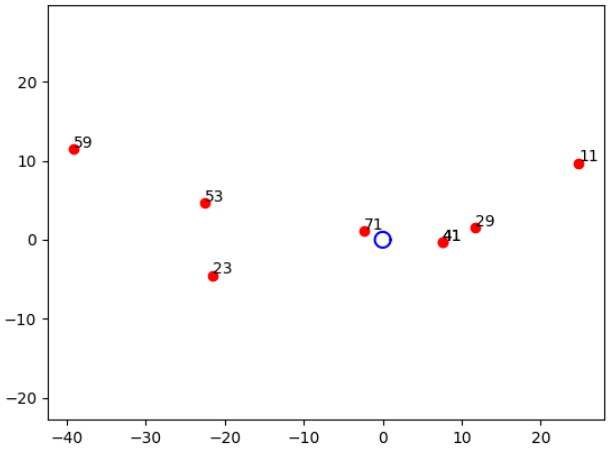
n=76



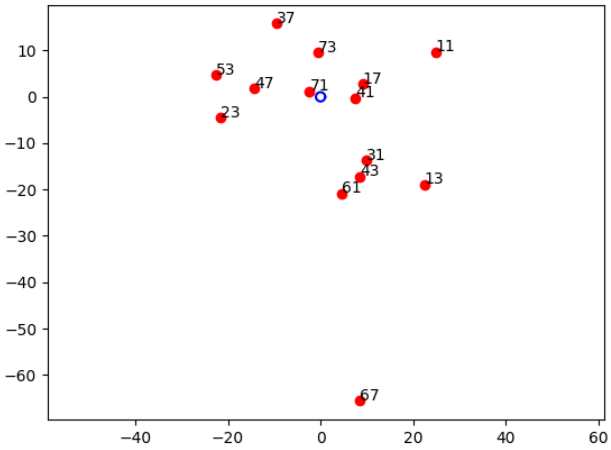
n=78


19
37
13
61
43
67

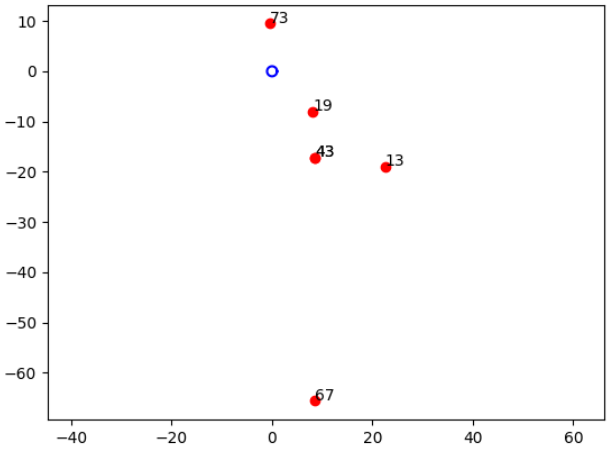
n=80


59
11
53
71
29
41
23

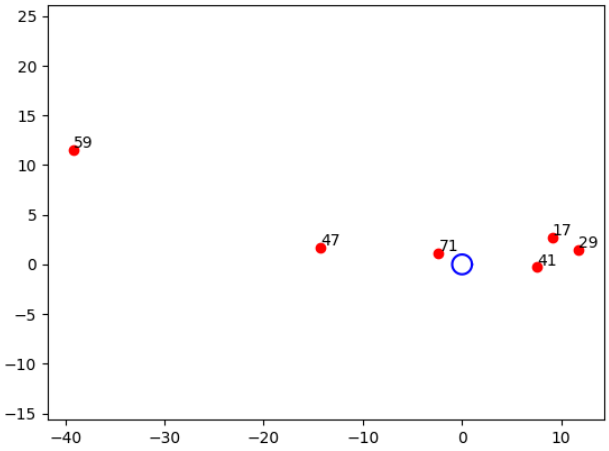
n=82


37
73
11
53
47
71
17
41
23
31
43
13
61
67

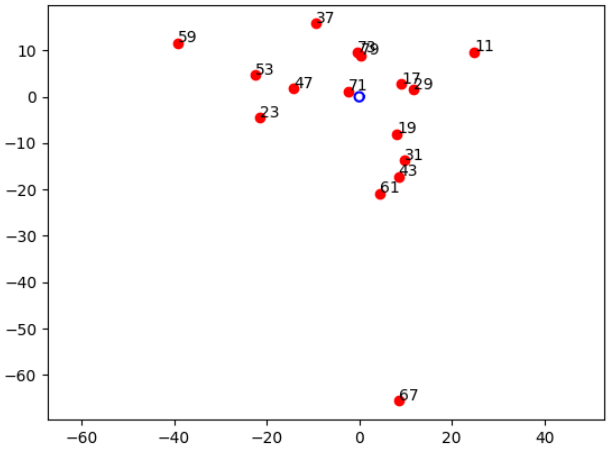
n=84


73
19
43
13
67

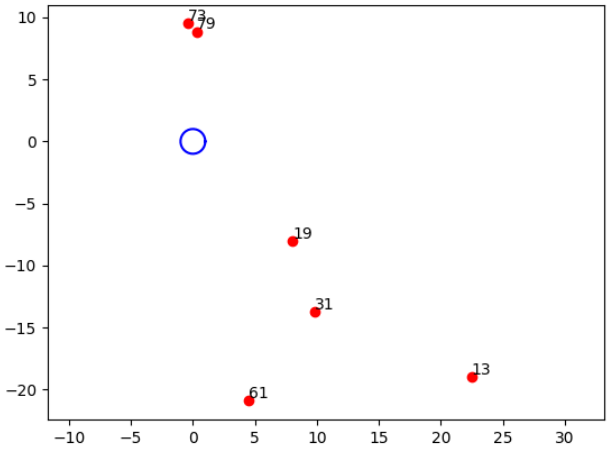
n=86


59
47
71
17
29
41

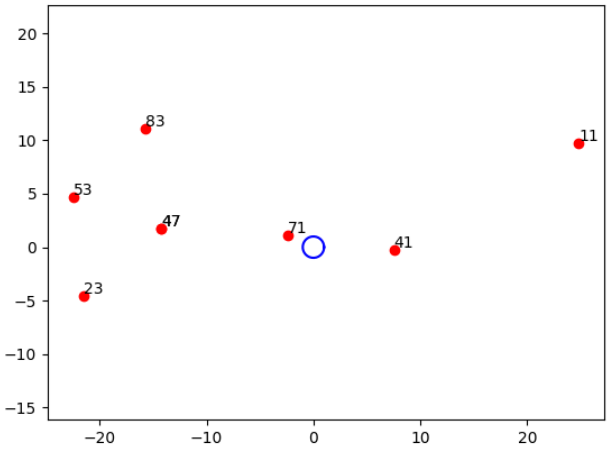
n=88


37
59
11
53
47
71
17
29
23
19
31
43
61
67

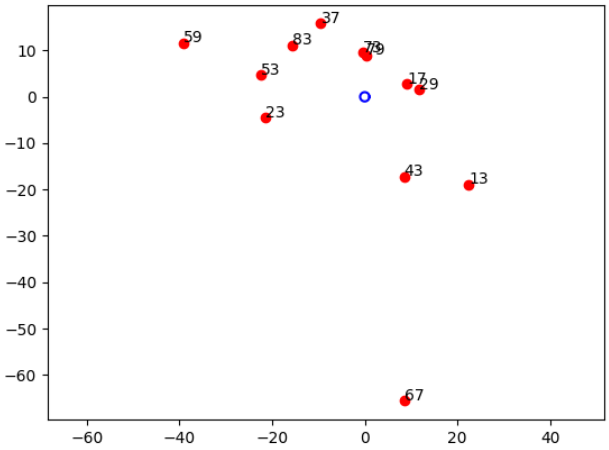
n=90



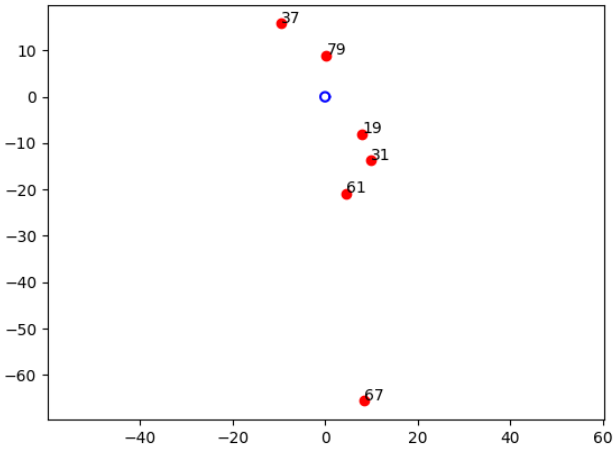
n=92



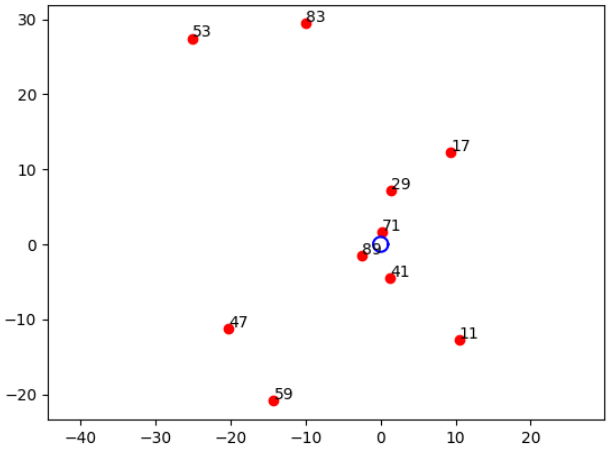
n=94



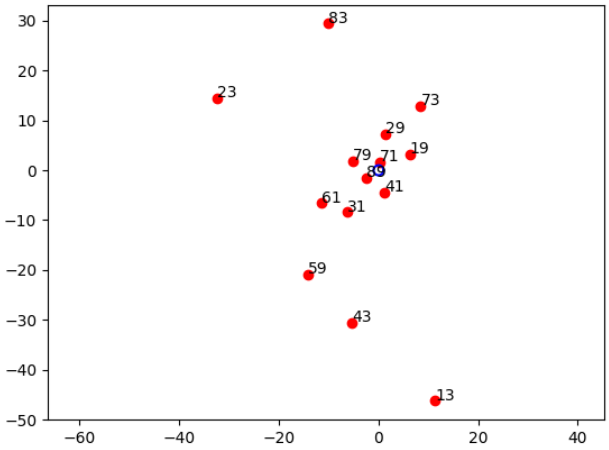
n=96



n=98



n=100



n=102