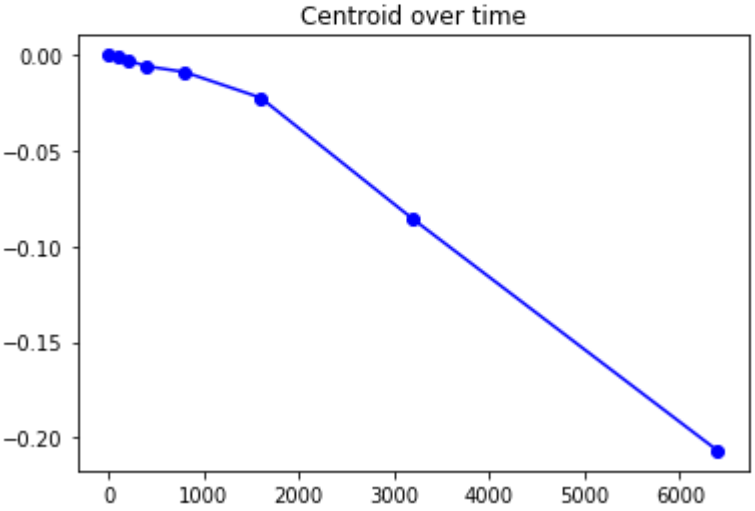
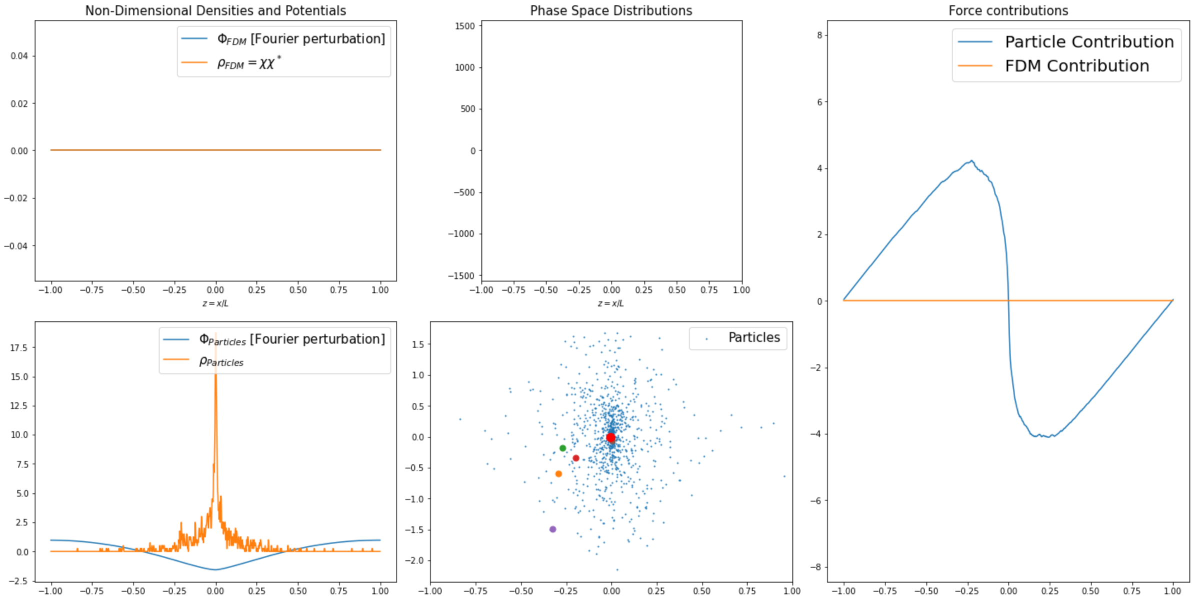
In [ ]:

```
import Analysis

#[ ..., [r,m,Num_bosons,sigma,Num_stars],...]
# Args = [
#     [0.5,1.0,0,1,10000],
#     [0.5,1.0,10000,1,10000],
#     [1,0.5,20000,1,10000],
#     [5,0.1,100000,1,10000],
#     [10,0.05,200000,1,10000],
#     [50,0.01,1000000,1,10000],
#     [0.5,1.0,10000,1,0]
# ]

Args = [
    [1,1,0,0.001,1000],
    [1,1,0,0.0002,5000],
    [1,1,0,0.0001,10000],
    [1,1,0,0.00002,50000],
    [1,1,0,0.00001,100000]
]

for args in Args:
    print("----------New Analysis--------")
    print(
        f"r = {args[0]}",
        f"mu = {args[1]}",
        f"Num_bosons = {args[2]}",
        f"sigma = {args[3]}",
        f"Num_stars = {args[4]}"
    )
    Analysis.analysis(*args)
```

```
/home/boris/Documents/Research/FDM_n_Bodies/1D_Codes/Non-Dim/Analysis
----------New Analysis--------
r = 1 mu = 1 Num_bosons = 0 sigma = 0.001 Num_stars = 1000
```

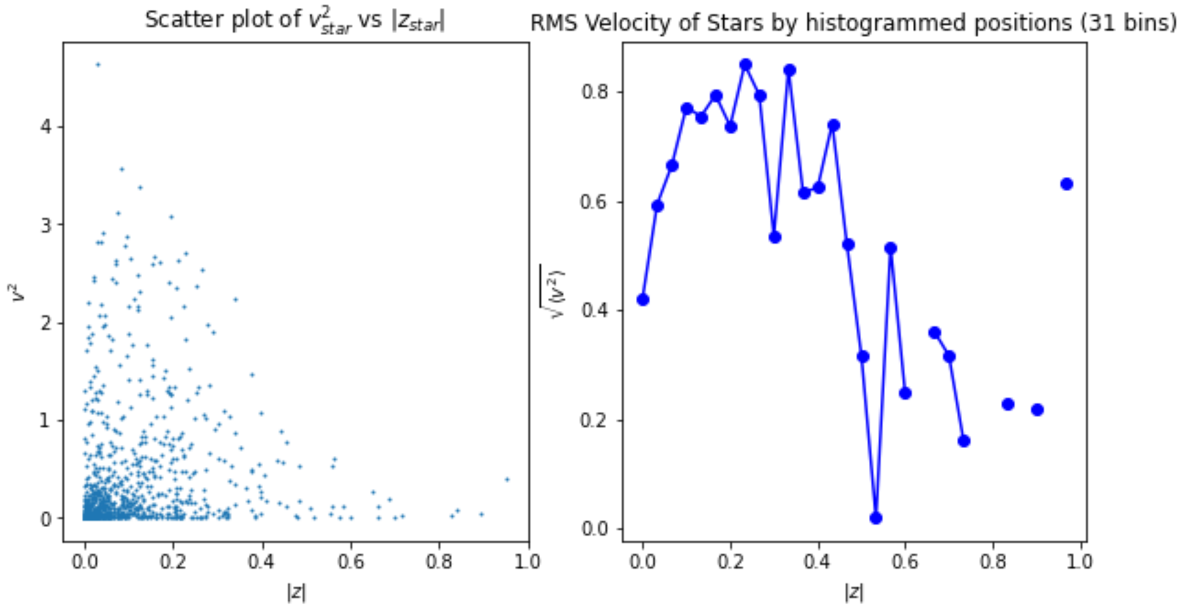


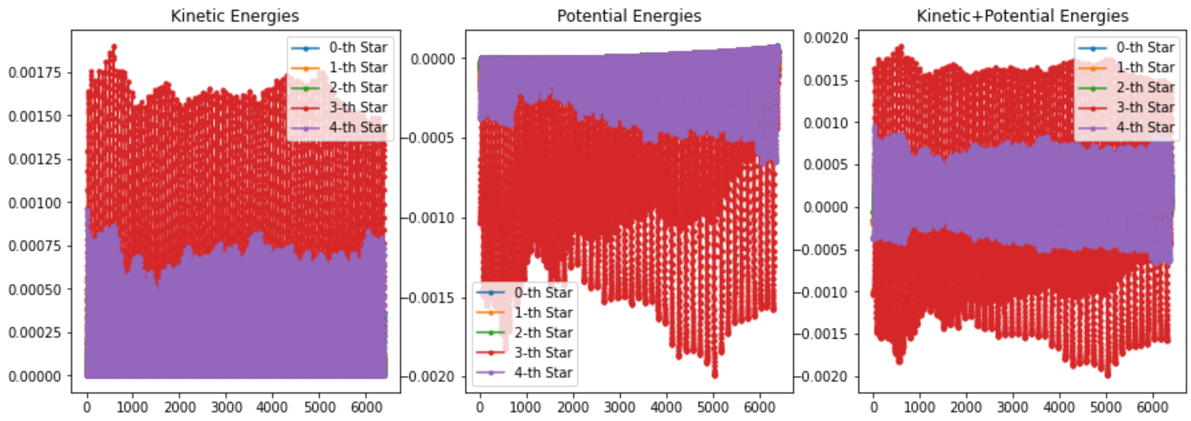
```
/home/boris/Documents/Research/FDM_n_Bodies/OneD/WaveNonDim.py:129: Runtime
Warning: invalid value encountered in true_divide
  F_s = F_s/Norm_const
```

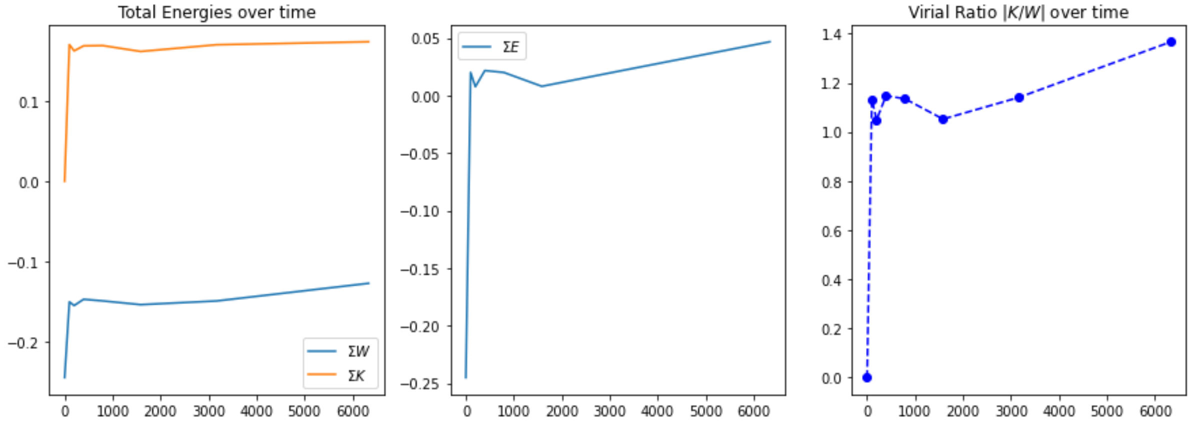


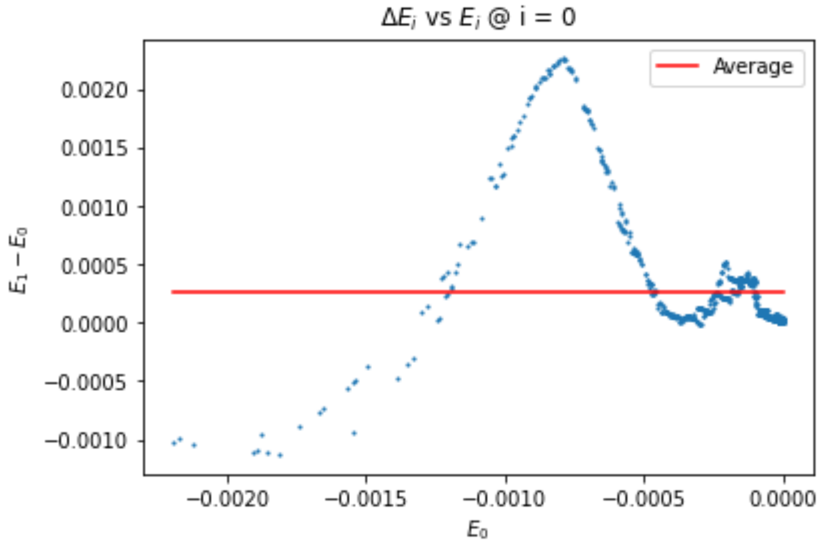
```
v_rms = 0.5923277051957598
z_rms = 0.15871675883738892
K_avg = 0.5*m*v_rms^2 = 0.1754260551712375 (m=1)
=> 2*K_avg = 0.350852110342475
W_avg = 158.7167588373889
------------------
K_tot = 0.17542605517123733
K_avg = 0.00017542605517123733
W_tot = -0.14596981745731766
W_avg = -0.00014596981745731765
```

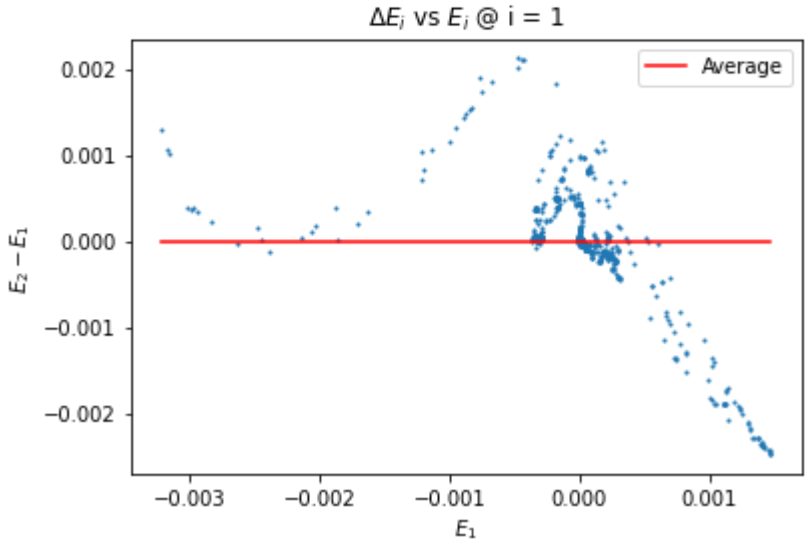
```
/home/boris/Documents/Research/FDM_n_Bodies/1D_Codes/Non-Dim/Analysis/Analy
sis.py:266: RuntimeWarning: invalid value encountered in true_divide
  v_rms_array = bins/bins_counts
```

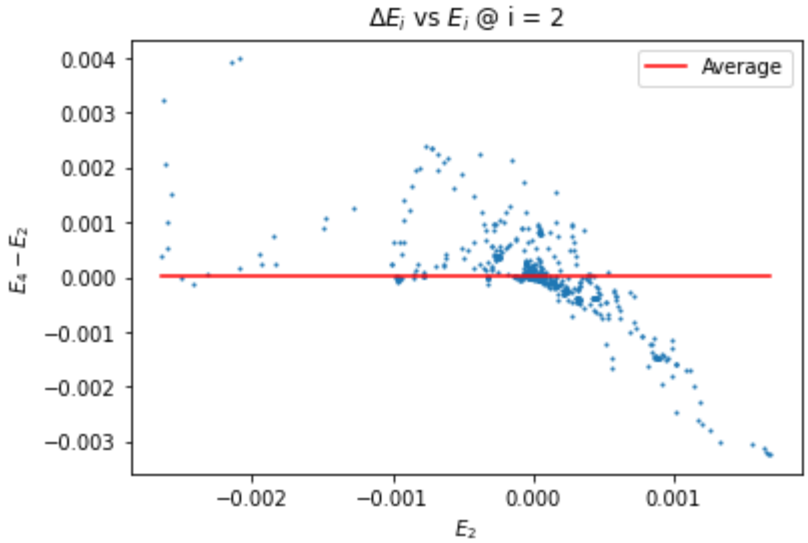


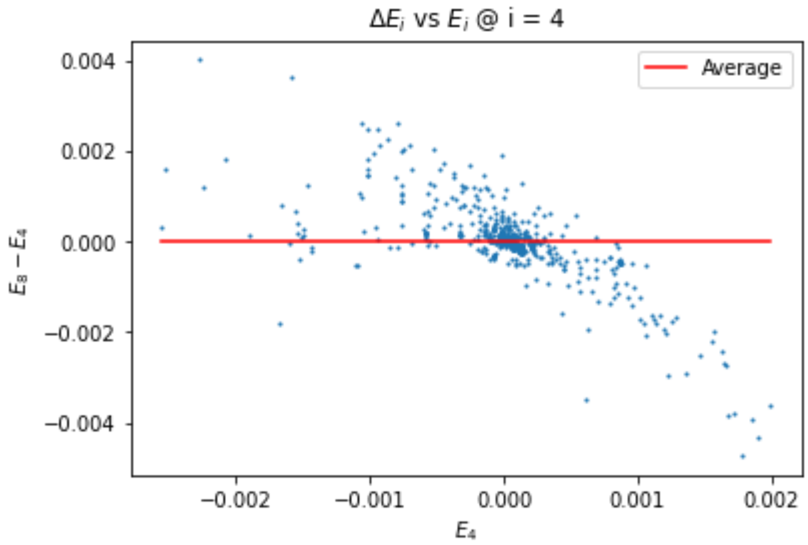

Kinetic Energies
0.00175
0.00150
0.00125
0.00100
0.00075
0.00050
0.00025
0.00000
0
1000
2000
3000
4000
5000
6000
0-th Star
1-th Star
2-th Star
3-th Star
4-th Star
Potential Energies
0.0000
−0.0005
−0.0010
−0.0015
−0.0020
0-th Star
1-th Star
2-th Star
3-th Star
4-th Star
Kinetic+Potential Energies
0.0020
0.0015
0.0010
0.0005
0.0000
−0.0005
−0.0010
−0.0015
−0.0020
0-th Star
1-th Star
2-th Star
3-th Star
4-th Star

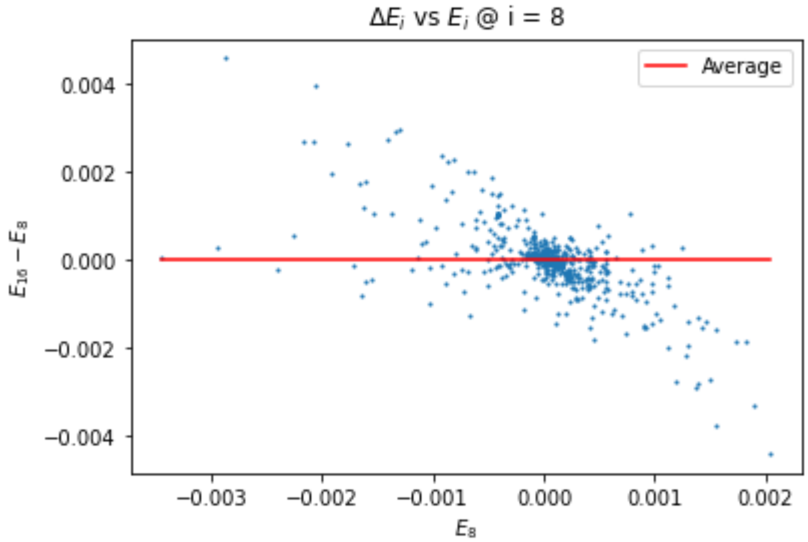

Total Energies over time
0.1
0.0
−0.1
−0.2
ΣW
ΣK
0.05
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
ΣE
Virial Ratio |K/W| over time
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
1000
2000
3000
4000
5000
6000

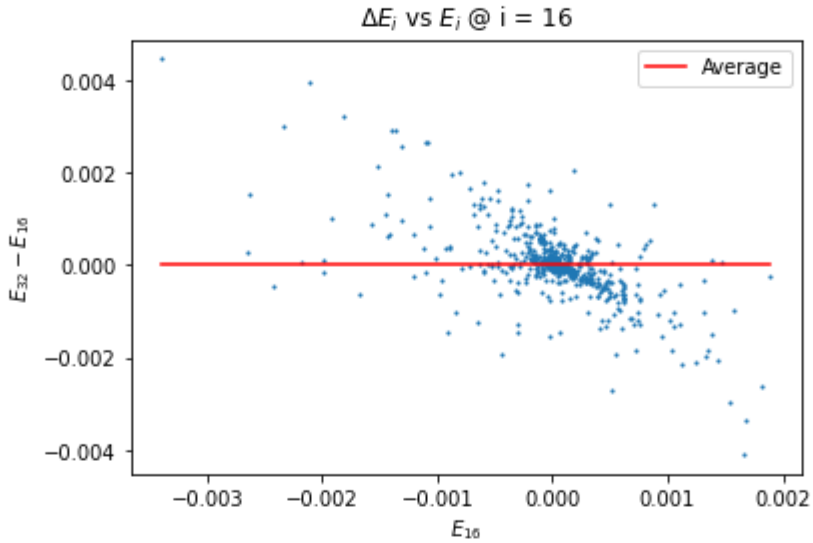

ΔE_i vs E_i @ i = 0
Average
0.0020
0.0015
0.0010
0.0005
0.0000
−0.0005
−0.0010
$E_1 - E_0$
−0.0020
−0.0015
−0.0010
−0.0005
0.0000
E_0

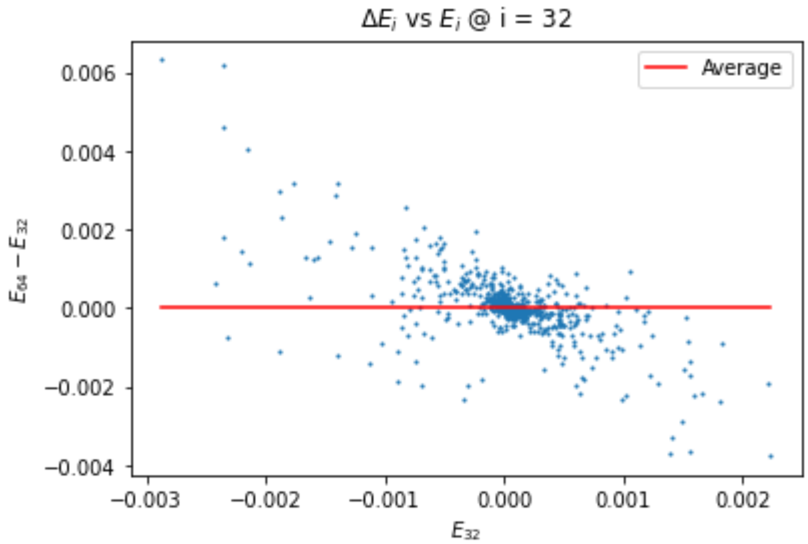

ΔE_i vs E_i @ i = 1
Average
0.002
0.001
0.000
−0.001
−0.002
$E_2 - E_1$
−0.003
−0.002
−0.001
0.000
0.001
E_1

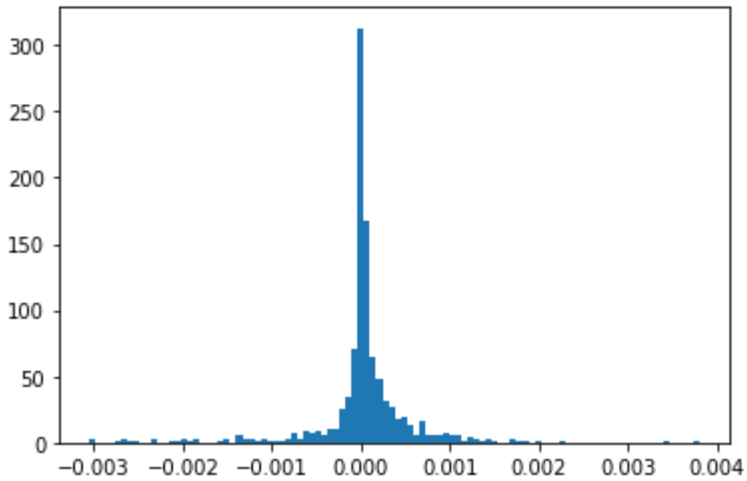

ΔE_i vs E_i @ i = 2
Average
0.004
0.003
0.002
0.001
0.000
−0.001
−0.002
−0.003
$E_4 - E_2$
−0.002
−0.001
0.000
0.001
E_2

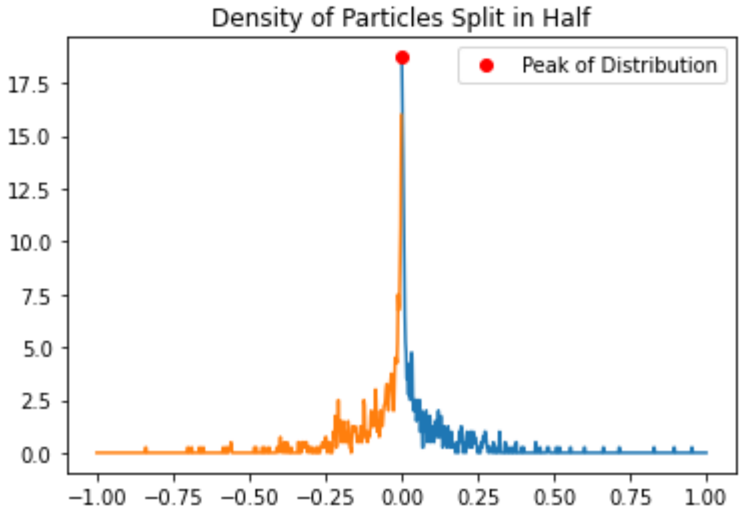

ΔE_i vs E_i @ i = 4
Average
0.004
0.002
0.000
−0.002
−0.004
$E_8 - E_4$
−0.002
−0.001
0.000
0.001
0.002
E_4

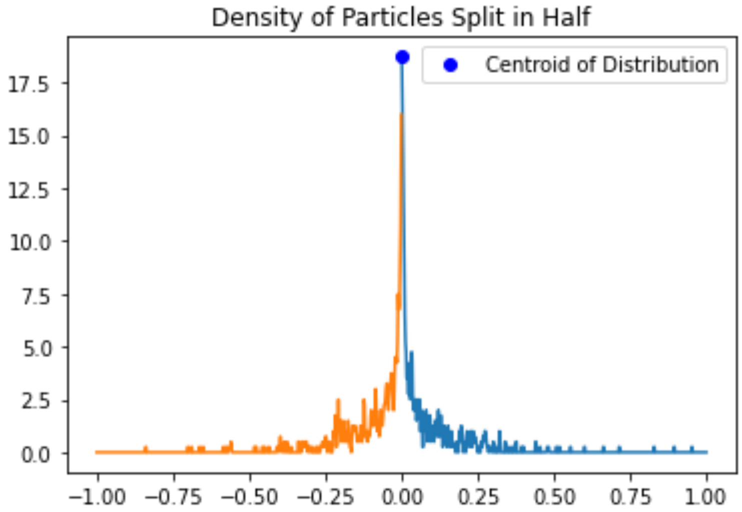

ΔE_i vs E_i @ i = 8


ΔE_i vs E_i @ i = 16

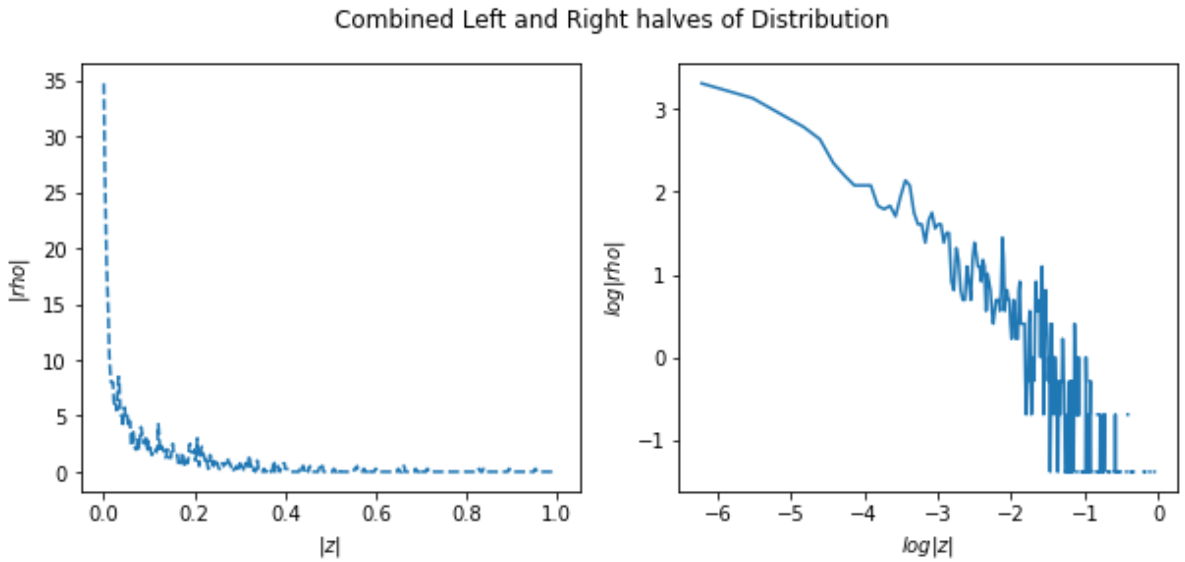

ΔE_i vs E_i @ i = 32



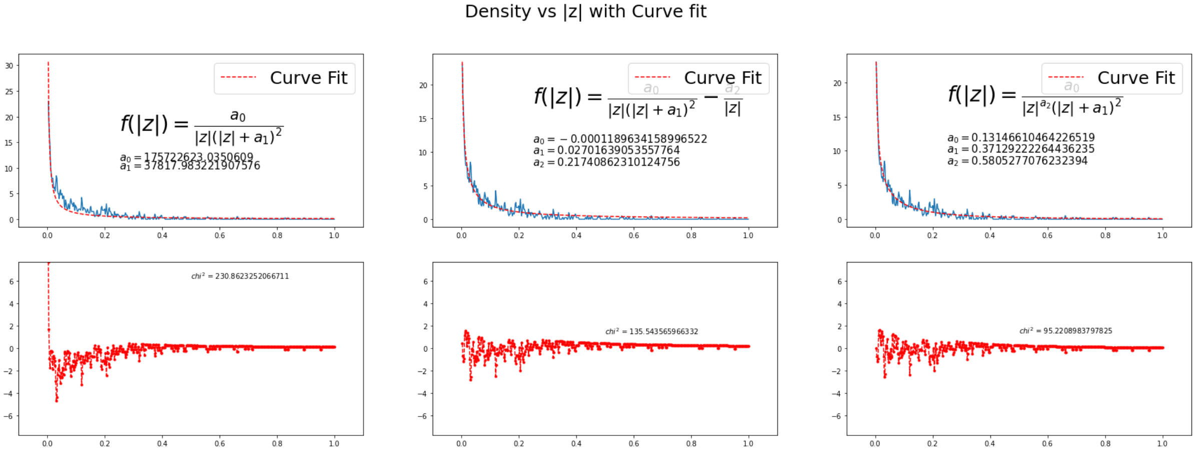


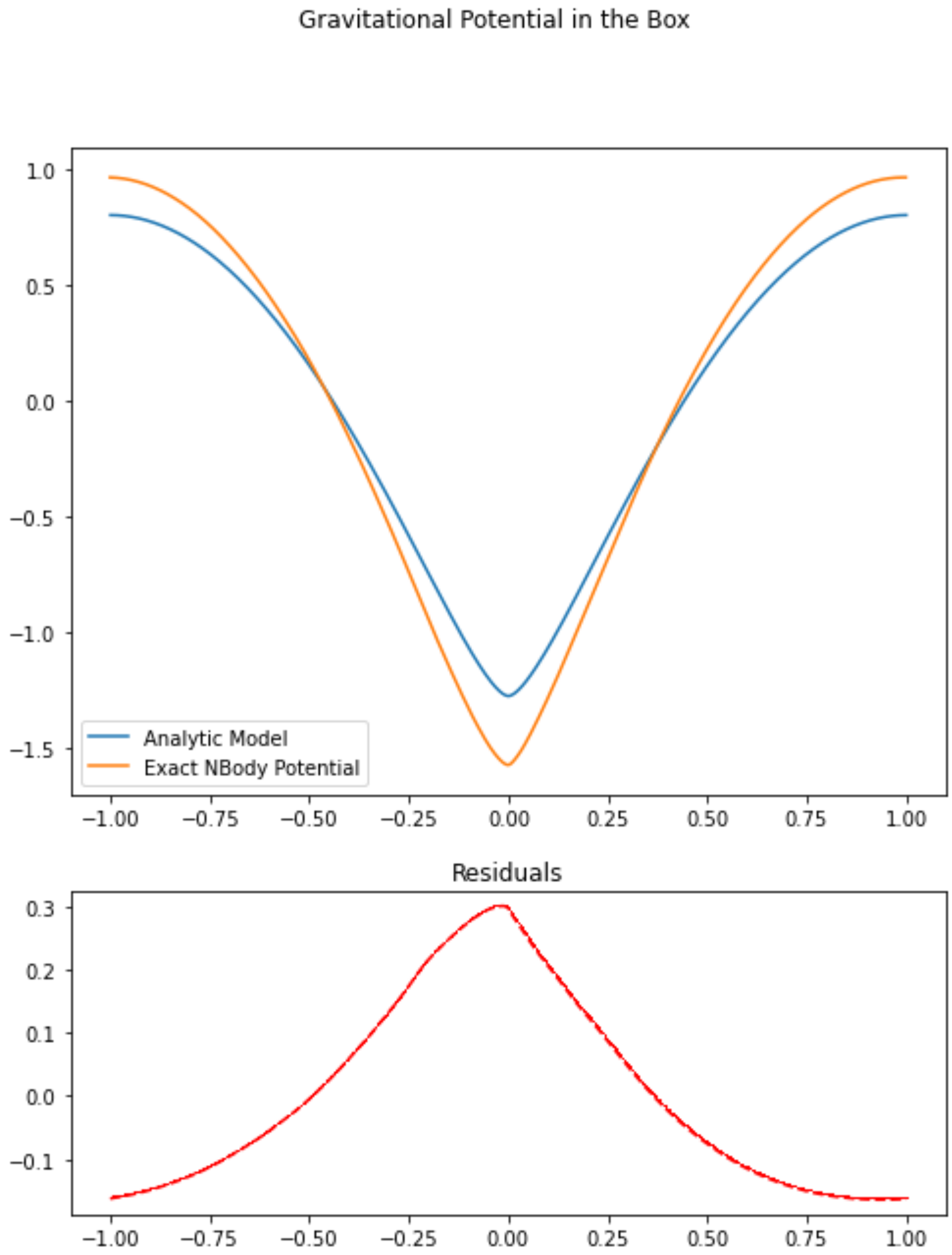


```
/home/boris/Documents/Research/FDM_n_Bodies/1D_Codes/Non-Dim/Analysis/Analy
sis.py:442: RuntimeWarning: divide by zero encountered in log
  ax[1].plot(np.log(z_right),np.log(rho_whole))
```

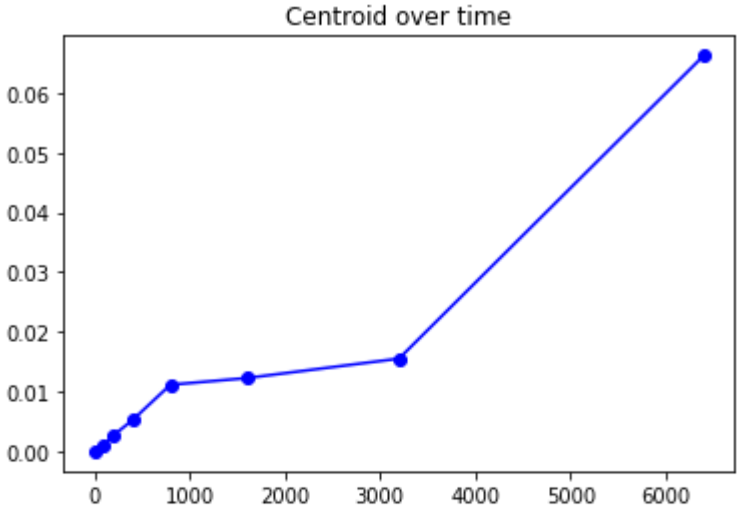


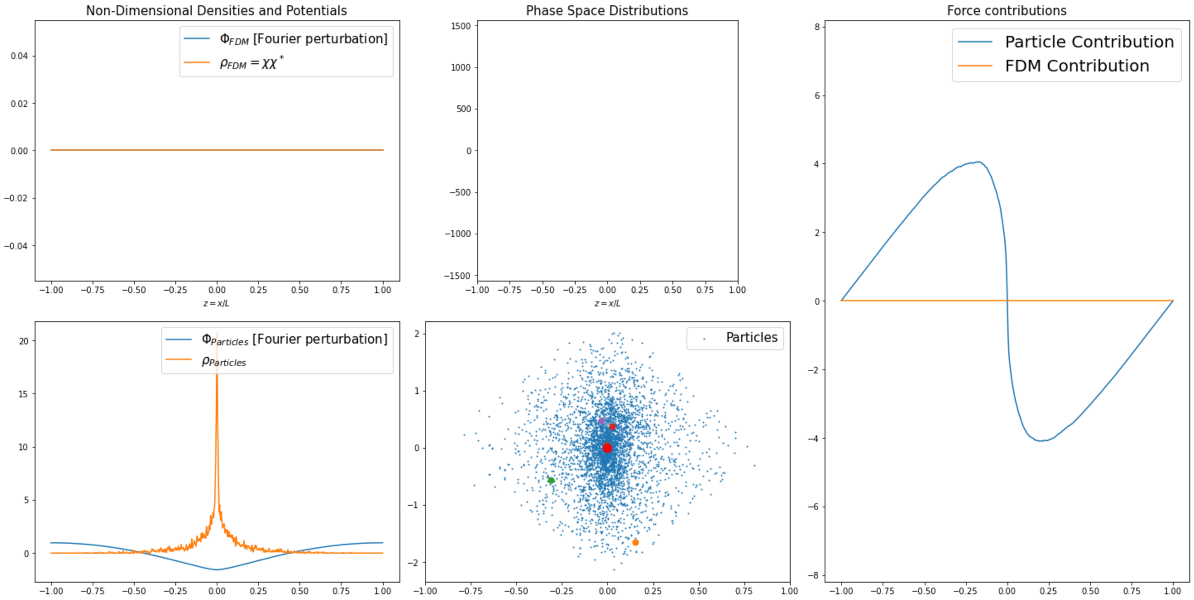
```
Check
Check
Check
```



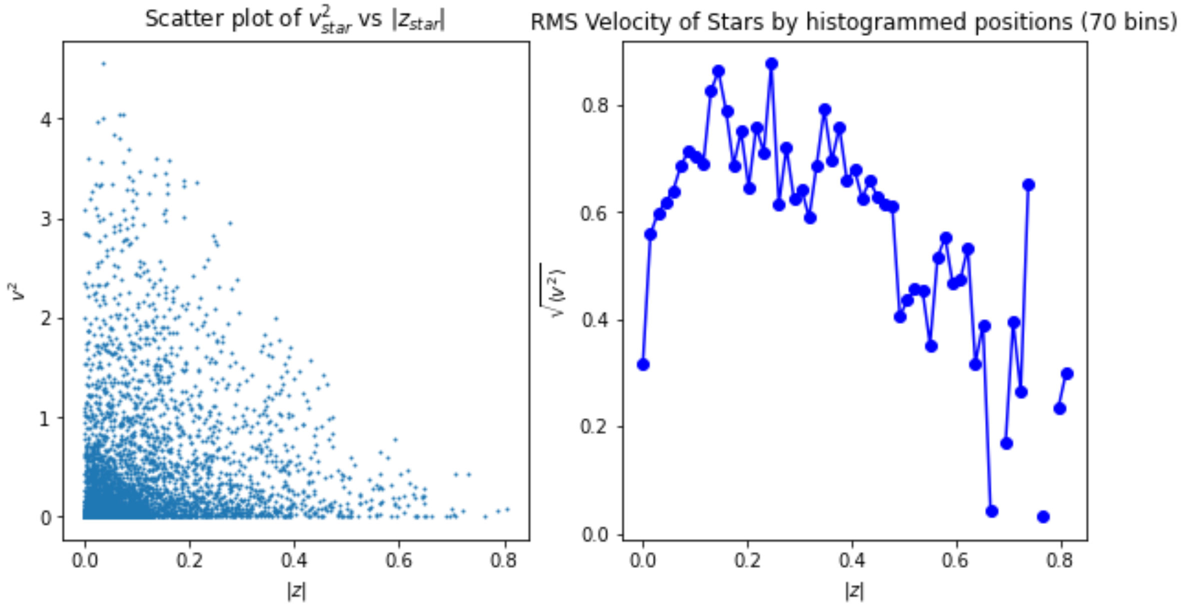


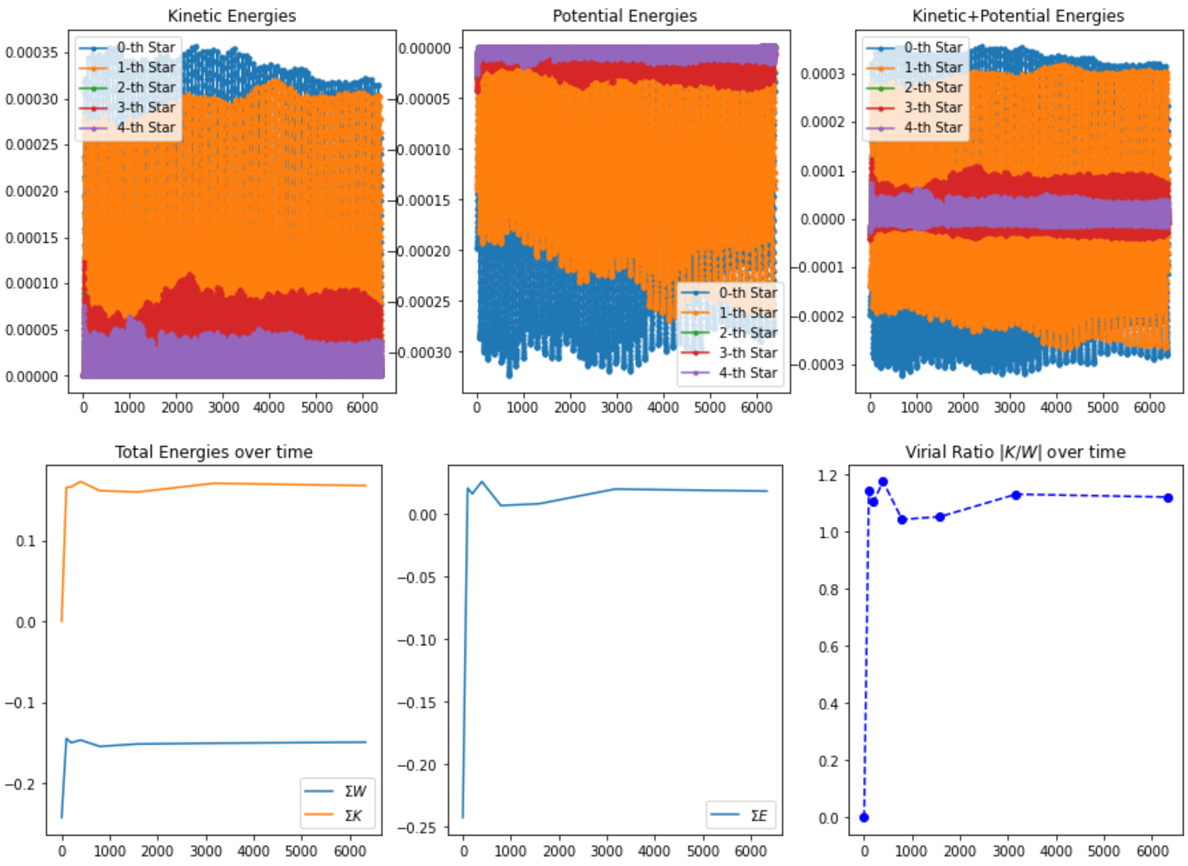
```
----------New Analysis--------
r = 1 mu = 1 Num_bosons = 0 sigma = 0.0002 Num_stars = 5000
```

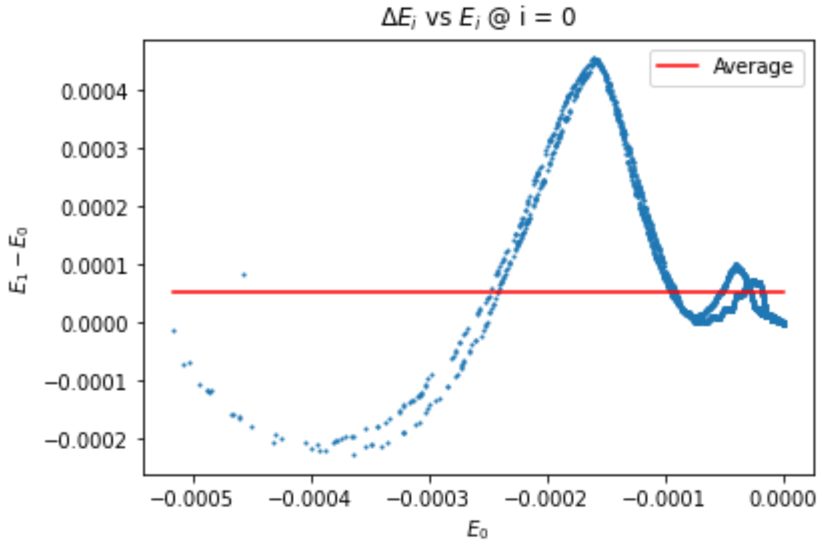


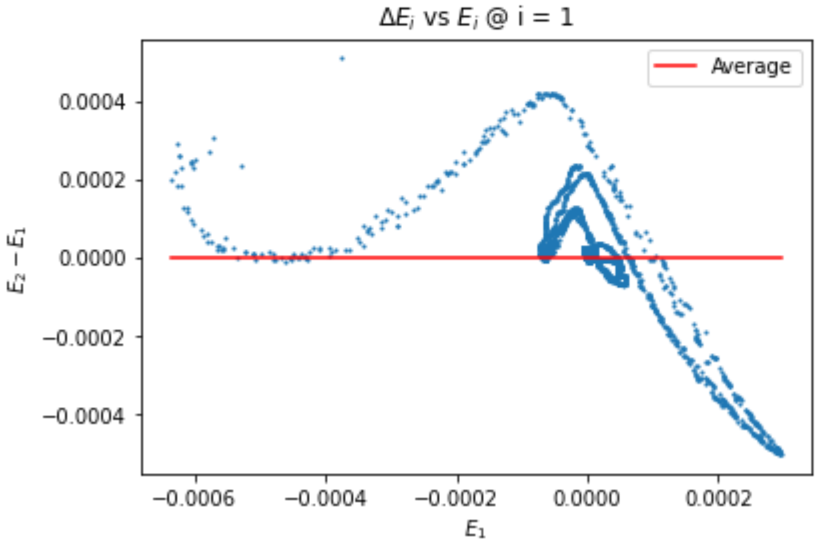


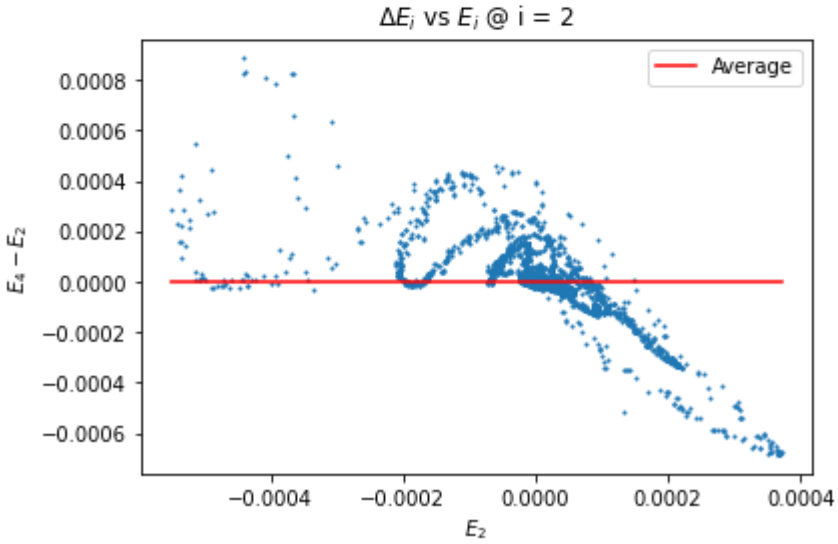
```
v_rms = 0.5851737606787921
z_rms = 0.1541549084183391
K_avg = 0.5*m*v_rms^2 = 0.17121416509348014 (m=1)
=> 2*K_avg = 0.3424283301869603
W_avg = 770.7745420916955
------------------
K_tot = 0.1712141650934794
K_avg = 3.424283301869588e-05
W_tot = -0.14558193639607853
W_avg = -2.9116387279215706e-05
```

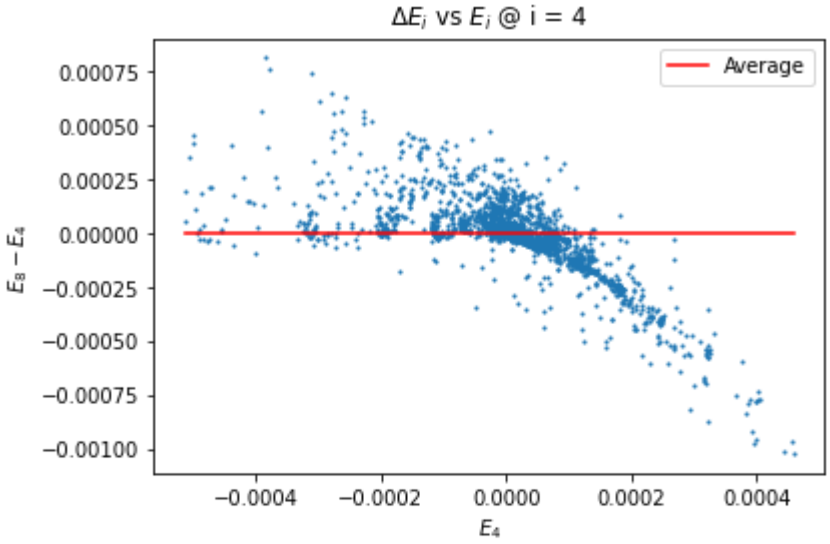


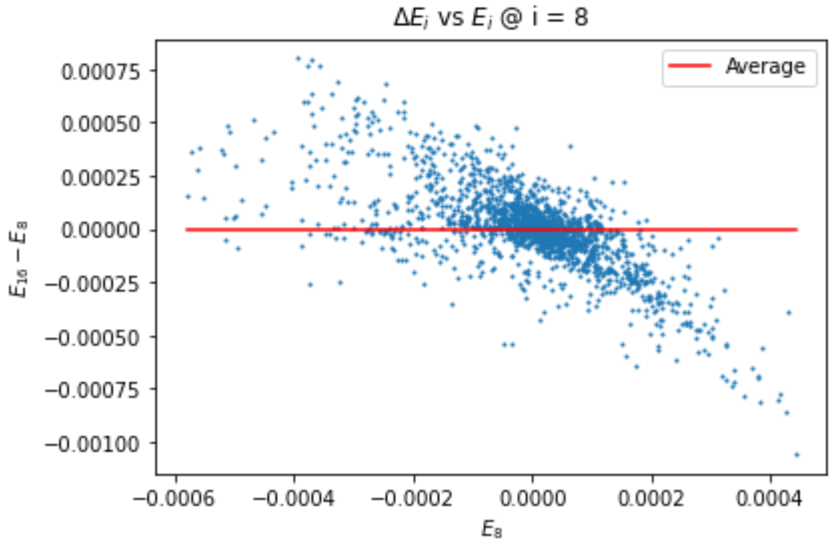

Kinetic Energies
0.00035
0.00030
0.00025
0.00020
0.00015
0.00010
0.00005
0.00000
0-th Star
1-th Star
2-th Star
3-th Star
4-th Star
0 1000 2000 3000 4000 5000 6000
Potential Energies
0.00000
−0.00005
−0.00010
−0.00015
−0.00020
−0.00025
−0.00030
0-th Star
1-th Star
2-th Star
3-th Star
4-th Star
0 1000 2000 3000 4000 5000 6000
Kinetic+Potential Energies
0.0003
0.0002
0.0001
0.0000
−0.0001
−0.0002
−0.0003
0-th Star
1-th Star
2-th Star
3-th Star
4-th Star
0 1000 2000 3000 4000 5000 6000
Total Energies over time
0.1
0.0
−0.1
−0.2
ΣW
ΣK
0 1000 2000 3000 4000 5000 6000
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
ΣE
0 1000 2000 3000 4000 5000 6000
Virial Ratio |K/W| over time
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0 1000 2000 3000 4000 5000 6000

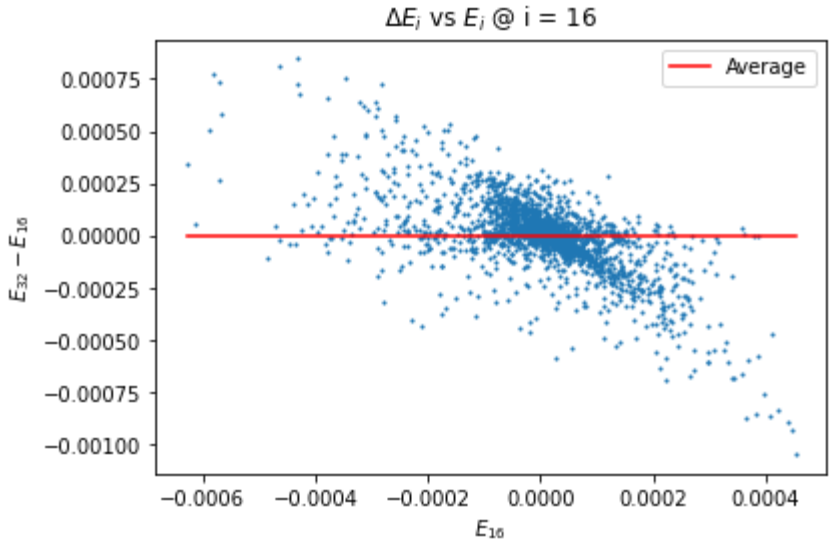

ΔE_i vs E_i @ i = 0
Average
$E_1 - E_0$
0.0004
0.0003
0.0002
0.0001
0.0000
−0.0001
−0.0002
−0.0005
−0.0004
−0.0003
−0.0002
−0.0001
0.0000
E_0

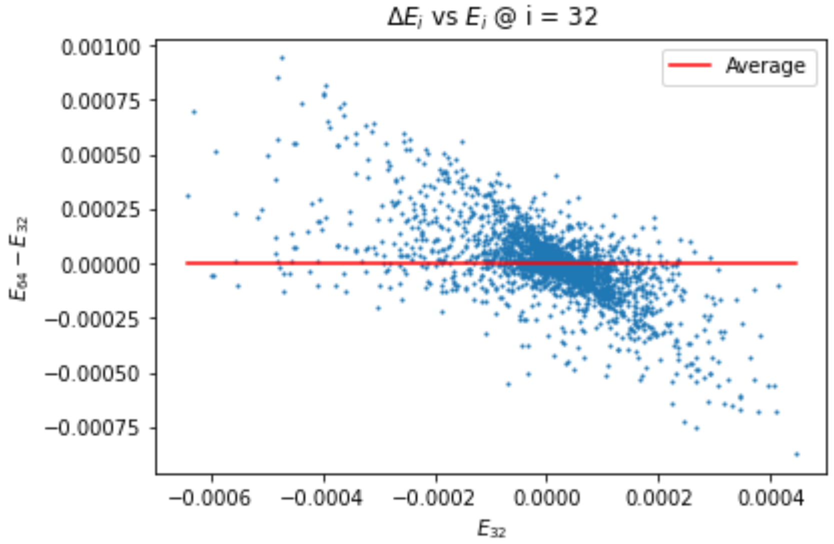

ΔE_i vs E_i @ i = 1
Average
$E_2 - E_1$
0.0004
0.0002
0.0000
−0.0002
−0.0004
−0.0006
−0.0004
−0.0002
0.0000
0.0002
E_1

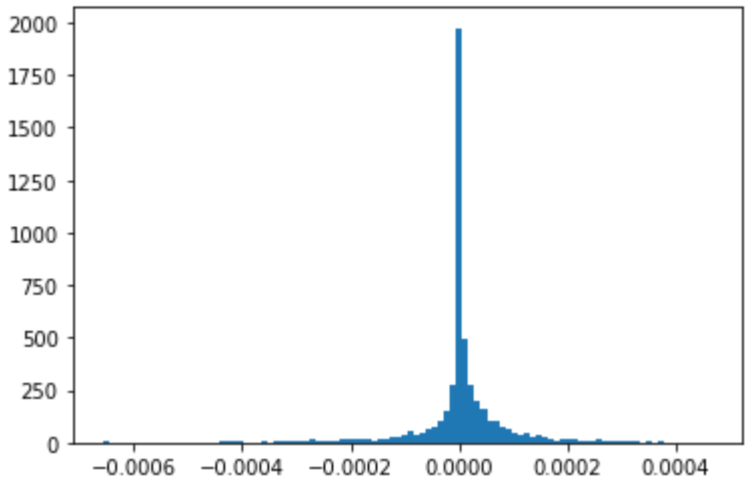

ΔE_i vs E_i @ i = 2
Average
$E_4 - E_2$
0.0008
0.0006
0.0004
0.0002
0.0000
−0.0002
−0.0004
−0.0006
−0.0004
−0.0002
0.0000
0.0002
0.0004
E_2

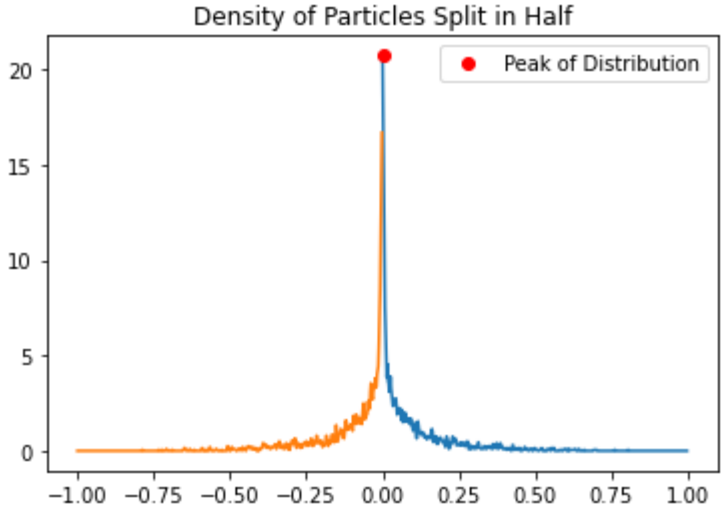

ΔE_i vs E_i @ i = 4
Average
$E_8 - E_4$
0.00075
0.00050
0.00025
0.00000
−0.00025
−0.00050
−0.00075
−0.00100
−0.0004
−0.0002
0.0000
0.0002
0.0004
E_4

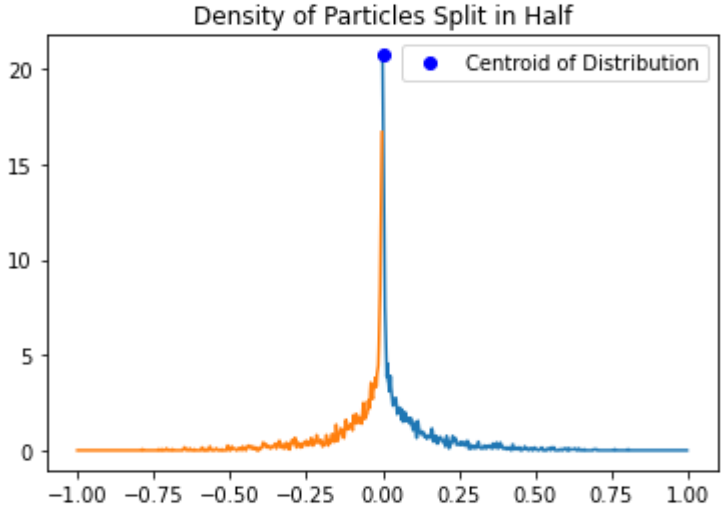

ΔE_i vs E_i @ i = 8
Average
0.00075
0.00050
0.00025
0.00000
−0.00025
−0.00050
−0.00075
−0.00100
$E_{16} - E_8$
−0.0006
−0.0004
−0.0002
0.0000
0.0002
0.0004
E_8

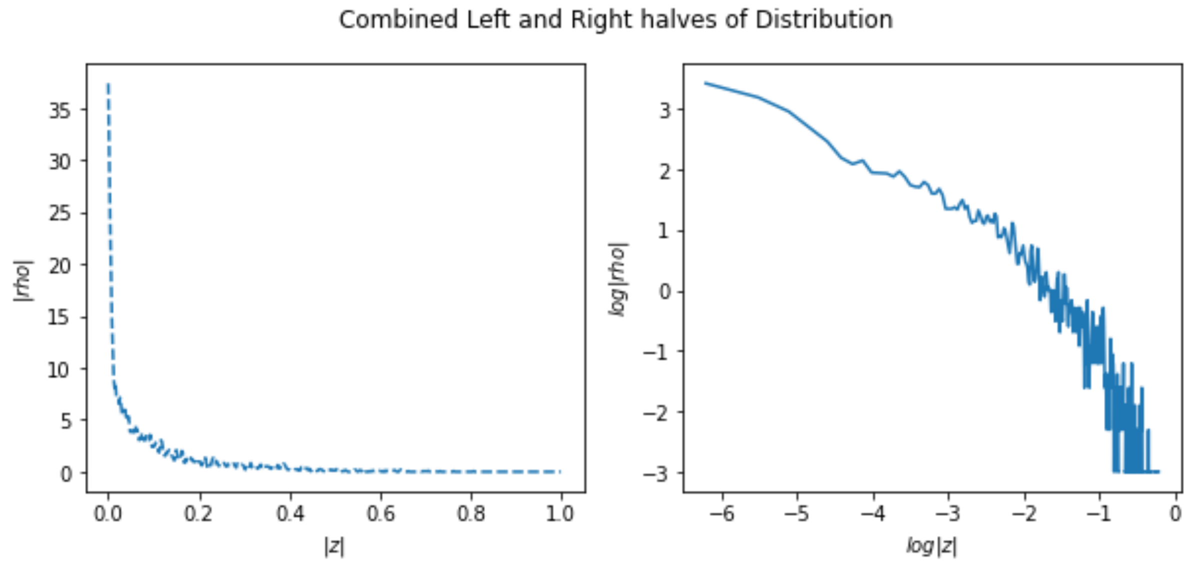

ΔE_i vs E_i @ i = 16
Average
0.00075
0.00050
0.00025
0.00000
−0.00025
−0.00050
−0.00075
−0.00100
$E_{32} - E_{16}$
−0.0006
−0.0004
−0.0002
0.0000
0.0002
0.0004
E_{16}


ΔE_i vs E_i @ i = 32
Average
0.00100
0.00075
0.00050
0.00025
0.00000
−0.00025
−0.00050
−0.00075
$E_{64} - E_{32}$
−0.0006
−0.0004
−0.0002
0.0000
0.0002
0.0004
E_{32}

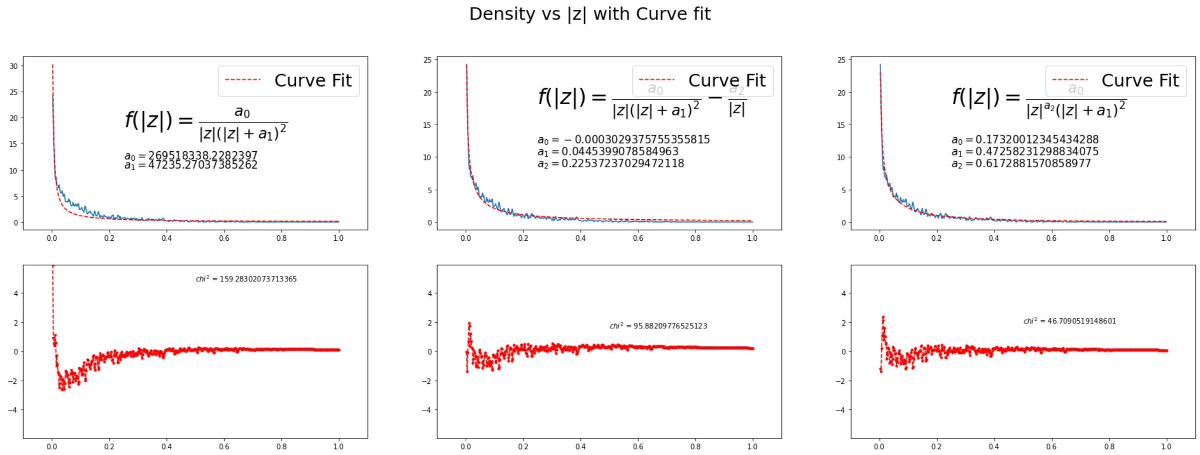

2000
1750
1500
1250
1000
750
500
250
0
−0.0006
−0.0004
−0.0002
0.0000
0.0002
0.0004

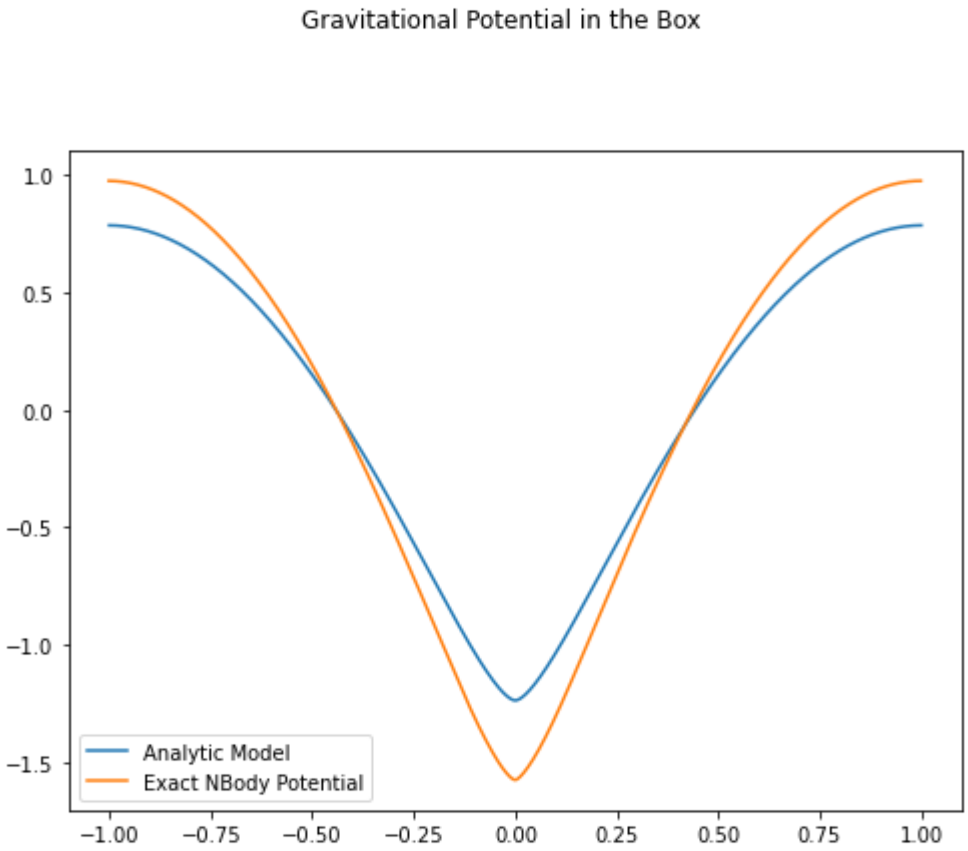

Density of Particles Split in Half
● Peak of Distribution
20
15
10
5
0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

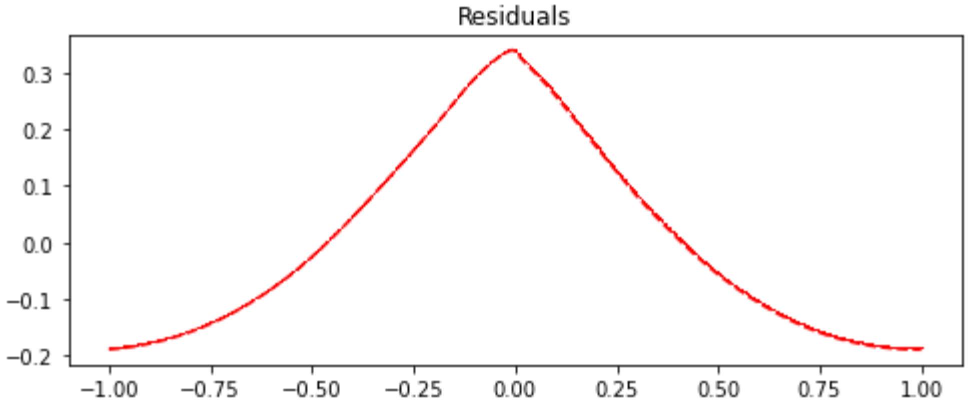

Density of Particles Split in Half
● Centroid of Distribution
20
15
10
5
0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00



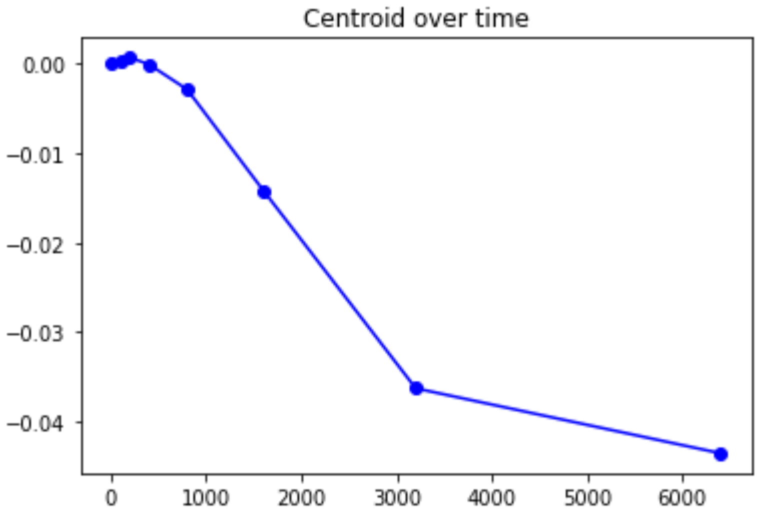
```
Check
Check
Check
```

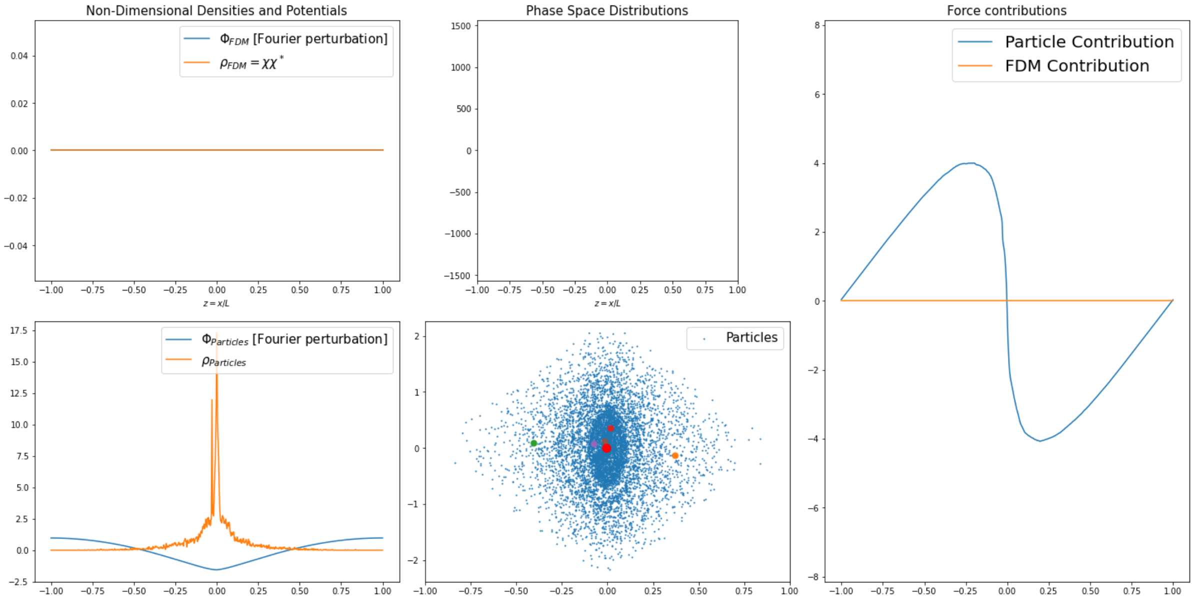




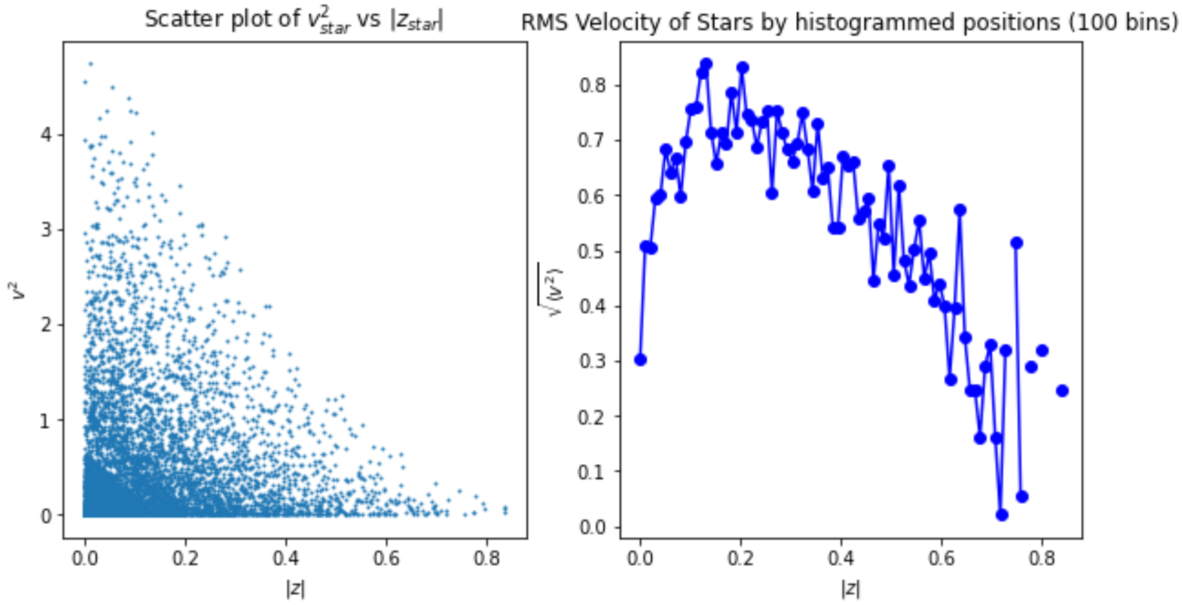


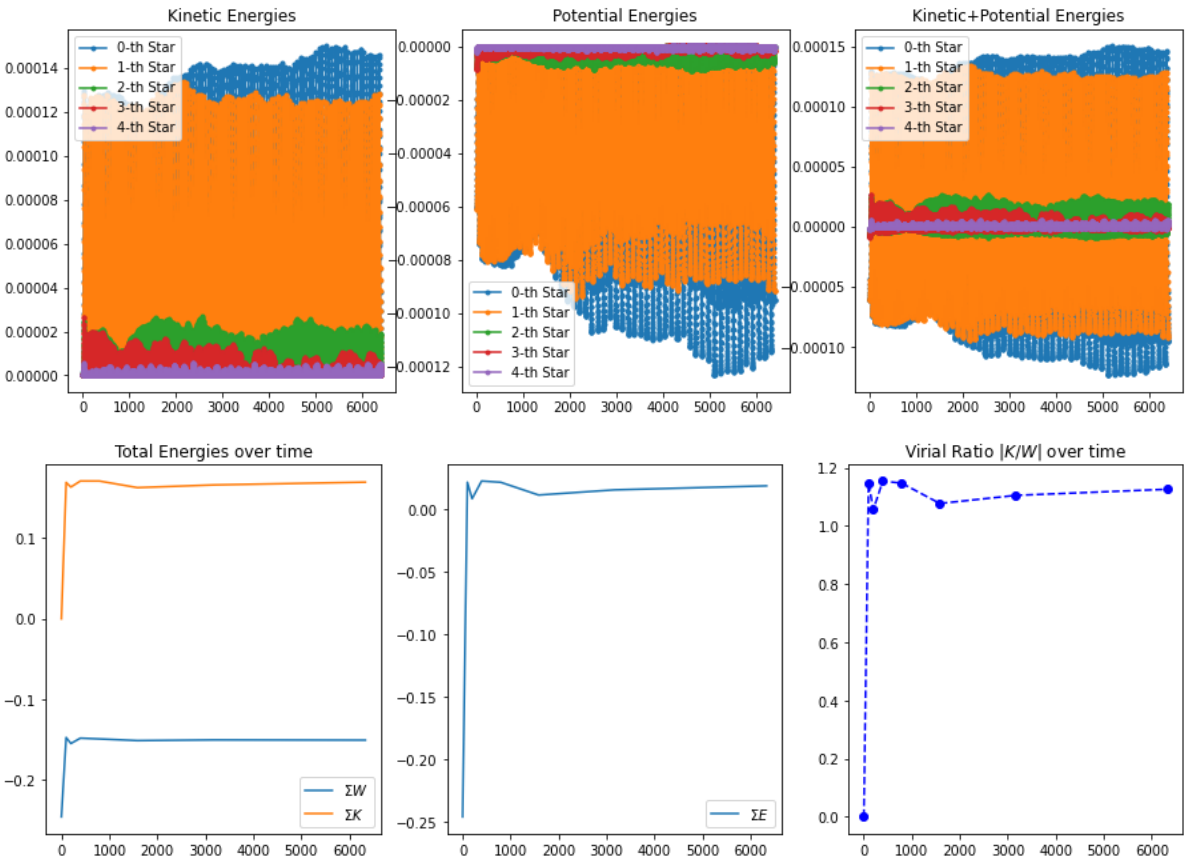
```
----------New Analysis--------
r = 1 mu = 1 Num_bosons = 0 sigma = 0.0001 Num_stars = 10000
```

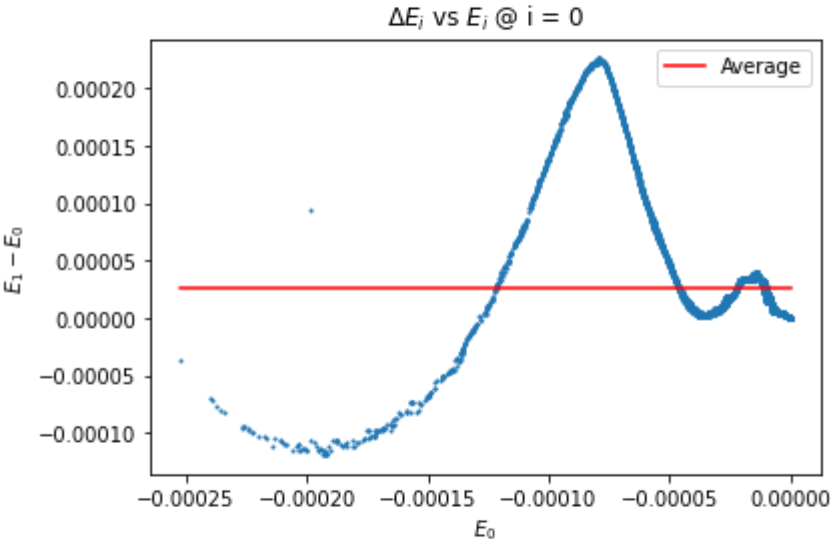


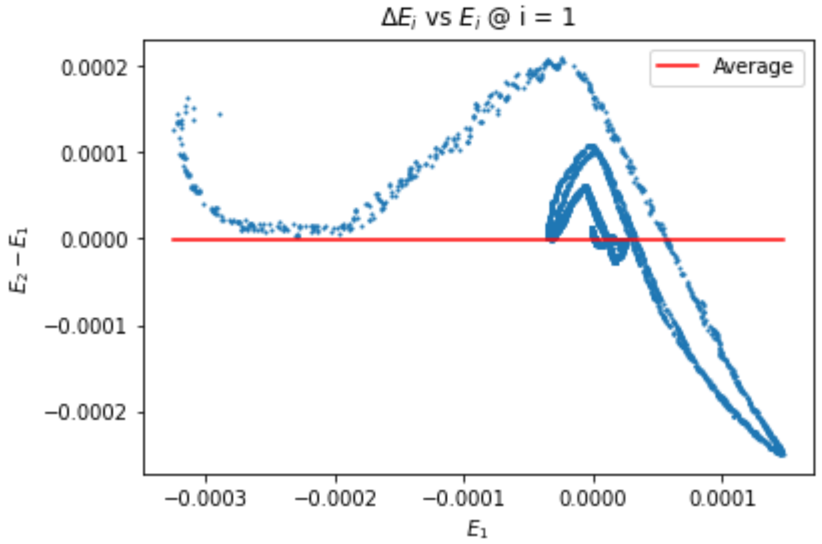


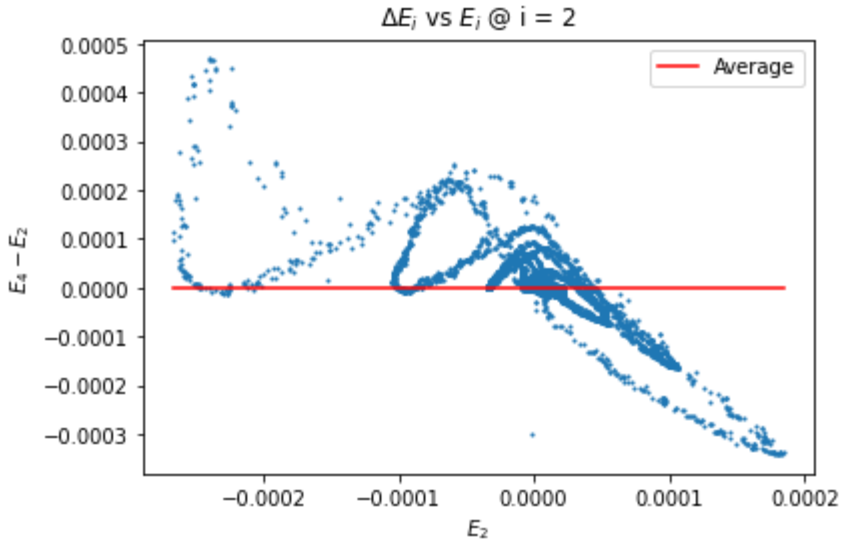
```
v_rms = 0.581649317774426
z_rms = 0.15731951911022882
K_avg = 0.5*m*v_rms^2 = 0.16915796443372758 (m=1)
=> 2*K_avg = 0.33831592886745515
W_avg = 1573.1951911022882
------------------
K_tot = 0.1691579644337266
K_avg = 1.691579644337266e-05
W_tot = -0.15082252821733694
W_avg = -1.5082252821733694e-05
```

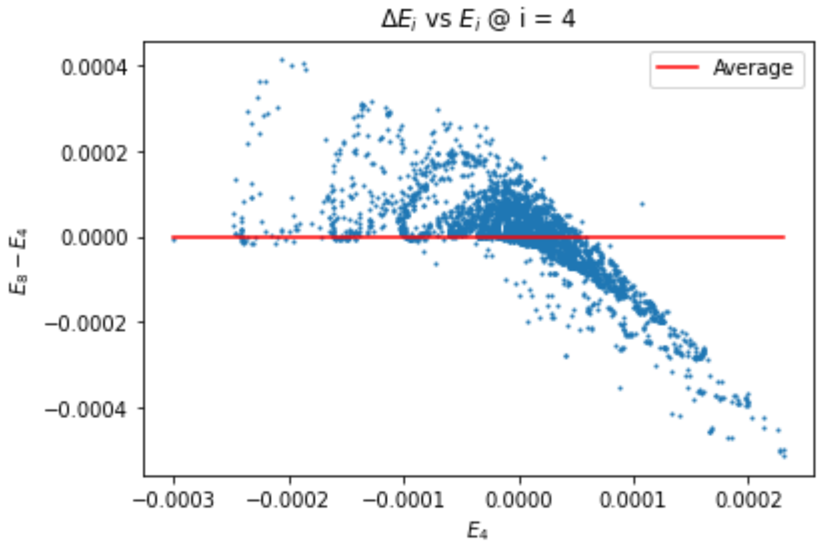


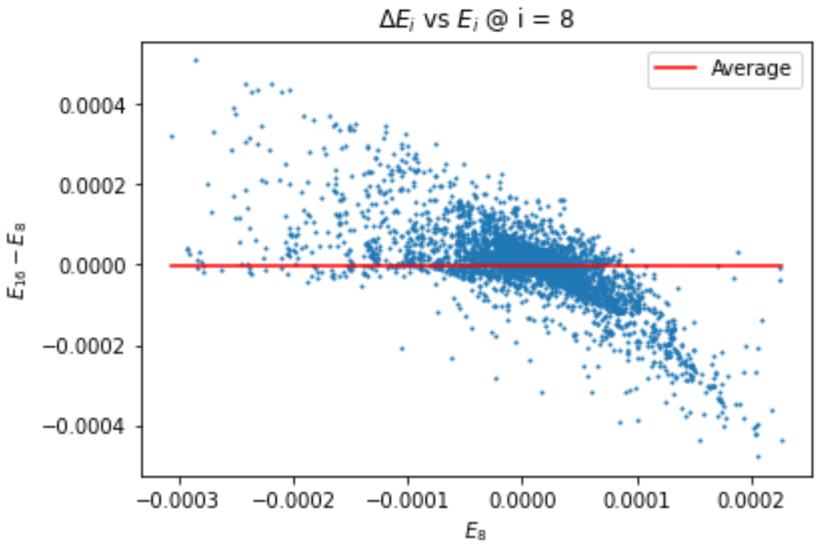

Kinetic Energies
0.00014
0.00012
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
0-th Star
1-th Star
2-th Star
3-th Star
4-th Star
Potential Energies
0.00000
−0.00002
−0.00004
−0.00006
−0.00008
−0.00010
−0.00012
Kinetic+Potential Energies
0.00015
0.00010
0.00005
0.00000
−0.00005
−0.00010
0
1000
2000
3000
4000
5000
6000
Total Energies over time
0.1
0.0
−0.1
−0.2
ΣW
ΣK
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
ΣE
Virial Ratio |K/W| over time
1.2
1.0
0.8
0.6
0.4
0.2
0.0

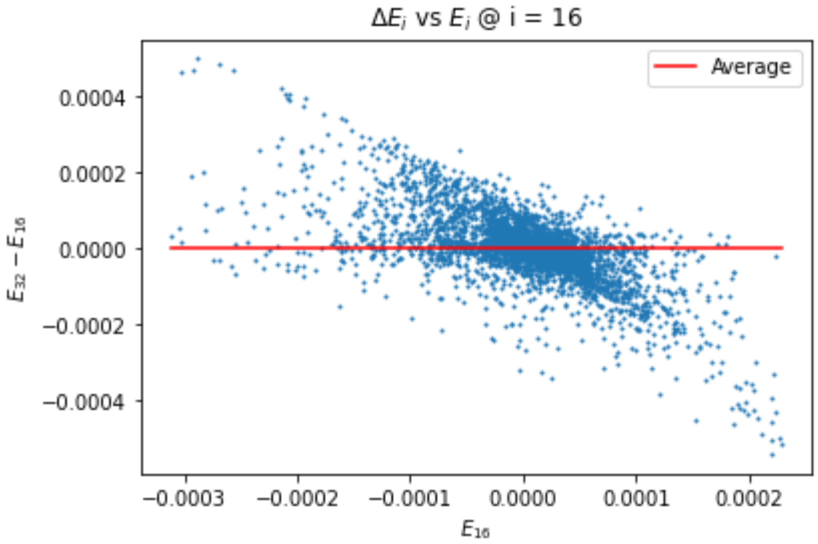

ΔE_i vs E_i @ i = 0
Average
$E_1 - E_0$
0.00020
0.00015
0.00010
0.00005
0.00000
−0.00005
−0.00010
−0.00025
−0.00020
−0.00015
−0.00010
−0.00005
0.00000
E_0

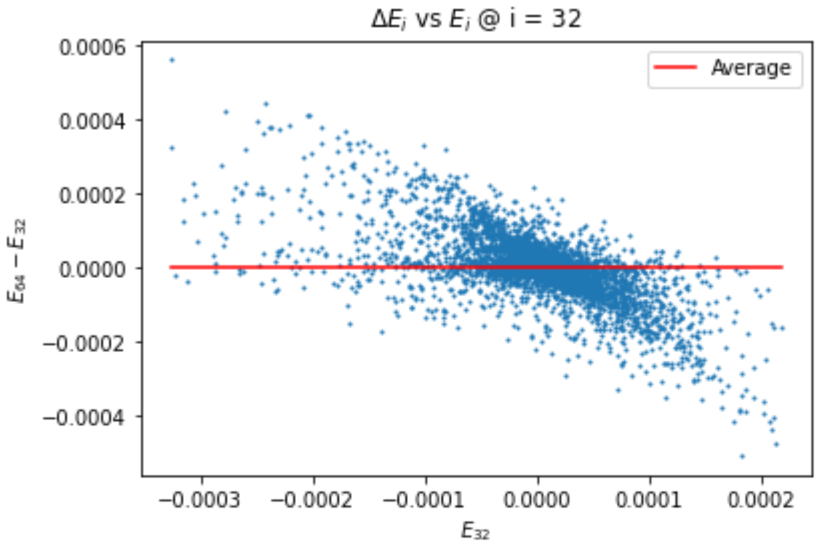

ΔE_i vs E_i @ i = 1
Average
$E_2 - E_1$
0.0002
0.0001
0.0000
−0.0001
−0.0002
−0.0003
−0.0002
−0.0001
0.0000
0.0001
E_1

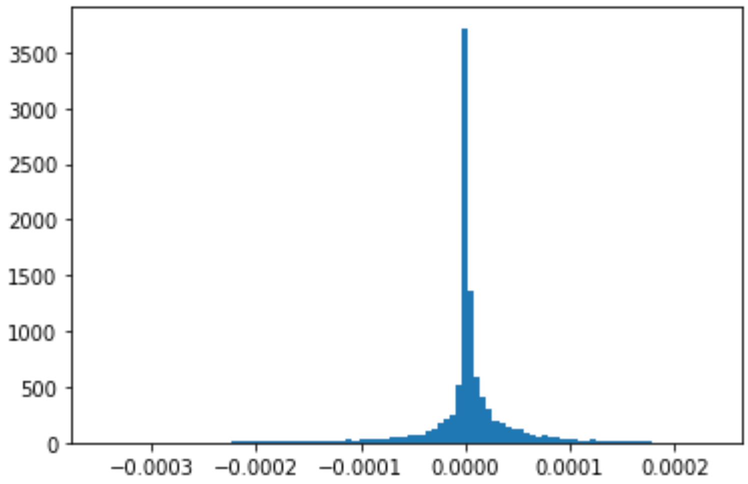

ΔE_i vs E_i @ i = 2
Average
$E_4 - E_2$
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
−0.0001
−0.0002
−0.0003
−0.0002
−0.0001
0.0000
0.0001
0.0002
E_2

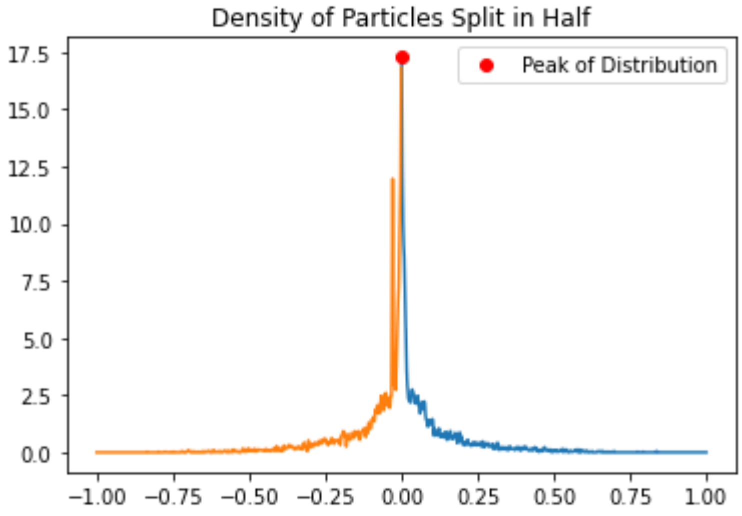

ΔE_i vs E_i @ i = 4
Average
$E_8 - E_4$
0.0004
0.0002
0.0000
−0.0002
−0.0004
−0.0003
−0.0002
−0.0001
0.0000
0.0001
0.0002
E_4

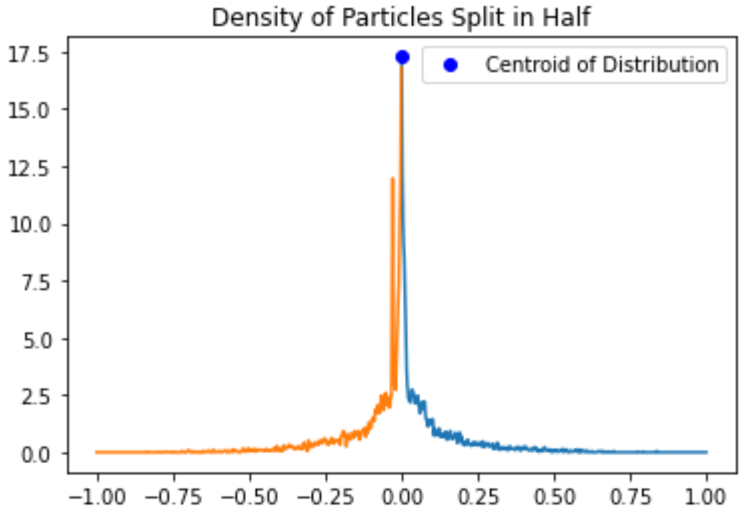

ΔE_i vs E_i @ i = 8
Average
$E_{16} - E_8$
0.0004
0.0002
0.0000
−0.0002
−0.0004
−0.0003
−0.0002
−0.0001
0.0000
0.0001
0.0002
E_8

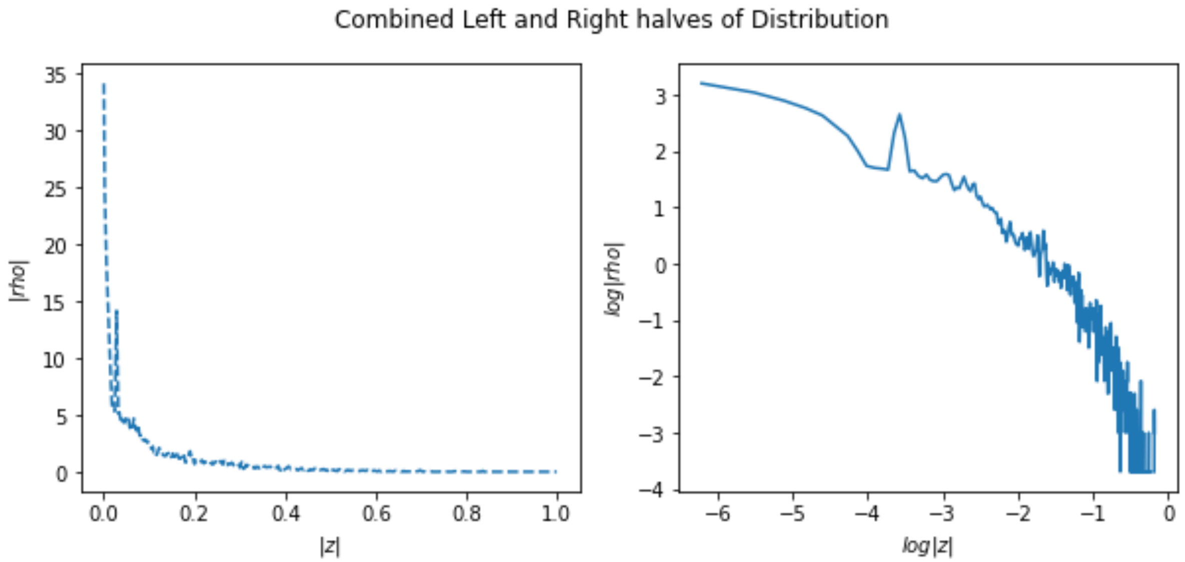

ΔE_i vs E_i @ i = 16
Average
$E_{32} - E_{16}$
0.0004
0.0002
0.0000
−0.0002
−0.0004
−0.0003
−0.0002
−0.0001
0.0000
0.0001
0.0002
E_{16}


ΔE_i vs E_i @ i = 32
Average
$E_{64} - E_{32}$
0.0006
0.0004
0.0002
0.0000
−0.0002
−0.0004
−0.0003
−0.0002
−0.0001
0.0000
0.0001
0.0002
E_{32}


3500
3000
2500
2000
1500
1000
500
0
−0.0003
−0.0002
−0.0001
0.0000
0.0001
0.0002

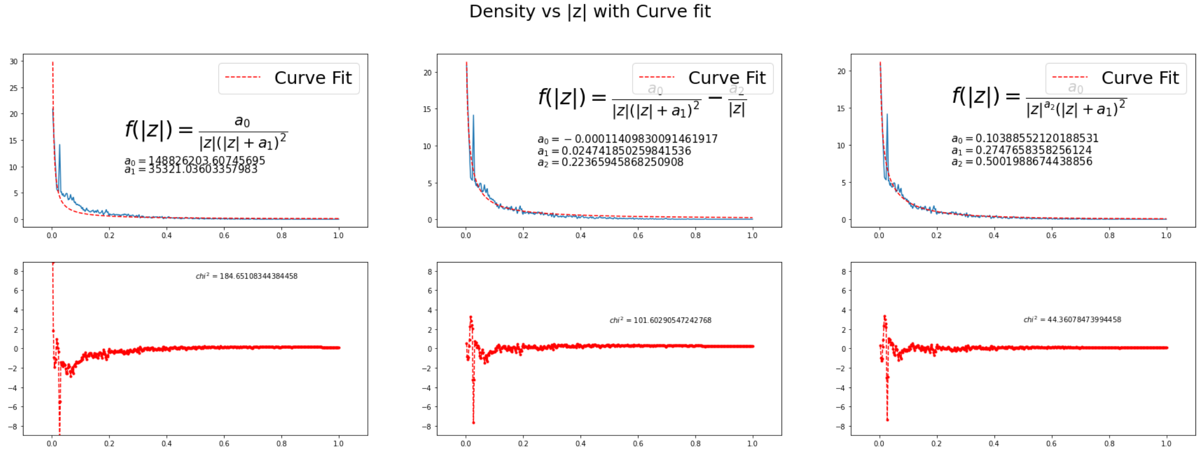

Density of Particles Split in Half
Peak of Distribution
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


Density of Particles Split in Half
Centroid of Distribution
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

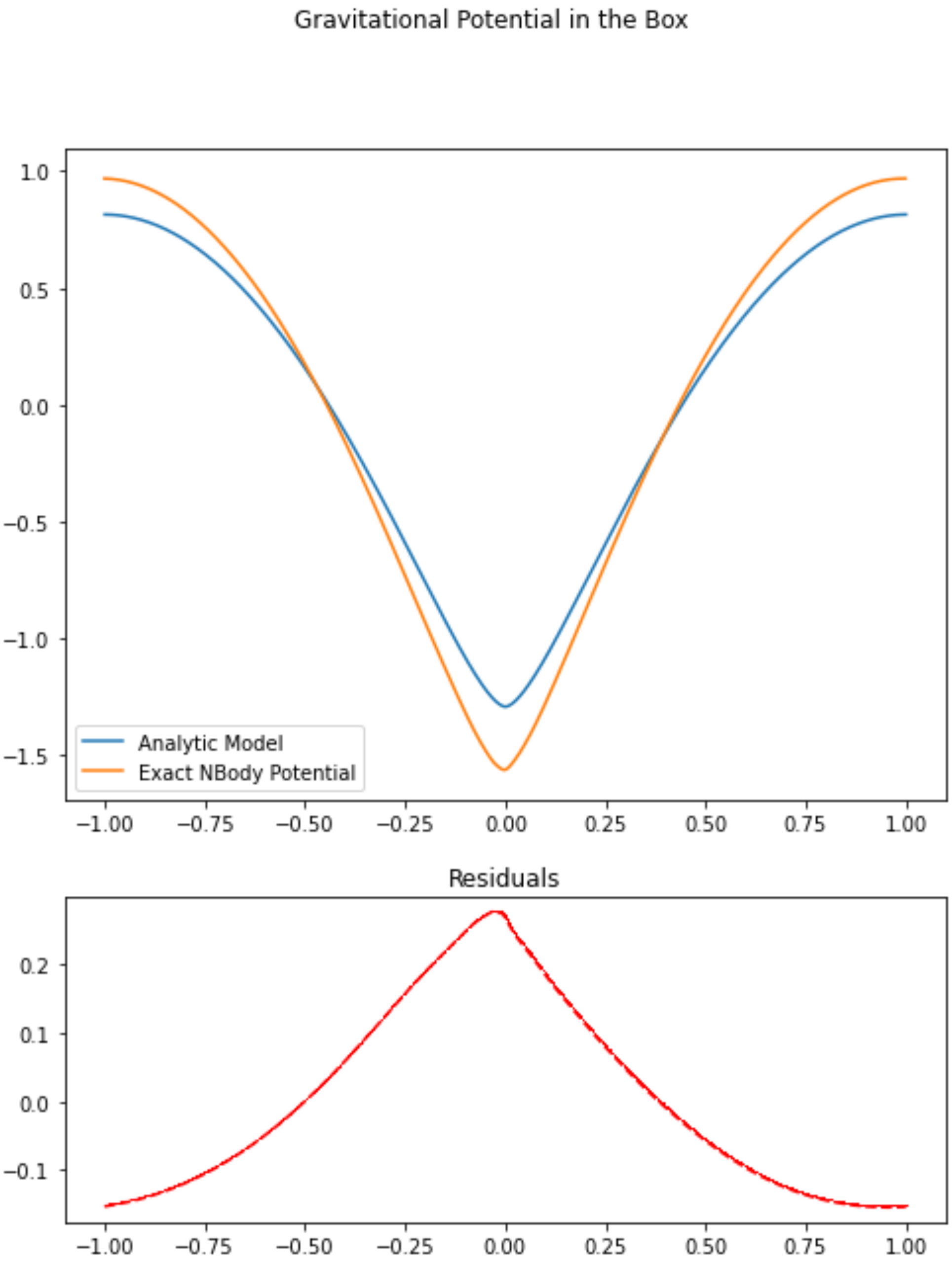


Combined Left and Right halves of Distribution

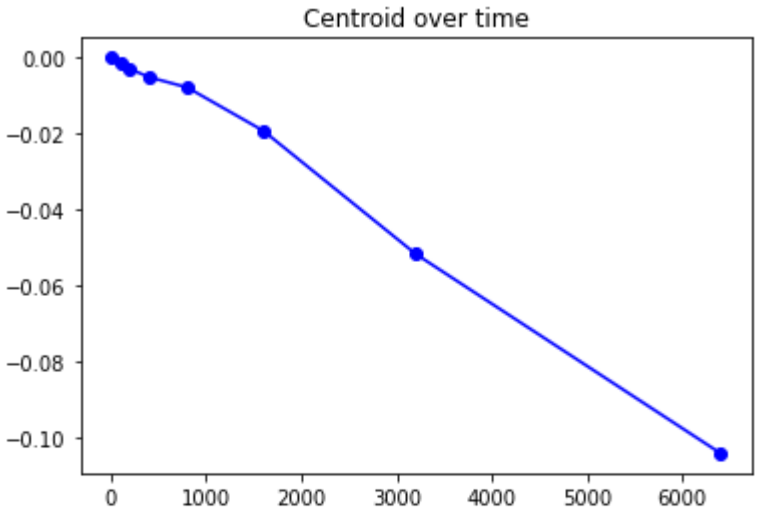
```
Check
Check
Check
```

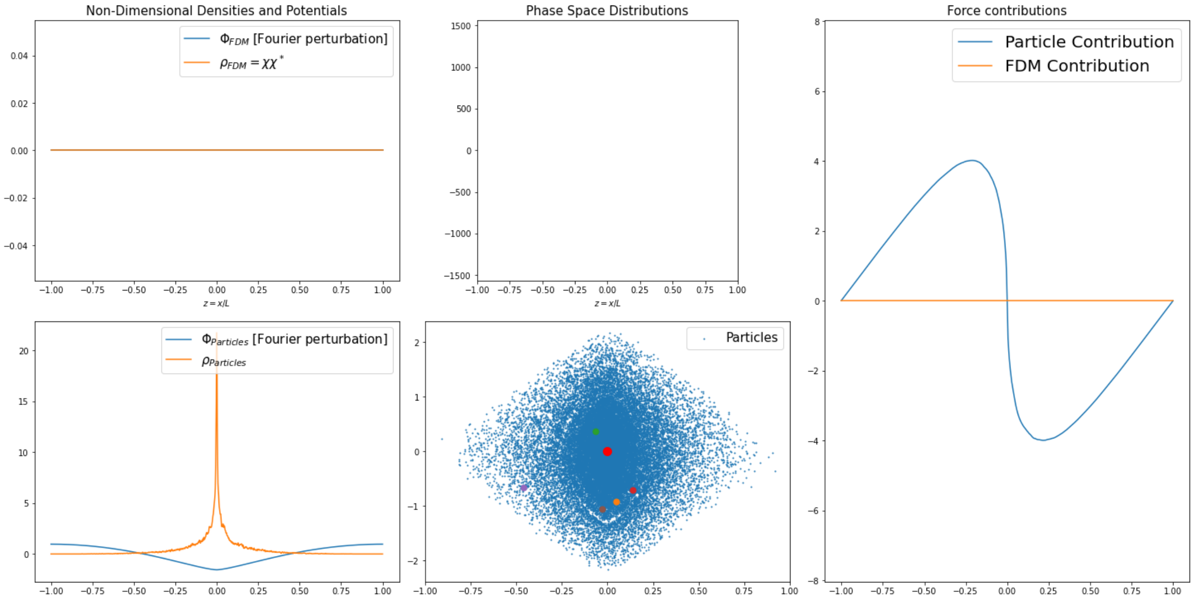


Density vs |z| with Curve fit

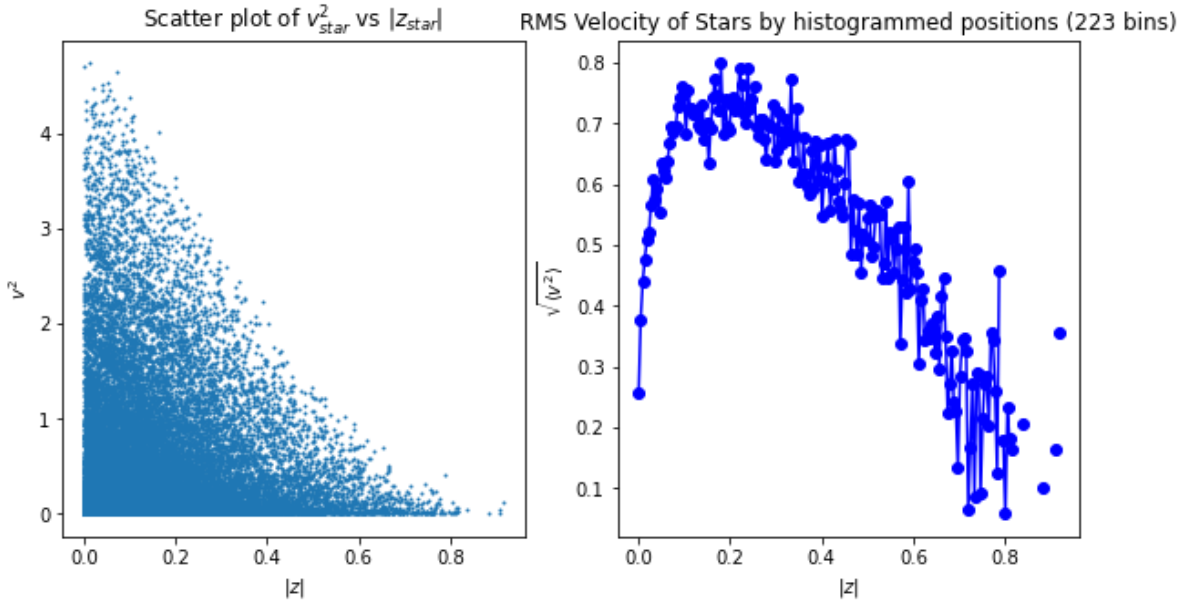


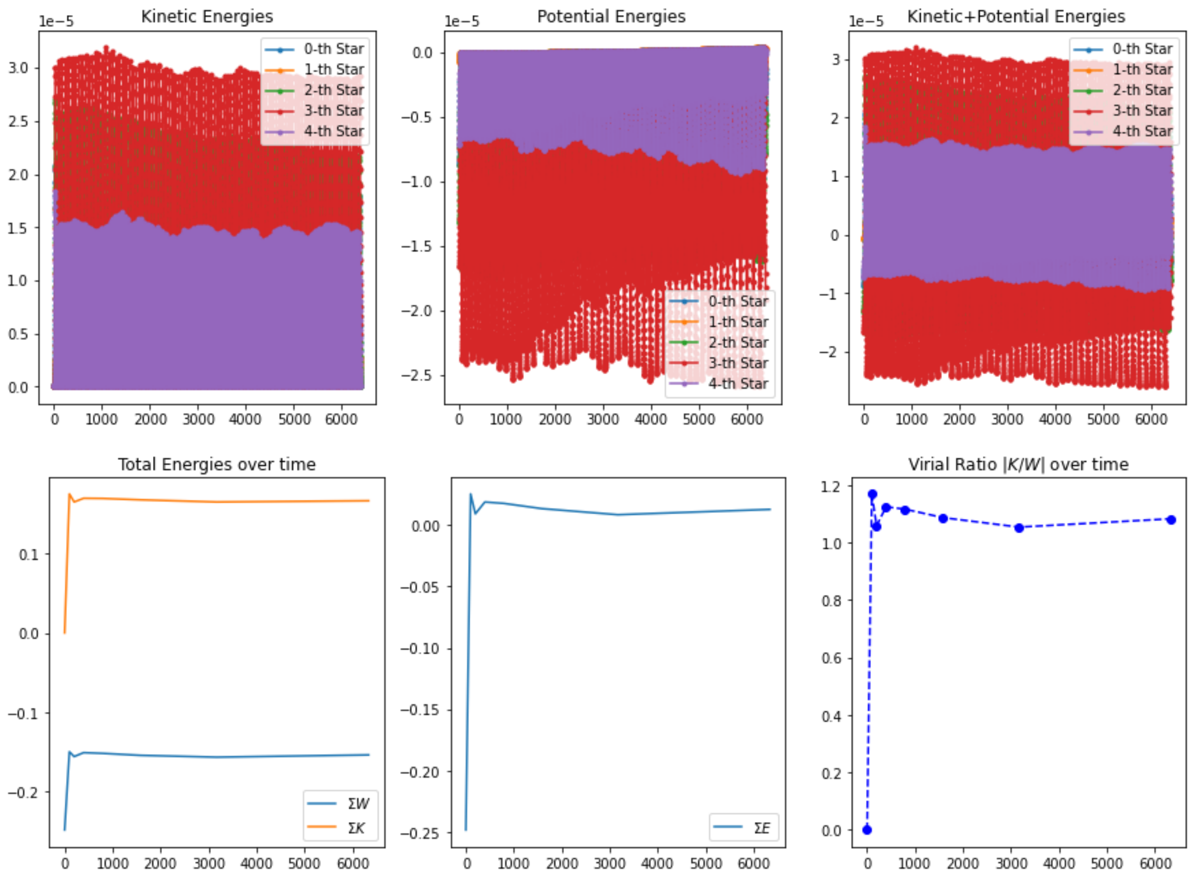
```
----------New Analysis--------
r = 1 mu = 1 Num_bosons = 0 sigma = 2e-05 Num_stars = 50000
```

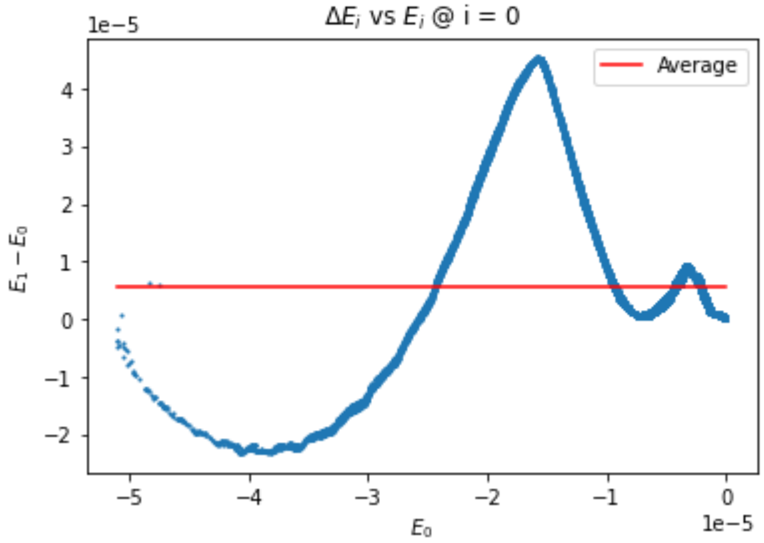


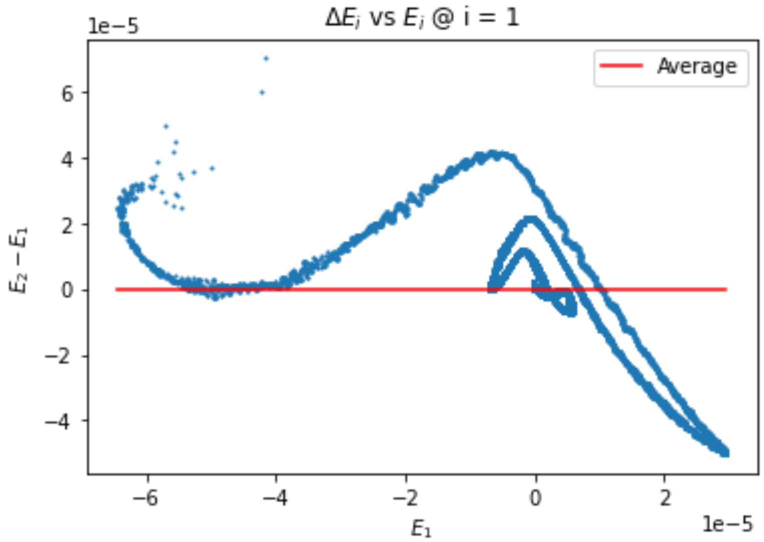


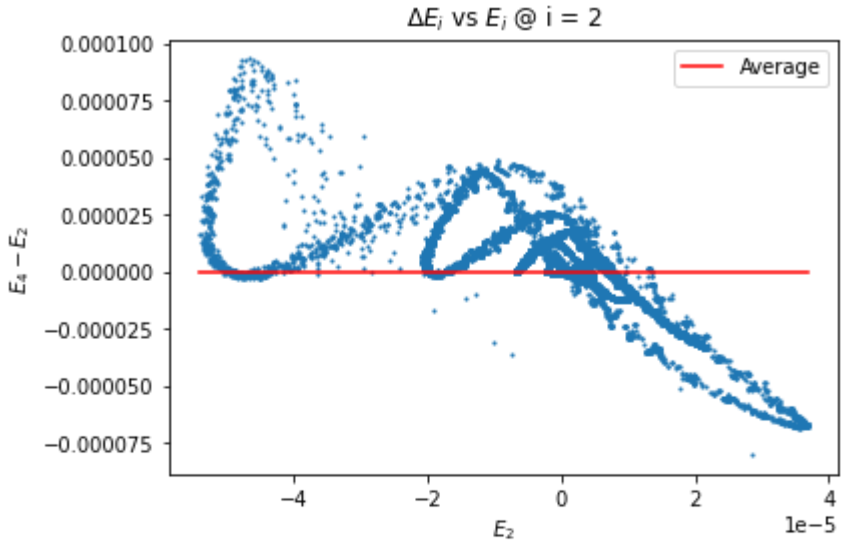
```
v_rms = 0.5731593297761615
z_rms = 0.15992388602013577
K_avg = 0.5*m*v_rms^2 = 0.1642558086547293 (m=1)
=> 2*K_avg = 0.3285116173094586
W_avg = 7996.194301006788
------------------
K_tot = 0.16425580865473047
K_avg = 3.285116173094609e-06
W_tot = -0.1556381101749108
W_avg = -3.112762203498216e-06
```

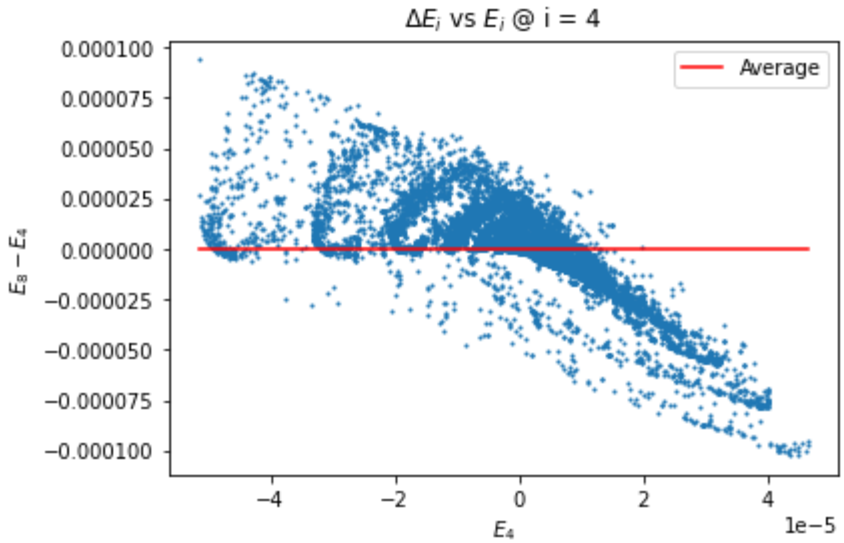


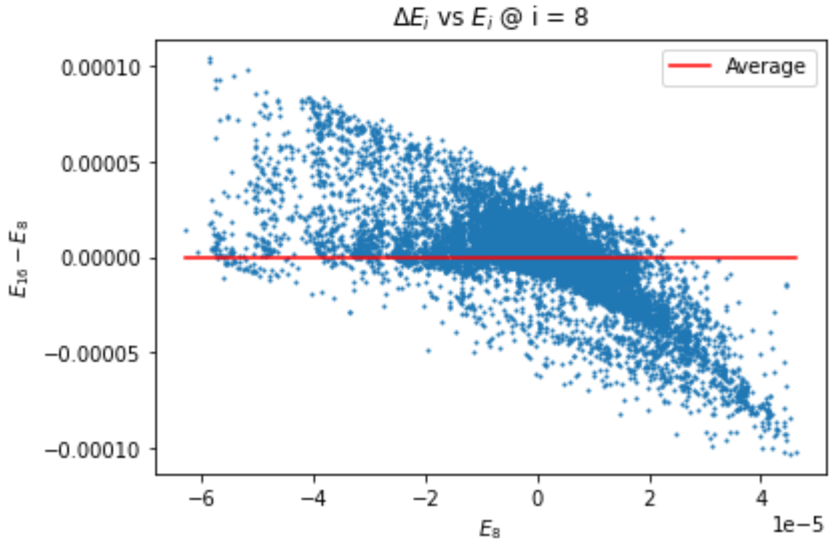

Kinetic Energies
1e−5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0-th Star
1-th Star
2-th Star
3-th Star
4-th Star
0
1000
2000
3000
4000
5000
6000
Potential Energies
1e−5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
Kinetic+Potential Energies
1e−5
3
2
1
0
−1
−2
Total Energies over time
0.1
0.0
−0.1
−0.2
ΣW
ΣK
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
ΣE
Virial Ratio |K/W| over time
1.2
1.0
0.8
0.6
0.4
0.2
0.0

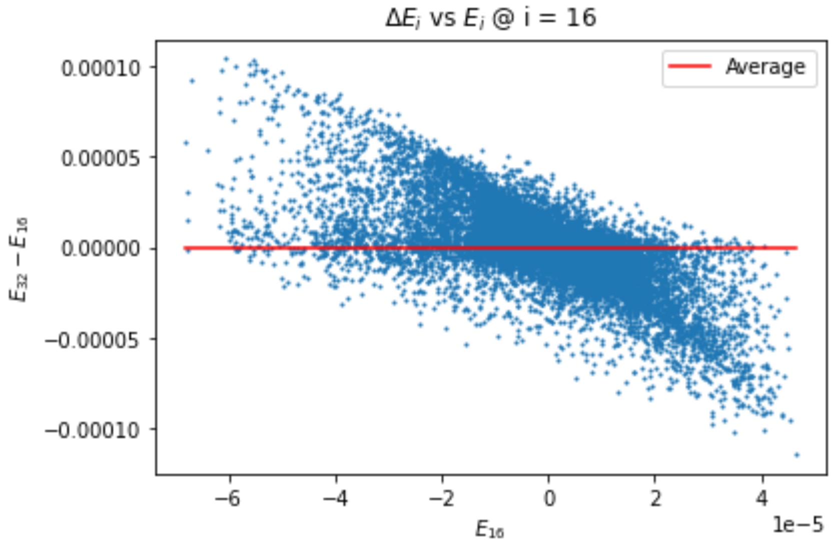

ΔE_i vs E_i @ i = 0
1e−5
Average
4
3
2
1
0
−1
−2
E_1 − E_0
−5
−4
−3
−2
−1
0
1e−5
E_0

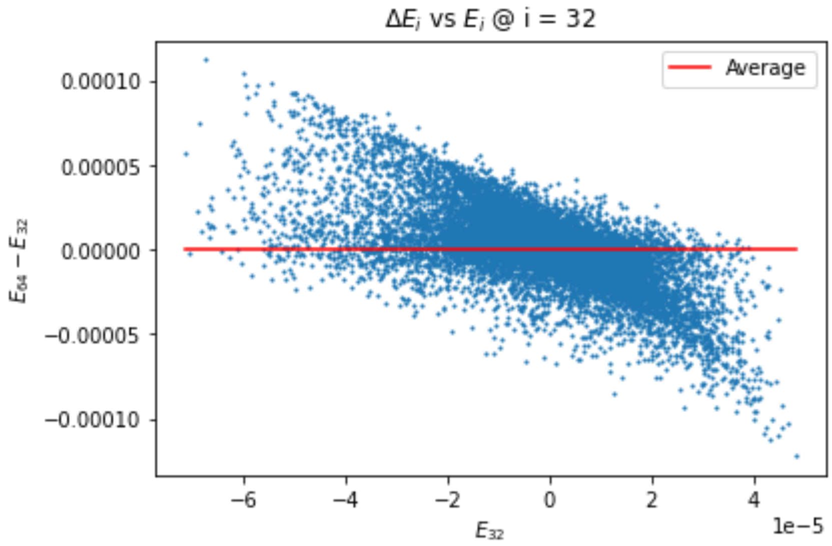

ΔE_i vs E_i @ i = 1

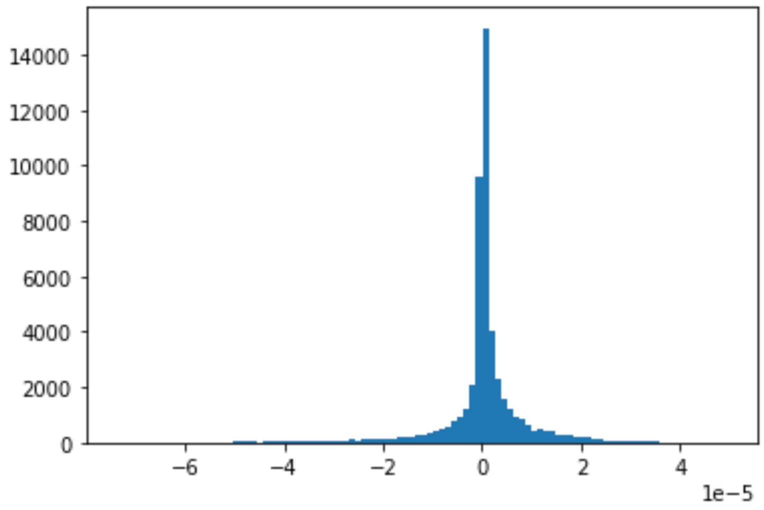

ΔE_i vs E_i @ i = 2

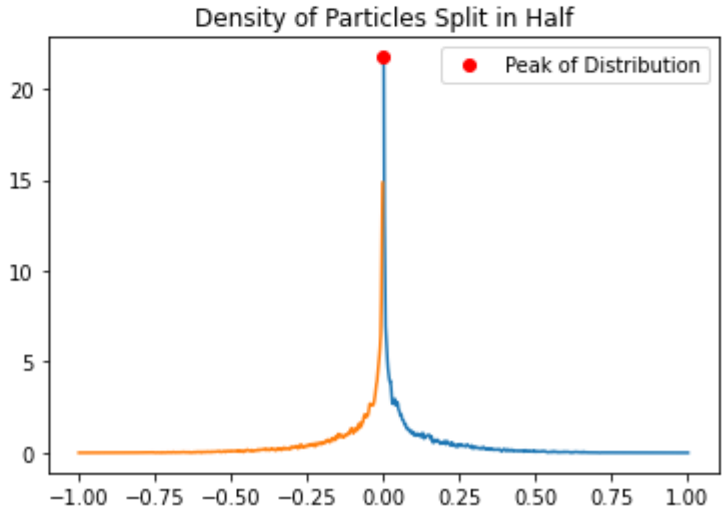

ΔE_i vs E_i @ i = 4

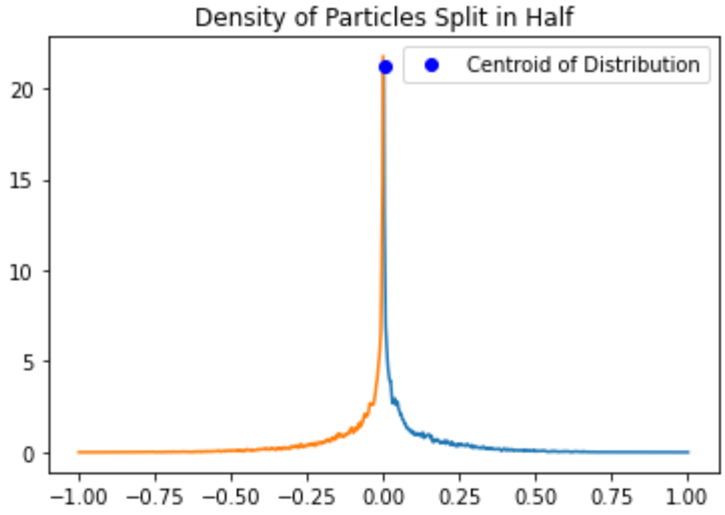

ΔE_i vs E_i @ i = 8
Average
0.00010
0.00005
0.00000
−0.00005
−0.00010
$E_{16} - E_8$
−6
−4
−2
0
2
4
1e−5
E_8

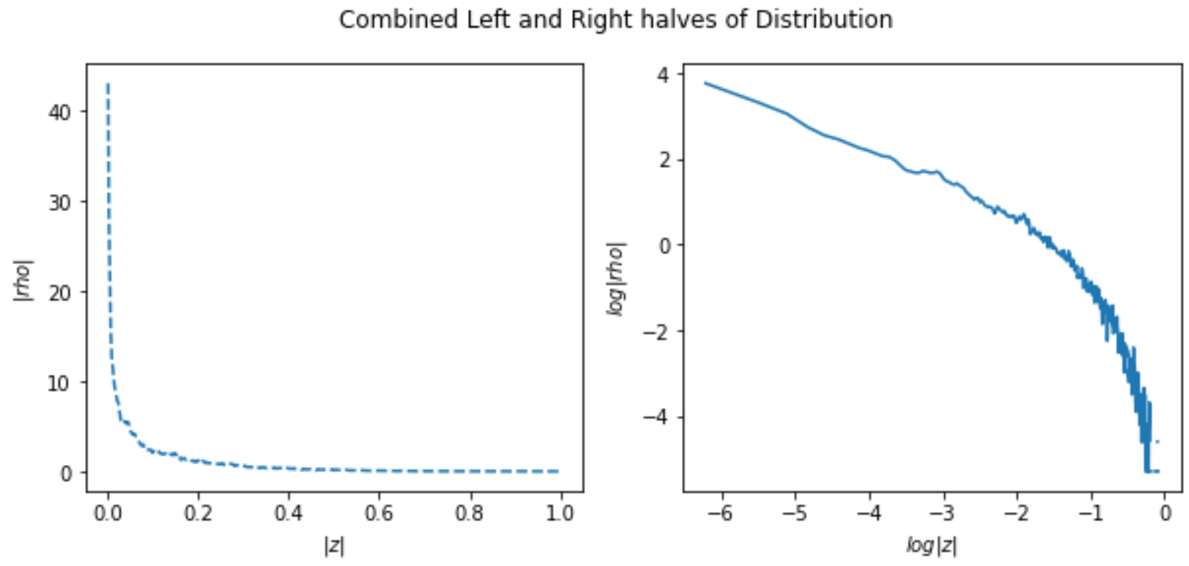

ΔE_i vs E_i @ i = 16
Average
0.00010
0.00005
0.00000
−0.00005
−0.00010
$E_{32} - E_{16}$
−6
−4
−2
0
2
4
1e−5
E_{16}


ΔE_i vs E_i @ i = 32
Average
0.00010
0.00005
0.00000
−0.00005
−0.00010
$E_{64} - E_{32}$
−6
−4
−2
0
2
4
1e−5
E_{32}

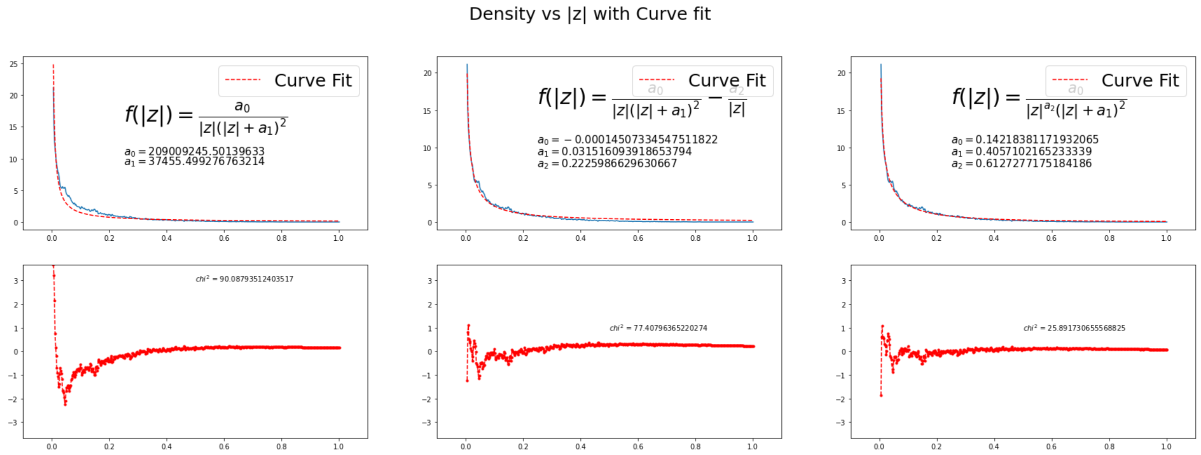

14000
12000
10000
8000
6000
4000
2000
0
−6
−4
−2
0
2
4
1e−5

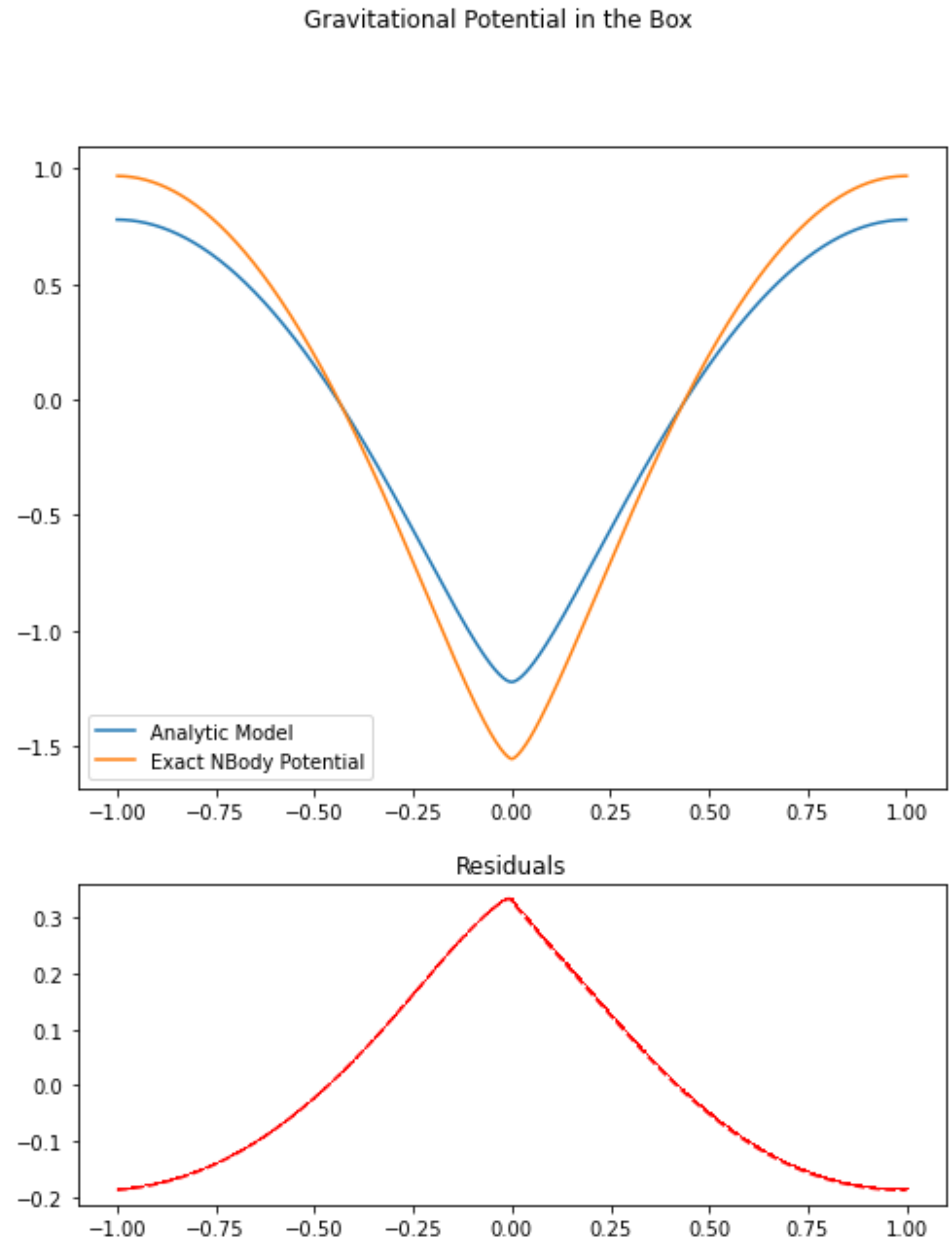

Density of Particles Split in Half
● Peak of Distribution
20
15
10
5
0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


Density of Particles Split in Half
● Centroid of Distribution
20
15
10
5
0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

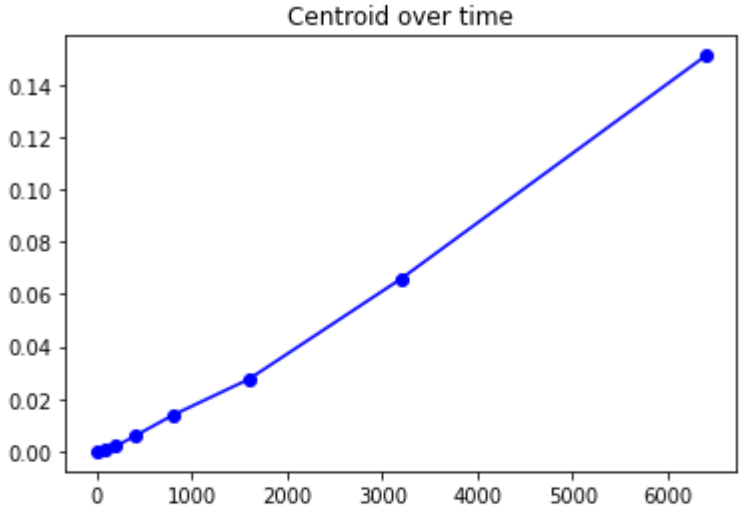


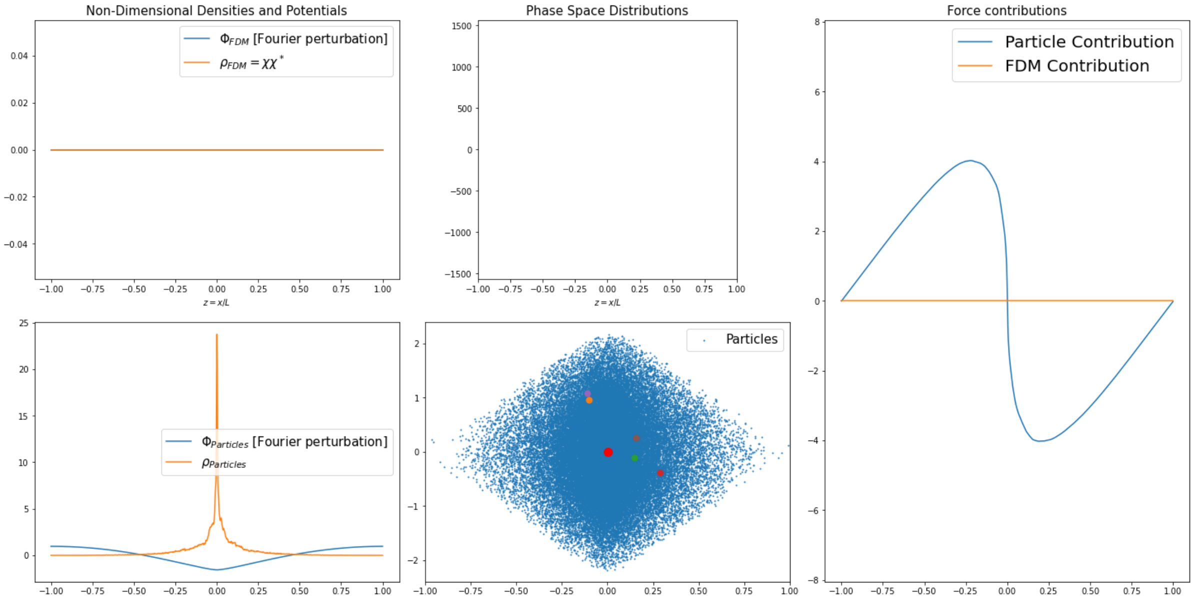
```
Check
Check
Check
```



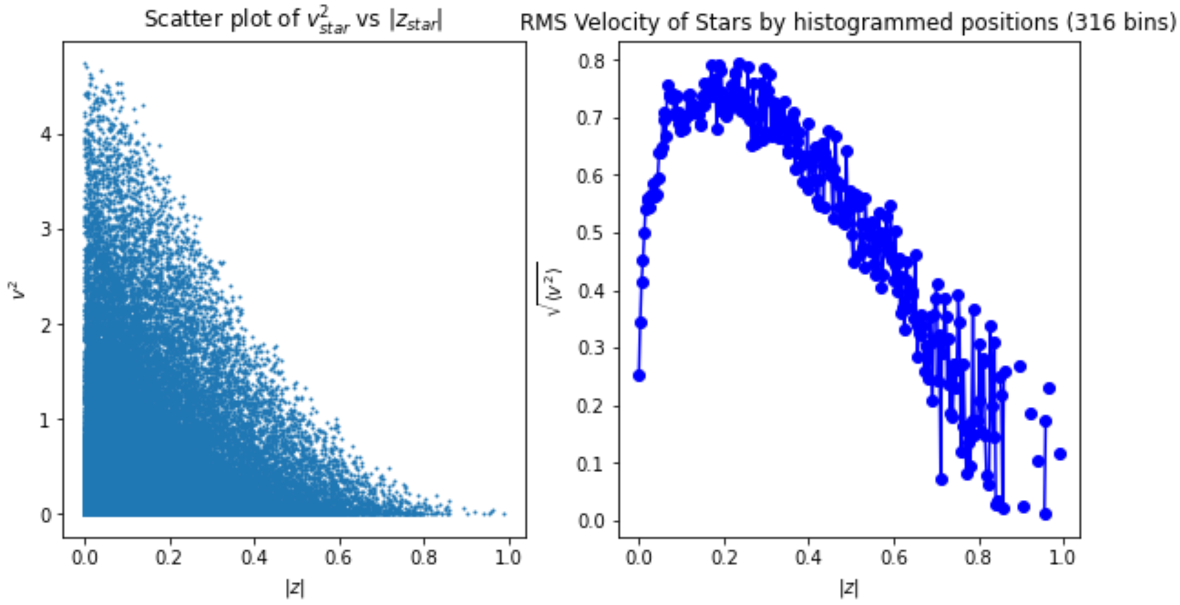


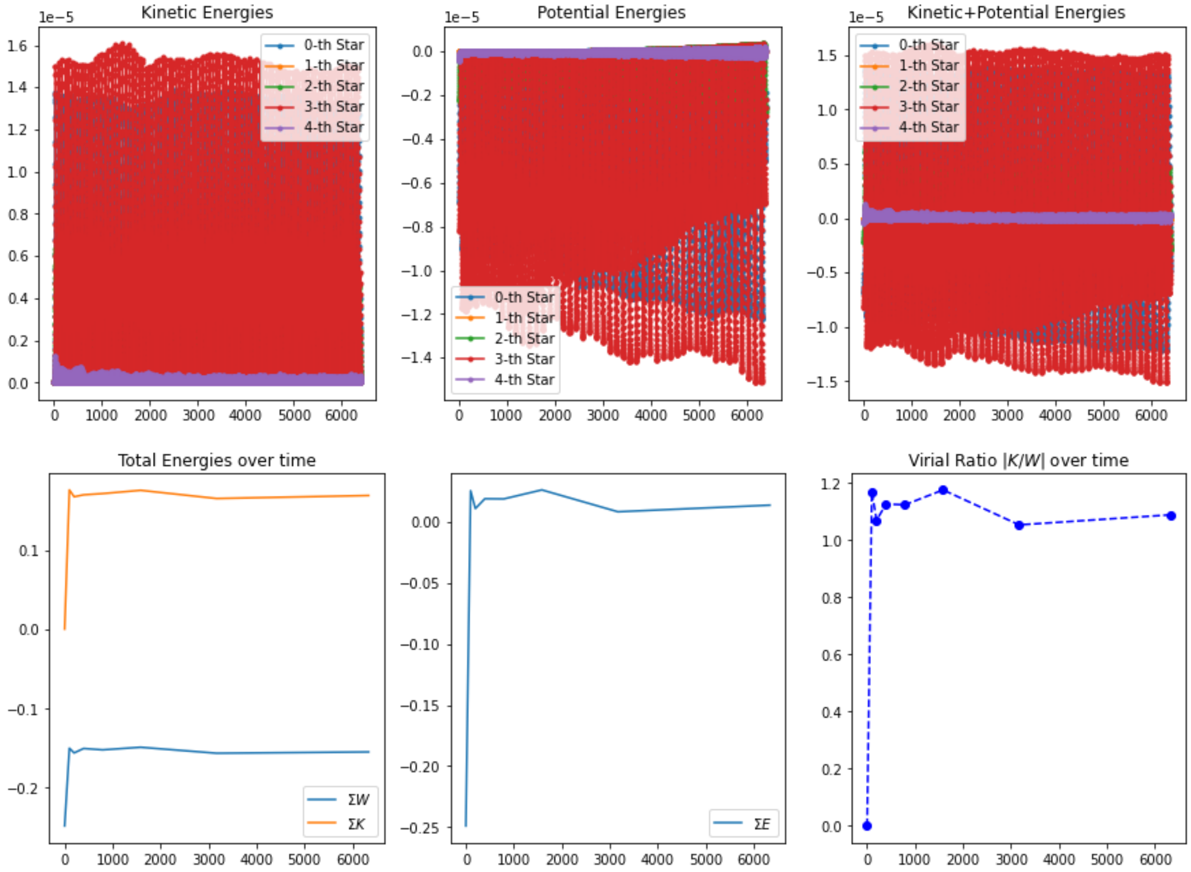
```
----------New Analysis--------
r = 1 mu = 1 Num_bosons = 0 sigma = 1e-05 Num_stars = 100000
```

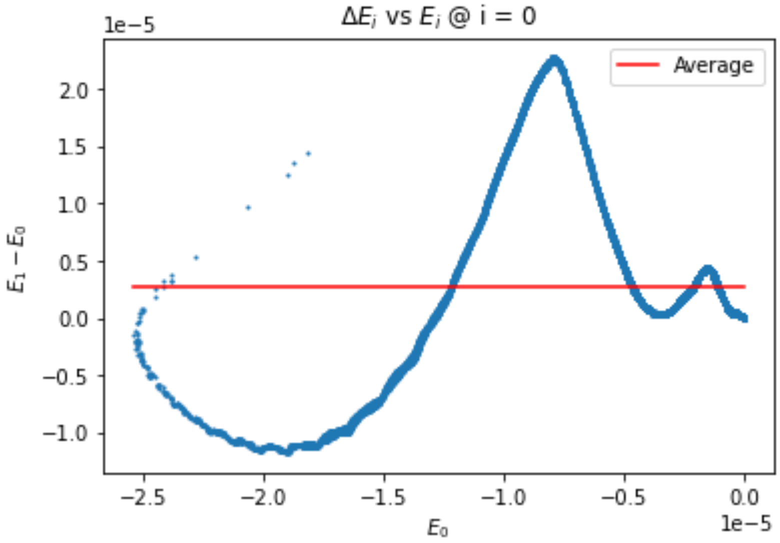


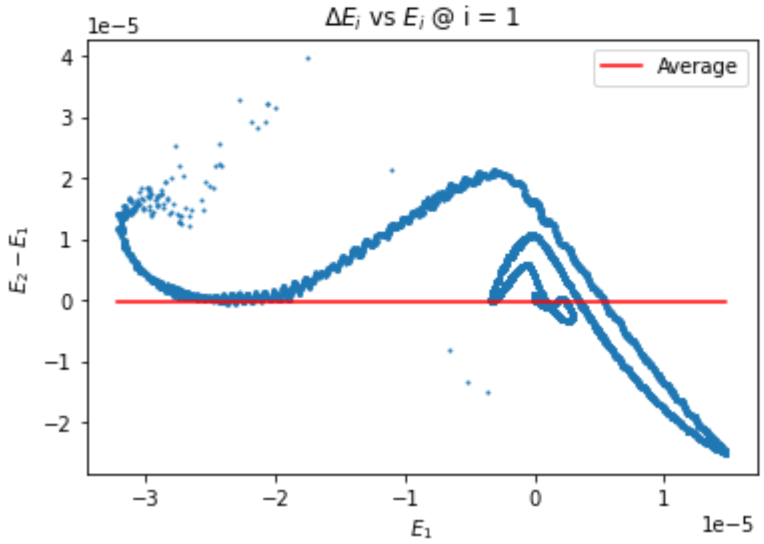


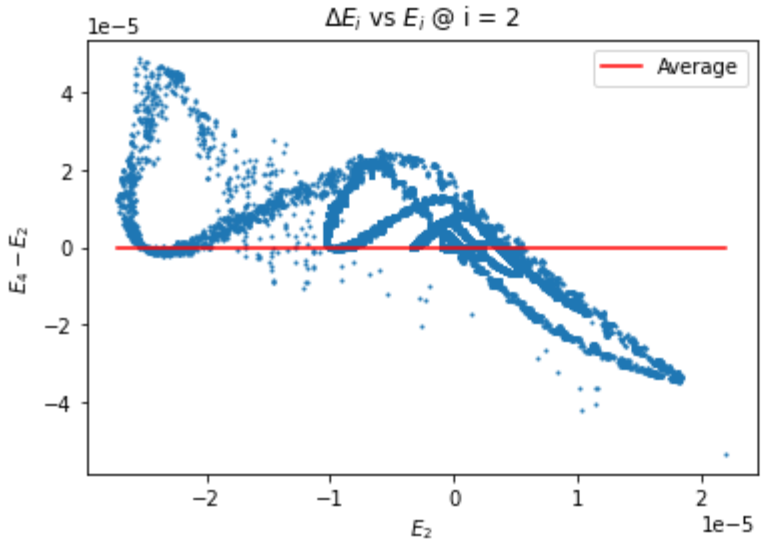
```
v_rms = 0.5840606962146646
z_rms = 0.1588461256252678
K_avg = 0.5*m*v_rms^2 = 0.17056344843137938 (m=1)
=> 2*K_avg = 0.34112689686275877
W_avg = 15884.612562526778
------------------
K_tot = 0.17056344843137822
K_avg = 1.7056344843137823e-06
W_tot = -0.1534179521919812
W_avg = -1.5341795219198121e-06
```

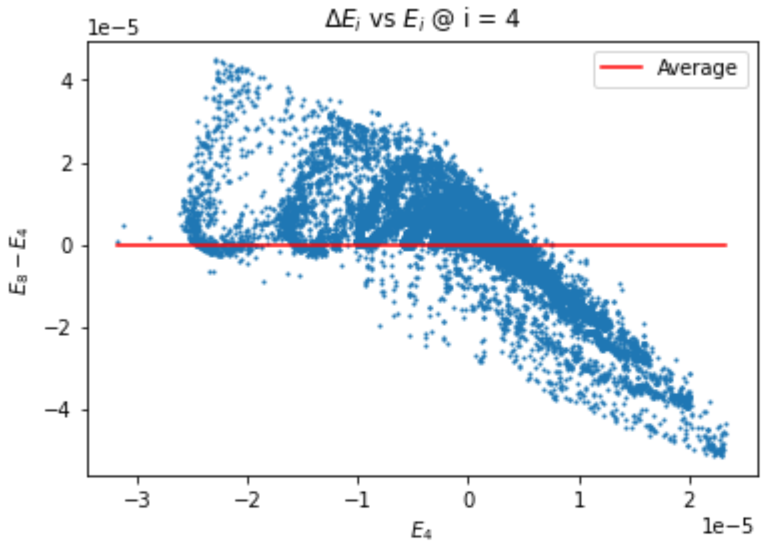


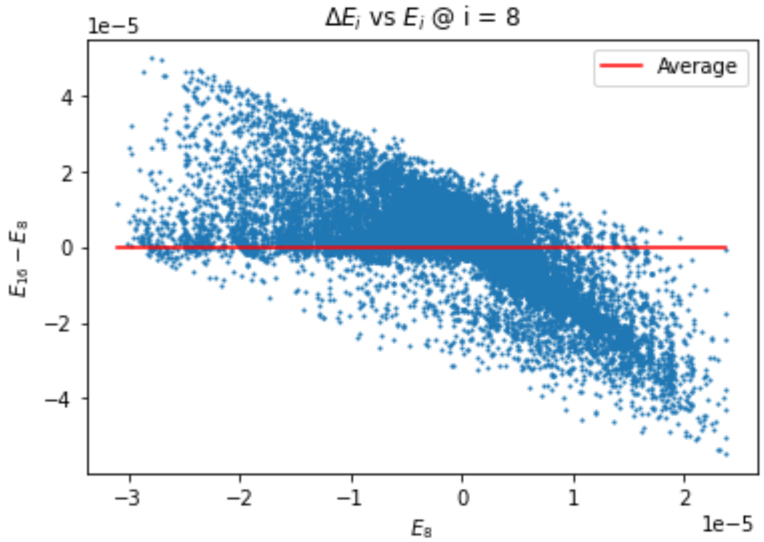

Kinetic Energies
1e−5
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0-th Star
1-th Star
2-th Star
3-th Star
4-th Star
0 1000 2000 3000 4000 5000 6000
Potential Energies
1e−5
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
0-th Star
1-th Star
2-th Star
3-th Star
4-th Star
0 1000 2000 3000 4000 5000 6000
Kinetic+Potential Energies
1e−5
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0-th Star
1-th Star
2-th Star
3-th Star
4-th Star
0 1000 2000 3000 4000 5000 6000
Total Energies over time
0.1
0.0
−0.1
−0.2
ΣW
ΣK
0 1000 2000 3000 4000 5000 6000
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
ΣE
0 1000 2000 3000 4000 5000 6000
Virial Ratio |K/W| over time
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0 1000 2000 3000 4000 5000 6000

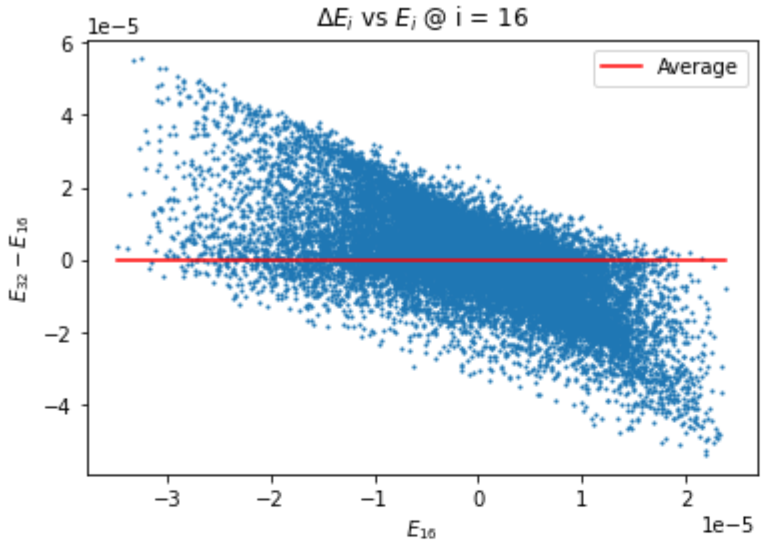

ΔE_i vs E_i @ i = 0
1e−5
Average
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
E_1 − E_0
−2.5 −2.0 −1.5 −1.0 −0.5 0.0
1e−5
E_0

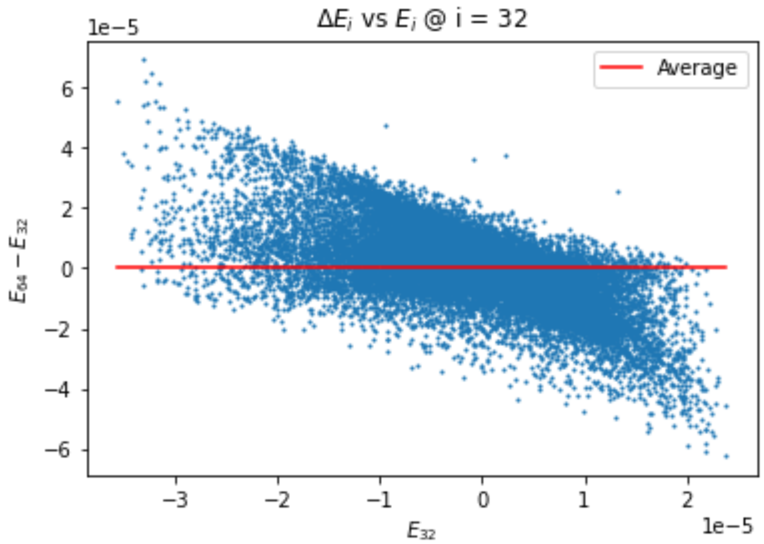

ΔE_i vs E_i @ i = 1
1e−5
4
3
2
1
0
−1
−2
$E_2 - E_1$
Average
−3
−2
−1
0
1
E_1
1e−5

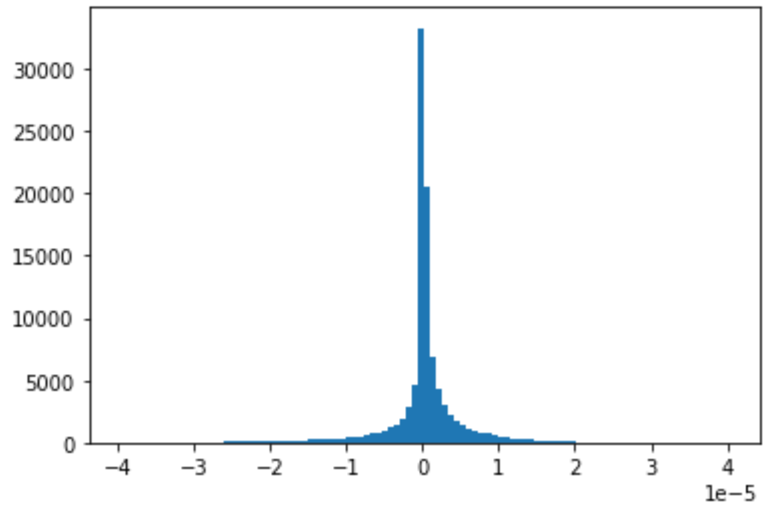

ΔE_i vs E_i @ i = 2
1e−5
4
2
0
−2
−4
$E_4 - E_2$
Average
−2
−1
0
1
2
E_2
1e−5

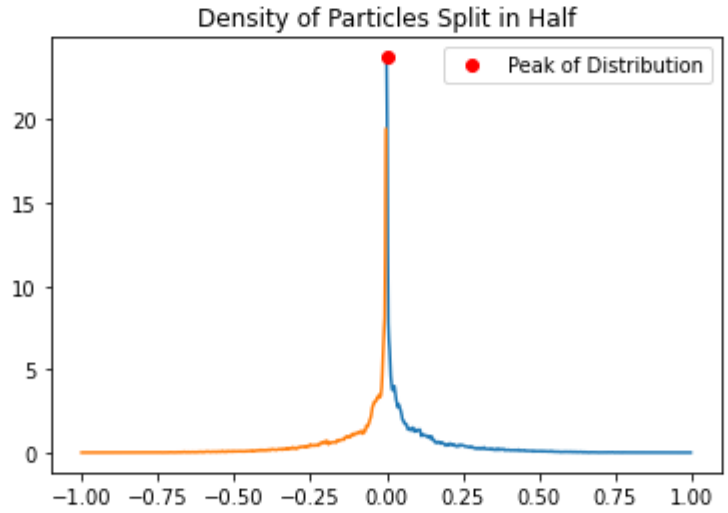

ΔE_i vs E_i @ i = 4
1e−5
4
2
0
−2
−4
$E_8 - E_4$
Average
−3
−2
−1
0
1
2
E_4
1e−5

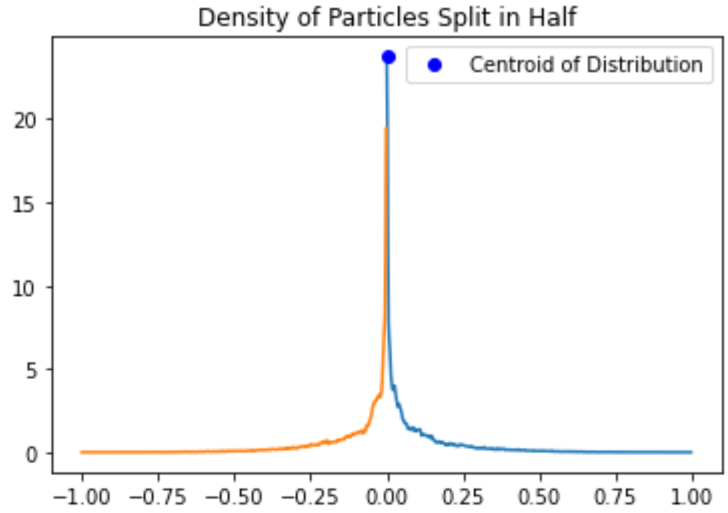

ΔE_i vs E_i @ i = 8

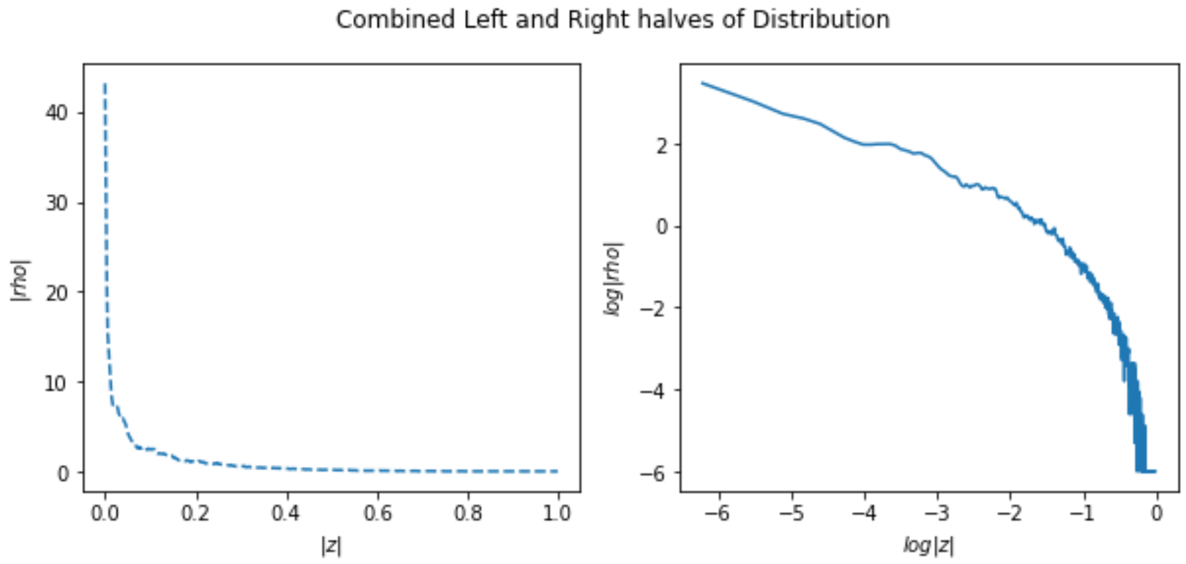

ΔE_i vs E_i @ i = 16


ΔE_i vs E_i @ i = 32

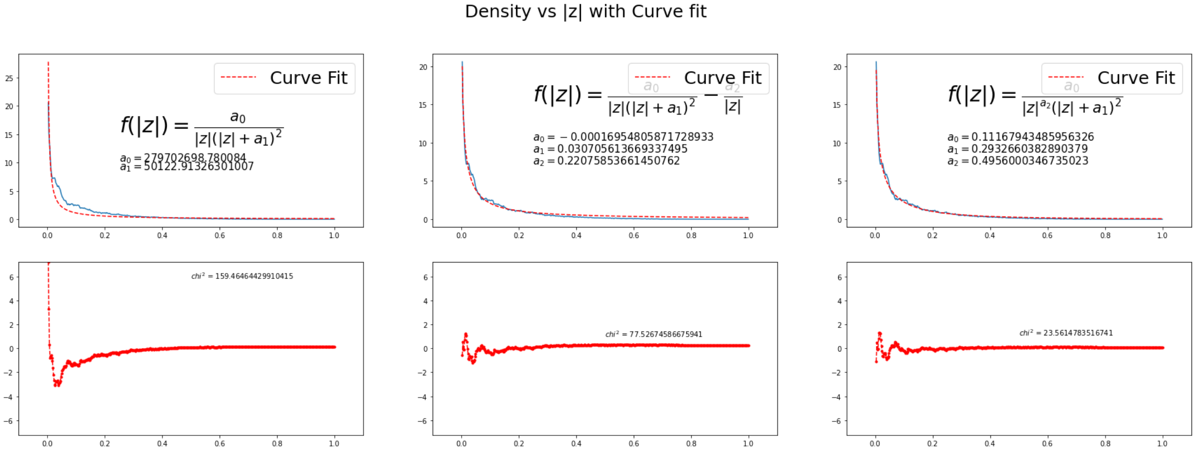

30000
25000
20000
15000
10000
5000
0
−4
−3
−2
−1
0
1
2
3
4
1e−5

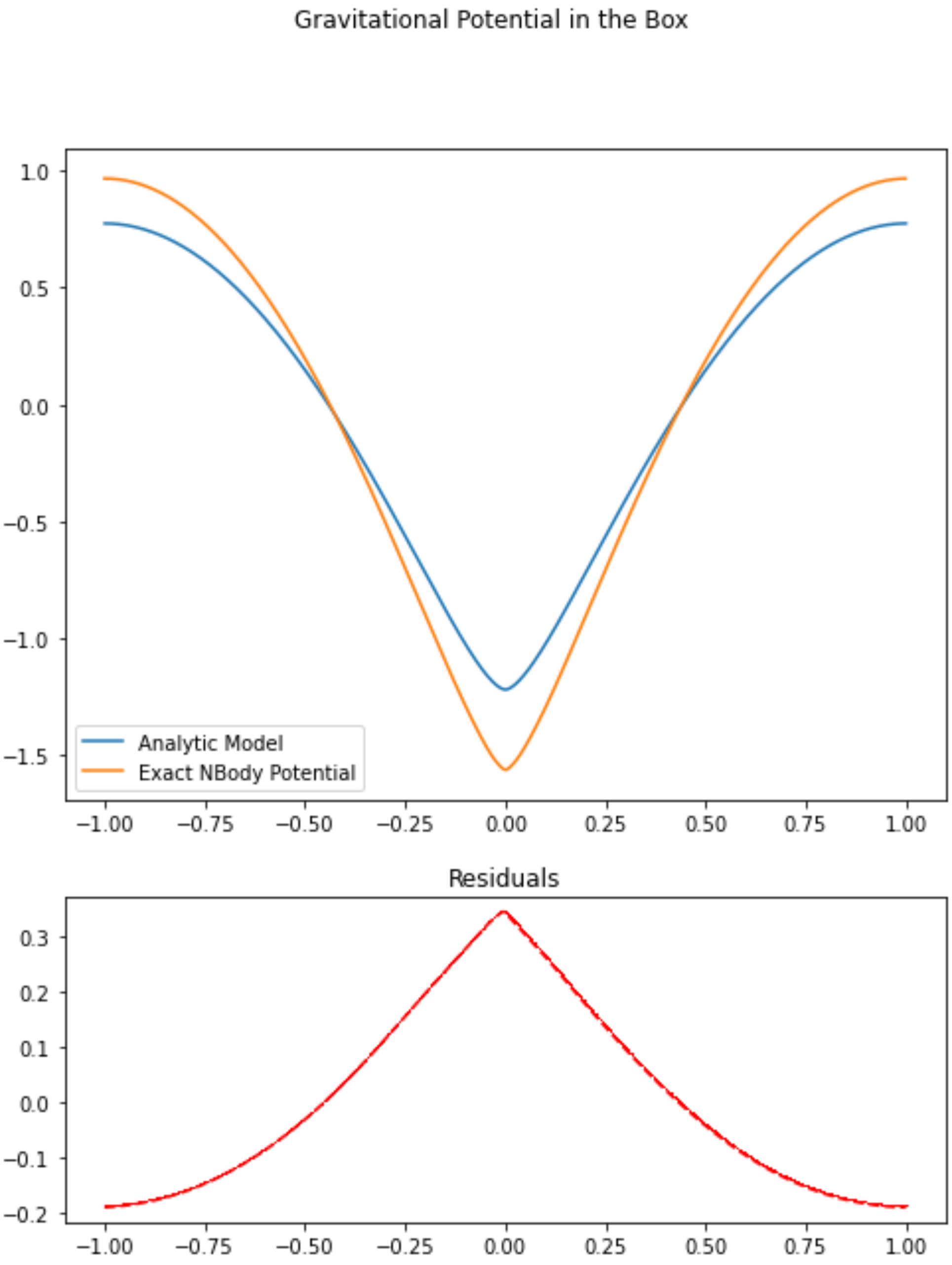

Density of Particles Split in Half
● Peak of Distribution
20
15
10
5
0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


Density of Particles Split in Half
● Centroid of Distribution
20
15
10
5
0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

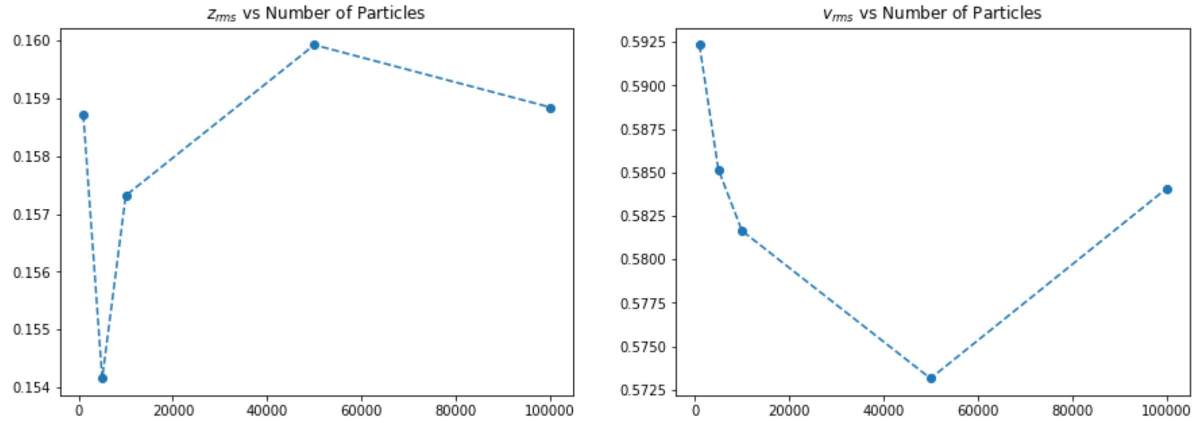


```
Check
Check
Check
```



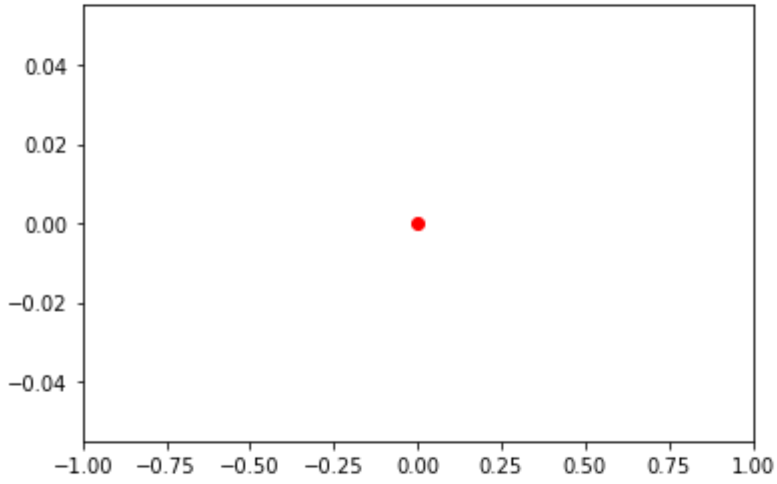


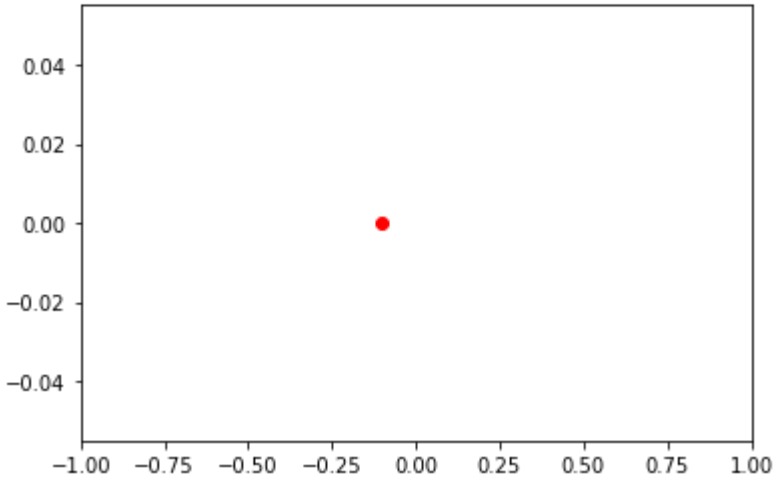
```
In [ ]: import matplotlib.pyplot as plt
        v_rms_s = [0.5923277051957598,0.5851737606787921,0.581649317774426,0.5731593
        z_rms_s  = [0.15871675883738892,0.1541549084183391,0.15731951911022882,0.159
        Num_p_s = [1000,5000,10000,50000,100000]

        fig,ax=plt.subplots(1,2,figsize= (15,5))
        ax[0].plot(Num_p_s,z_rms_s,'o--')
        ax[0].set_title("$z_{rms}$ vs Number of Particles")
        #ax[0].set_ylim(0,0.5)

        ax[1].plot(Num_p_s,v_rms_s,'o--')
        ax[1].set_title("$v_{rms}$ vs Number of Particles")

        plt.show()
```

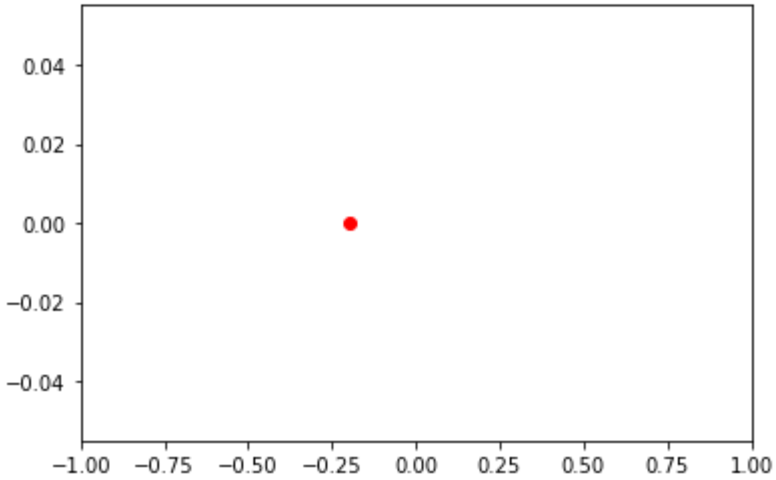


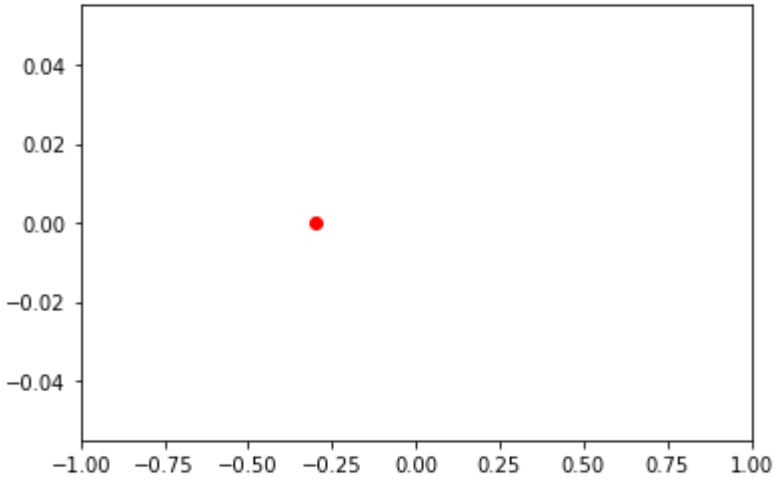
In [ ]:

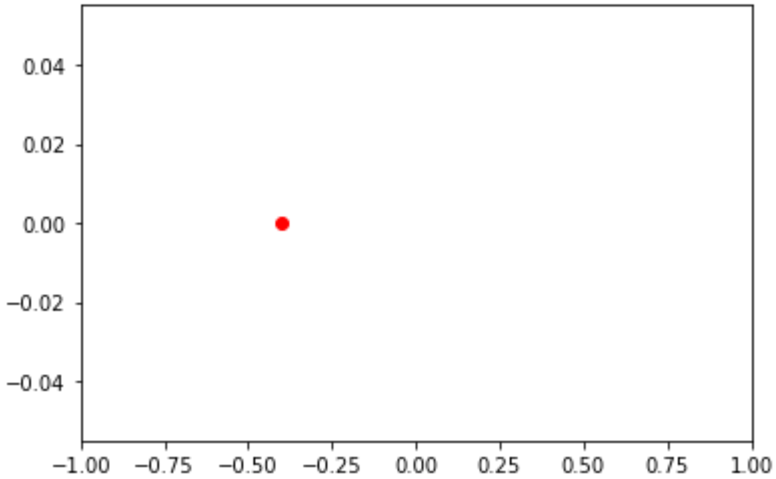
```
import numpy as np
import matplotlib.pyplot as plt

My_Package_PATH = "/home/boris/Documents/Research/Coding"
import sys
sys.path.insert(1, My_Package_PATH)
import OneD.NBody as NB

z = np.linspace(-1,1)

x = 0
v = 1
star = NB.star(0,1,x,v)

dt = 0.1
t = 0
i = 0
while t < 2:
    plt.plot(star.x,0,'ro')
    plt.xlim(-1,1)
    plt.show()

    star.x -= v*dt
    star.reposition(2)
    t += dt
```

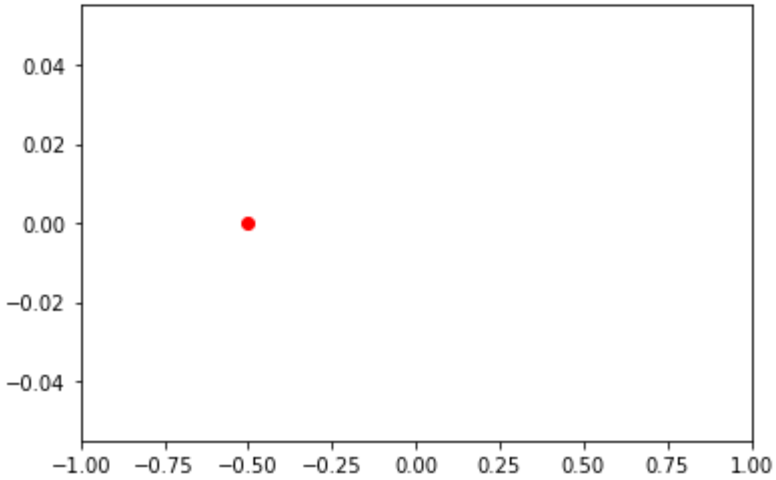


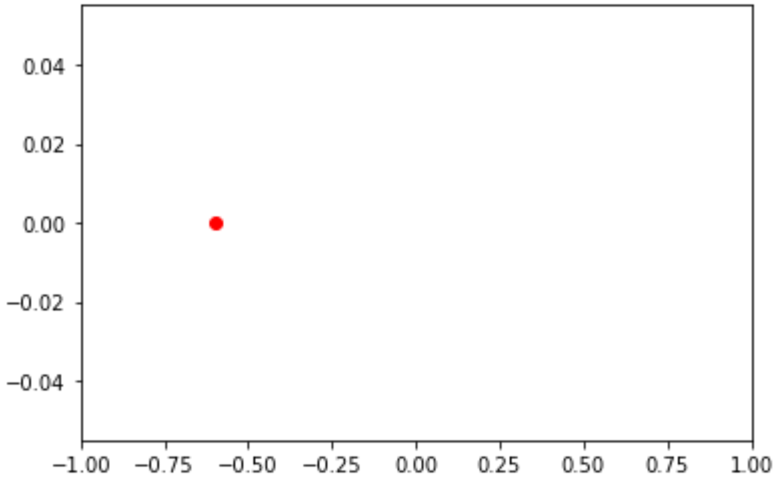

0.04
0.02
0.00
−0.02
−0.04
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

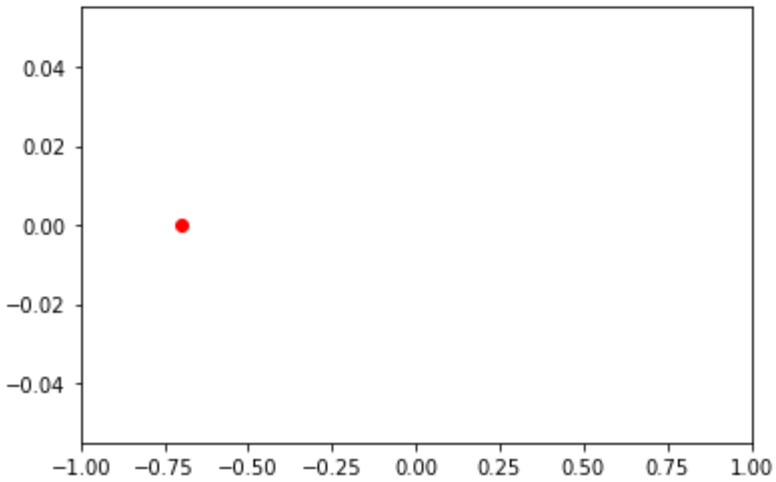

0.04
0.02
0.00
−0.02
−0.04
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

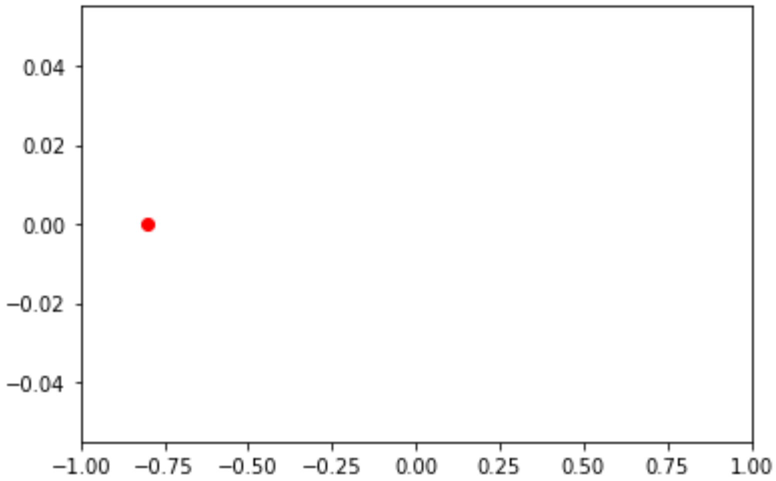

0.04
0.02
0.00
−0.02
−0.04
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

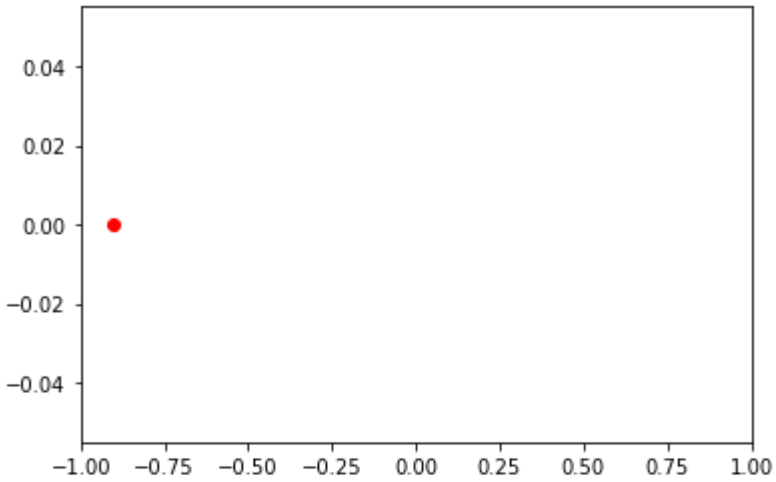

0.04
0.02
0.00
−0.02
−0.04
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00

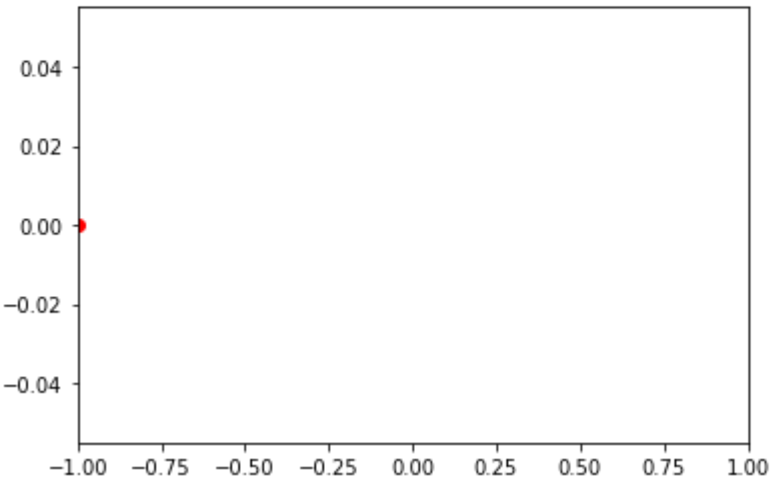

0.04
0.02
0.00
−0.02
−0.04
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00

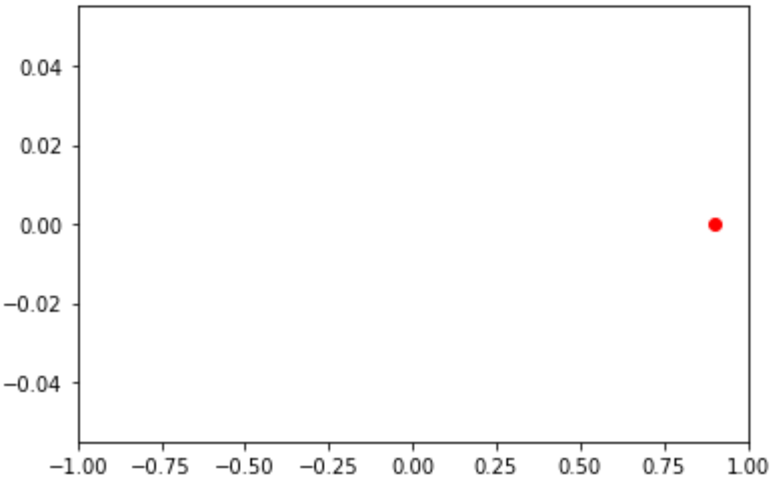

0.04
0.02
0.00
−0.02
−0.04
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00

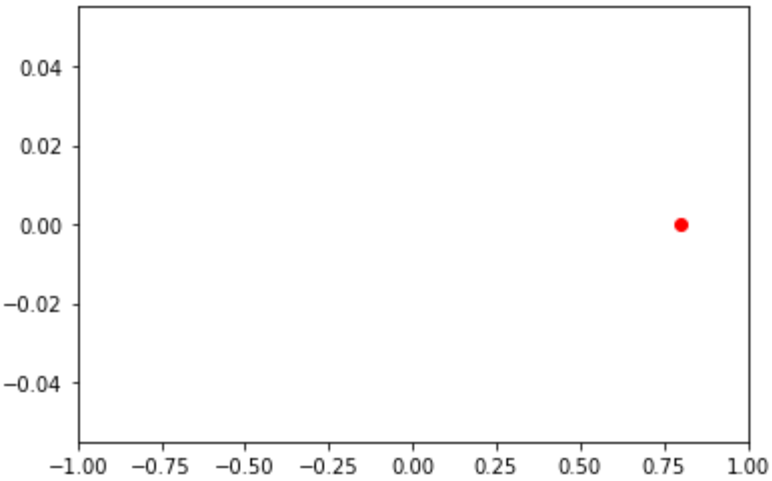

0.04
0.02
0.00
−0.02
−0.04
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

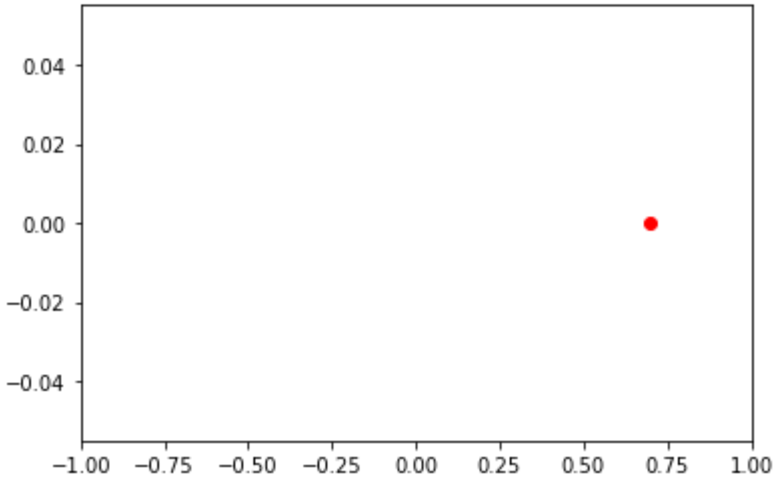

0.04
0.02
0.00
−0.02
−0.04
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

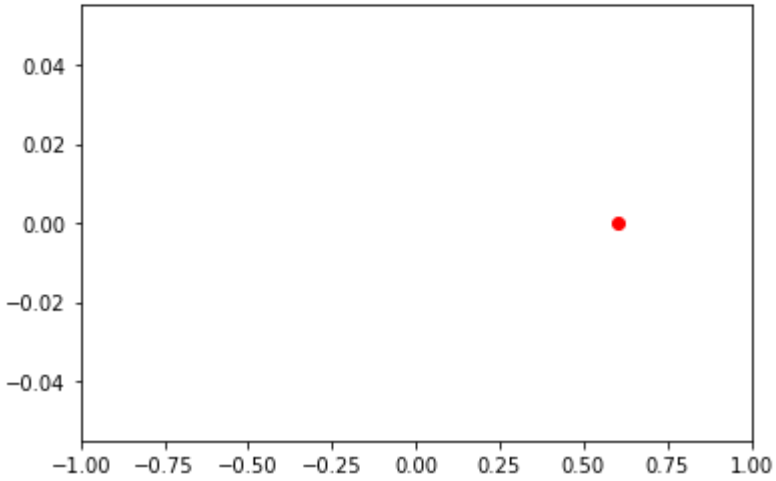

0.04
0.02
0.00
−0.02
−0.04
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

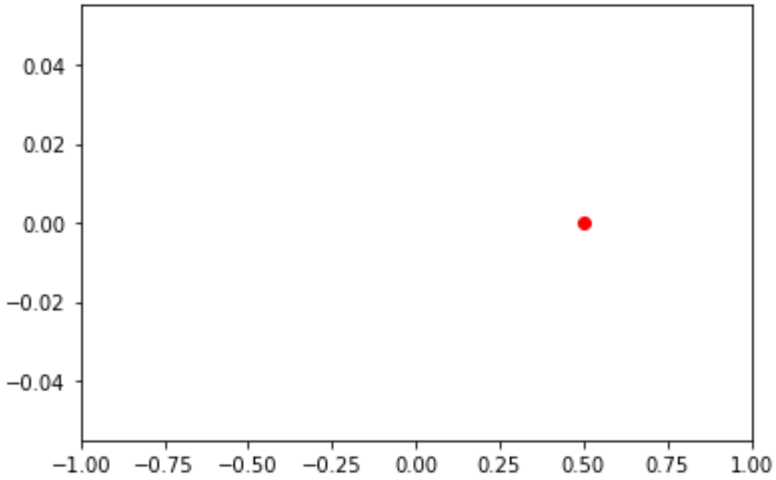

0.04
0.02
0.00
−0.02
−0.04
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

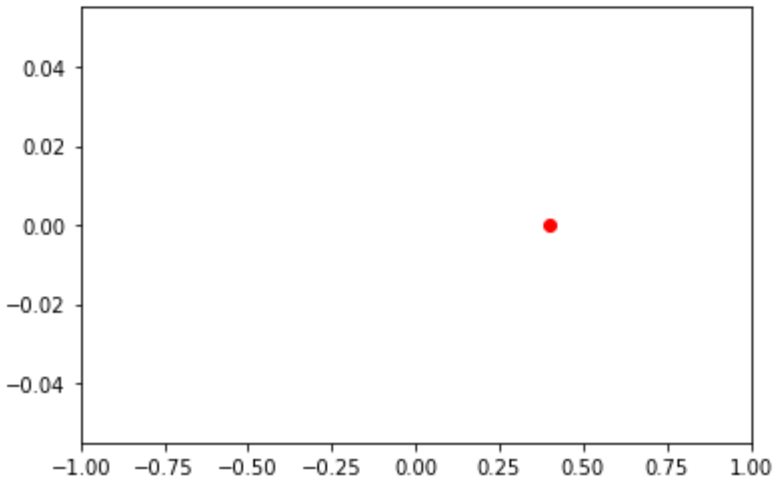

0.04
0.02
0.00
−0.02
−0.04
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

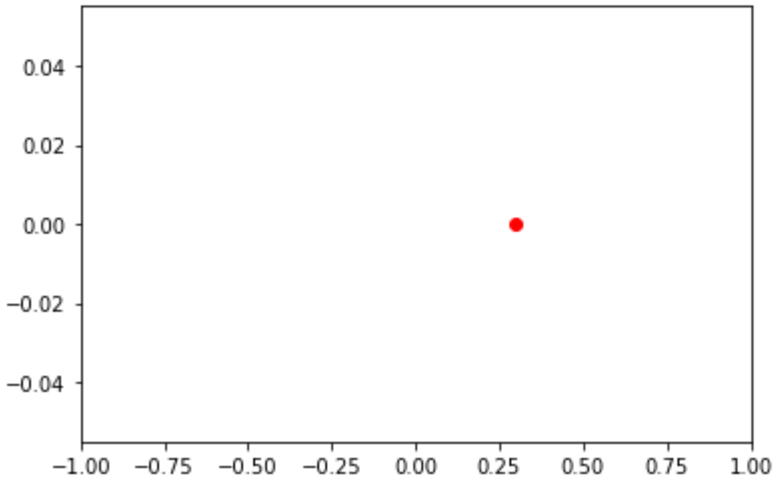

0.04
0.02
0.00
−0.02
−0.04
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

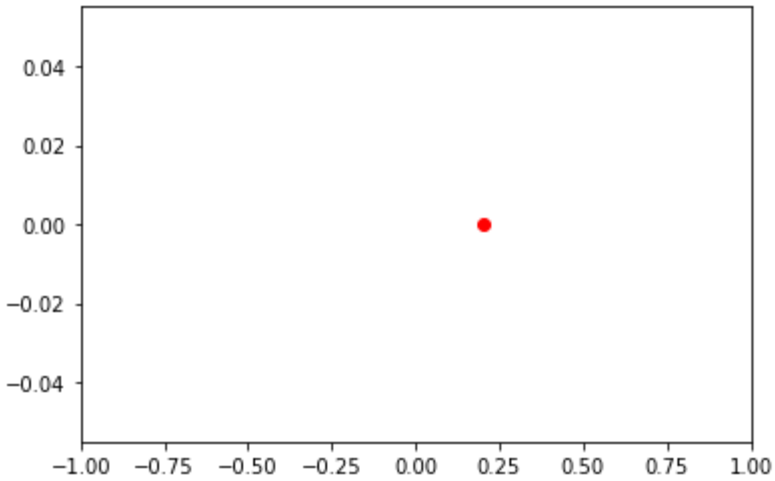

0.04
0.02
0.00
−0.02
−0.04
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

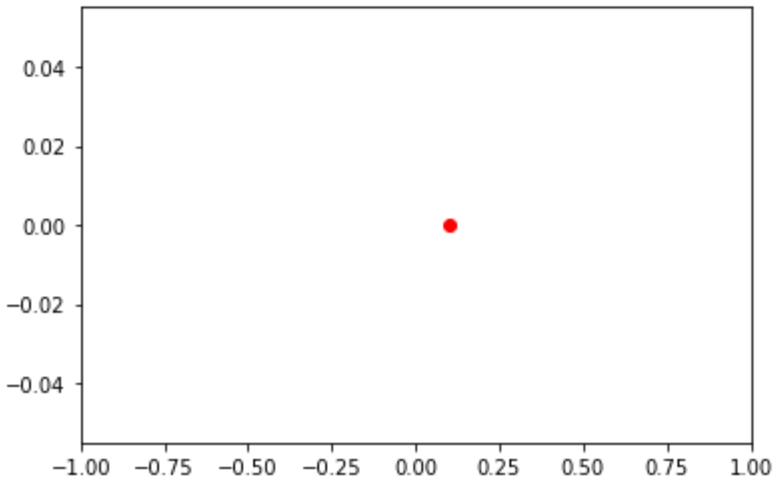

0.04
0.02
0.00
−0.02
−0.04
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


0.04
0.02
0.00
−0.02
−0.04
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


0.04
0.02
0.00
−0.02
−0.04
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


0.04
0.02
0.00
−0.02
−0.04
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


0.04
0.02
0.00
−0.02
−0.04
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


0.04
0.02
0.00
−0.02
−0.04
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm
from matplotlib.colors import LogNorm, Normalize
import os
import subprocess
import cv2
from PIL import Image
import scipy.optimize as opt

#Import My Library
My_Package_PATH = "/home/boris/Documents/Research/Coding"
import sys
sys.path.insert(1, My_Package_PATH)
import OneD.WaveNonDim as ND
import OneD.NBody as NB
import OneD.GlobalFuncs as GF

#Set up Directory for saving files/images/videos
# Will not rename this again
dirExtension = "1D_Codes/Non-Dim/Analysis"
Directory = os.getcwd()#+"/"+dirExtension #os.curdir() #"/home/boris/Documer
print(Directory)

r,m,Num_bosons,sigma,Num_stars = [0.5,1.0,0,1,10000]

mu = m #M_scale = 1

L = 2
N = 10**3
z = np.linspace(-L/2,L/2,N)
dz = z[1]-z[0]

folder = "ParticlesOnly_Snapshots"
stars_x = np.loadtxt(folder+"/"+f"StarsOnly_Pos.csv", dtype = float, delimit
stars_v = np.loadtxt(folder+"/"+f"StarsOnly_Vel.csv", dtype = float, delimit
Energies = np.loadtxt(folder+"/"+"Energies.csv", dtype = float,delimiter = "
#chi = np.loadtxt(folder+"/"+f"Chi.csv", dtype = complex, delimiter=",")
chi = np.zeros_like(z)
centroids = np.loadtxt(folder+"/"+"Centroids.csv",dtype = float, delimiter='

stars = [NB.star(i,sigma,stars_x[i],stars_v[i]) for i in range(len(stars_x))

grid_counts = NB.grid_count(stars,L,z)

rho = (grid_counts/dz)*sigma


i = 0
max_bool = False
while max_bool == False:
    for j in range(len(rho)):
        if rho[j] > rho[i]: #if you come across an index j that points to a
            #then set i equal to j
            i = j
            #break
        else:
            max_index = i
            max_bool = True
            max_bool = True
max_rho = rho[max_index]

#Other method to accumulate left and right sides:
for star in stars:
    star.x = star.x - z[max_index] #shift
    star.reposition(L) #reposition

grid_counts = NB.grid_count(stars,L,z)
rho_part = (grid_counts/dz)*sigma
#Add the density from the FDM
rho_FDM = mu*np.absolute(chi)**2
rho = rho_FDM + rho_part

centroid_z = 0
for j in range(len(grid_counts)):
    centroid_z += z[j]*grid_counts[j]
centroid_z = centroid_z / Num_stars

stars_x = [star.x for star in stars]

std = np.std(stars_v)
mean_x = np.mean(stars_x)


R = 0
while True:
    R += dz
    mass_enclosed = 0
    star_collection = []
    for star in stars:
        if np.abs(star.x-mean_x) <= R:
            mass_enclosed += 1
            star_collection.append(star)
    print(R,mass_enclosed)
    if mass_enclosed >= 0.5*Num_stars:
        break

print(R)
plt.figure()
plt.scatter(stars_x,stars_v,s=1)
xx = np.linspace(-R,R,100)
plt.plot(xx,np.sqrt(R-xx**2))
plt.plot(xx,-np.sqrt(R-xx**2))
plt.scatter([star.x for star in star_collection],[star.v for star in star_co
plt.show()

Sigma = std**2 / R
print(Sigma)
```

```
/home/boris/Documents/Research/Coding/1D_Codes/Non-Dim/Analysis
```

```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
/home/boris/Documents/Research/Coding/1D_Codes/Non-Dim/Analysis/CurveFittin
g.ipynb Cell 5' in <cell line: 37>()
     <a href='vscode-notebook-cell:/home/boris/Documents/Research/Coding/1D
_Codes/Non-Dim/Analysis/CurveFitting.ipynb#ch0000004?line=34'>35</a> stars_
x = np.loadtxt(folder+"/"+f"StarsOnly_Pos.csv", dtype = float, delimiter
=",")
     <a href='vscode-notebook-cell:/home/boris/Documents/Research/Coding/1D
_Codes/Non-Dim/Analysis/CurveFitting.ipynb#ch0000004?line=35'>36</a> stars_
v = np.loadtxt(folder+"/"+f"StarsOnly_Vel.csv", dtype = float, delimiter
=",")
---> <a href='vscode-notebook-cell:/home/boris/Documents/Research/Coding/1D
_Codes/Non-Dim/Analysis/CurveFitting.ipynb#ch0000004?line=36'>37</a> Energi
es = np.loadtxt(folder+"/"+"Energies.csv", dtype = float,delimiter = ",")
     <a href='vscode-notebook-cell:/home/boris/Documents/Research/Coding/1D
_Codes/Non-Dim/Analysis/CurveFitting.ipynb#ch0000004?line=37'>38</a> #chi =
np.loadtxt(folder+"/"+f"Chi.csv", dtype = complex, delimiter=",")
     <a href='vscode-notebook-cell:/home/boris/Documents/Research/Coding/1D
_Codes/Non-Dim/Analysis/CurveFitting.ipynb#ch0000004?line=38'>39</a> chi =
np.zeros_like(z)

File /usr/lib/python3/dist-packages/numpy/lib/npyio.py:1067, in loadtxt(fna
me, dtype, comments, delimiter, converters, skiprows, usecols, unpack, ndmi
n, encoding, max_rows, like)
   1065     fname = os_fspath(fname)
   1066 if _is_string_like(fname):
-> 1067     fh = np.lib._datasource.open(fname, 'rt', encoding=encoding)
   1068     fencoding = getattr(fh, 'encoding', 'latin1')
   1069     fh = iter(fh)

File /usr/lib/python3/dist-packages/numpy/lib/_datasource.py:193, in open(p
ath, mode, destpath, encoding, newline)
    156 """
    157 Open `path` with `mode` and return the file object.
    158 
   (...)
    189 
    190 """
    192 ds = DataSource(destpath)
--> 193 return ds.open(path, mode, encoding=encoding, newline=newline)

File /usr/lib/python3/dist-packages/numpy/lib/_datasource.py:533, in DataSo
urce.open(self, path, mode, encoding, newline)
    530     return _file_openers[ext](found, mode=mode,
    531                               encoding=encoding, newline=newline)
    532 else:
--> 533     raise IOError("%s not found." % path)

OSError: ParticlesOnly_Snapshots/Energies.csv not found.
```

In [ ]:

```
G = 6.67E-11
print(R)
print("--------------------")
print("")

Sigma = std**2 / (np.pi* R**(3/2))
print(Sigma)

print(10000/R)
print(std**2)
print(10000/(np.pi*R**2))

print(std**2 * R)

print("--------------------")
print("")
new_std = np.std([star.v for star in star_collection])
Sigma = new_std**2 / (np.pi* R**(3/2))
print(f"Sigma = {Sigma}" )

print(10000/R)
print(new_std**2)
print(10000/(np.pi*R**2))

print(new_std**2 * R)
```

```
0.048048048048046965
--------------------

100502.33917739333
208125.0000000047
3325.3676414862753
1378791.600352967
159.77742421555317
--------------------

Sigma = 61907.311466298816
208125.0000000047
2048.3560084912815
1378791.600352967
98.41950791549479
```

In [ ]:

```
v_rms = np.sqrt(np.mean([star.v**2 for star in stars]))
print(v_rms)
```

```
57.66620191650019
```

In [ ]:

```
v_mean = np.mean([star.v for star in stars])
std = np.sqrt(np.sum([(star.v - v_mean)**2 for star in stars])/(len(stars)-1
print(std)

Sigma = std**2 / (np.pi* R**(3/2))
print(Sigma)

print(10000/R)
print(std**2)
print(10000/(np.pi*R**2))

print(std**2 * R)
```

```
57.66888425752163
100512.39041643498
208125.0000000047
3325.7002115074265
1378791.600352967
159.79340355590878
```

In [ ]:

```
phi_part = GF.fourier_potentialV2(rho_part,L)
phi_part = phi_part - np.mean(phi_part)
print(np.mean(phi_part))

phi_part = phi_part - np.max(phi_part)

# Compute Chandrasekhar's potential energy tensor:
a_part = NB.acceleration(phi_part,L)
W = 0
for i in range(len(z)):
    dW = rho_part[i]*z[i]*a_part[i]
    W += dW
print(W)

a_part = NB.acceleration(phi_part,L)
W = 0
for i in range(len(z)):
    dW = -0.5*rho_part[i]*phi_part[i]
    W += dW
print(W)

# Compute only for the stars that exist:
a_part = NB.acceleration(phi_part,L)
W = 0
for star in stars:
    g = NB.g(star,a_part,dz)

    dW = - star.x*g
    W += dW / Num_stars
print(W)

# phi_part = GF.fourier_potentialV2(rho_part,L)
# a_part = NB.acceleration(phi_part,L)
# W = 0
# for i in range(len(z)):
#     dW = - dz*a_part[i]**2 / (8*np.pi)
#     W += dW
# print(W)
#W = np.sum(phi_part)
#print(W)

# Compute only for the stars that exist:
W = 0
for star in stars:
    #g = NB.g(star,a_part,dz)
    i = int(star.x//dz)
    rem = star.x % dz

    if i != len(phi_part)-1:
        value = phi_part[i] + rem*(phi_part[i+1]-phi_part[i])/dz
    elif i == len(phi_part)-1:
        # then i+1 <=> 0
        value = phi_part[i] + rem*(phi_part[0]-phi_part[i])/dz

    phi_star = value
    dW = phi_star
    W += dW
print(W)
```

```
0.0
-16785655529.943058
55851483242.48459
-1500.1026550328993
-7621006.55368937
```

Compute Total KE and Total Potential Energy of Stars

In [ ]:

```
# Compute total KE of stars:
K = 0
for star in stars:
    dK = 0.5*sigma*star.v**2
    K += dK
print(K)
#average KE:
print(K/Num_stars)

# #Compute Total Potential
# W = 0
# for star in stars:
#     #g = NB.g(star,a_part,dz)
#     i = int(star.x//dz)
#     rem = star.x % dz

#     if i != len(phi_part)-1:
#         value = phi_part[i] + rem*(phi_part[i+1]-phi_part[i])/dz
#     elif i == len(phi_part)-1:
#         # then i+1 <=> 0
#         value = phi_part[i] + rem*(phi_part[0]-phi_part[i])/dz

#     phi_star = value
#     dW = phi_star
#     W += dW
# print(W)
# #average W:
# print(W/Num_stars)

# Compute only for the stars that exist:
a_part = NB.acceleration(phi_part,L)
W = 0
for star in stars:
    g = NB.g(star,a_part,dz)

    dW = - sigma*star.x*g
    W += dW
print(W)
print(W/Num_stars)
```

```
16626954.217372933
1662.6954217372934
-15001026.55032902
-1500.102655032902
```

Calculate v_{rms} and R_{syst}

Want to verify

$$\langle v^2 \rangle = \frac{GM}{R_{syst}}$$

In [ ]:

```
v_rms = np.sqrt(np.mean([star.v**2 for star in stars]))
z_rms = np.sqrt(np.mean([star.x**2 for star in stars]))
print(f"v_rms = {v_rms}")
print(z_rms)
#v_rms = np.sqrt(np.sum([star.v**2 for star in stars])/Num_stars)

K = 0.5 * v_rms**2
print(f"K_avg = 0.5*m*v_rms^2 = {K} (m=1)")
print(F"=> 2*K_avg = {2*K}")

print(z_rms*Num_stars)

print("-----------------------")




R_syst = Num_stars / v_rms**2
print(R_syst)



rho_0 = np.mean(rho_part)
print(4*rho_0*z_rms)

print(v_rms**2 / (2*np.pi*z_rms))

print(16*np.pi*rho_0**2*z_rms**3 / Num_stars)
```

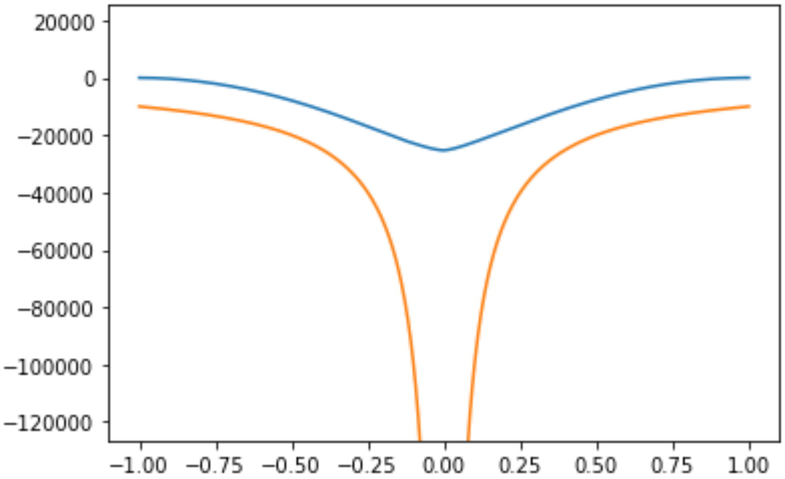
```
v_rms = 57.66620191650019
0.1567842541769929
K_avg = 0.5*m*v_rms^2 = 1662.6954217372852 (m=1)
=> 2*K_avg = 3325.3908434745704
1567.842541769929
-----------------------
3.0071653140030277
3132.549398456389
3375.673107160588
483.3349205244898
```

In [ ]:

```
plt.plot(z,phi_part)
plt.plot(z,-Num_stars/np.abs(z))
plt.ylim(5*np.min(phi_part),-np.min(phi_part))
```

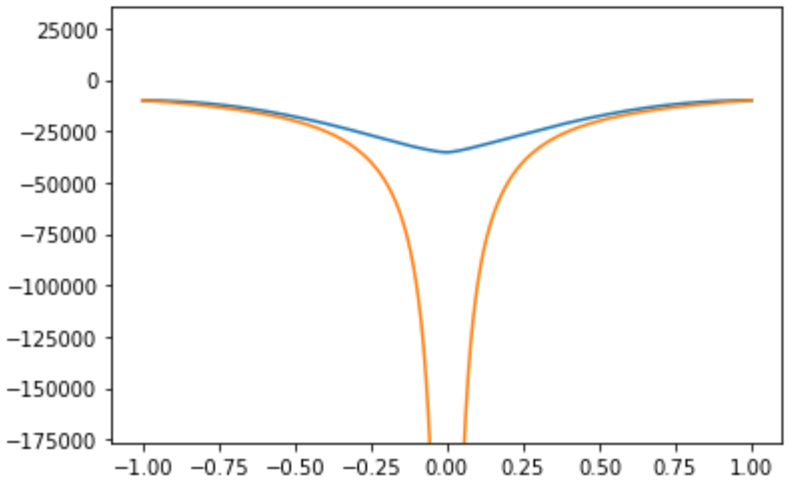
Out[ ]:

```
(-126523.1647523825, 25304.632950476498)
```



In [ ]:

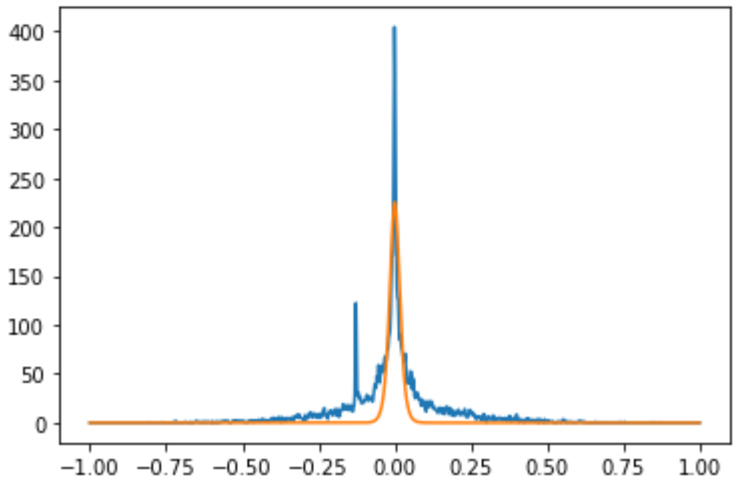
```
phi_part = phi_part - (np.max(phi_part)-np.max(-Num_stars/np.abs(z)))

plt.plot(z,phi_part)
plt.plot(z,-Num_stars/np.abs(z))
plt.ylim(5*np.min(phi_part),-np.min(phi_part))
plt.show()

# Compute total KE of stars:
K = 0
for star in stars:
    dK = 0.5*star.v**2
    K += dK
print(K)
#average KE:
print(K/Num_stars)

#Compute Total Potential
W = 0
for star in stars:
    #g = NB.g(star,a_part,dz)
    i = int(star.x//dz)
    rem = star.x % dz

    if i != len(phi_part)-1:
        value = phi_part[i] + rem*(phi_part[i+1]-phi_part[i])/dz
    elif i == len(phi_part)-1:
        # then i+1 <=> 0
        value = phi_part[i] + rem*(phi_part[0]-phi_part[i])/dz

    phi_star = value
    dW = phi_star
    W += dW
print(W)
#average W:
print(W/Num_stars)
```

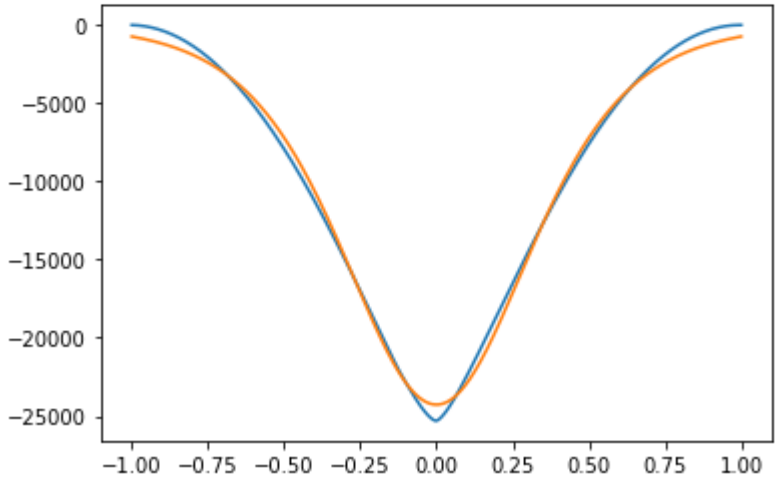


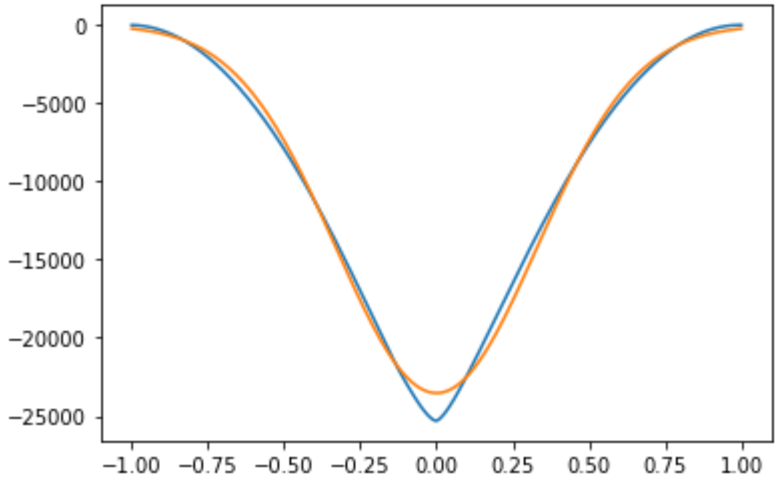
```
16626954.217372933
1662.6954217372934
-107621006.55368945
-10762.100655368946
```

In [ ]:

```
def f(z,*p):
    u_0 = p[0]
    z_0 = p[1]
    return u_0 / np.cosh(0.5*z/z_0)**2

guess = [rho_0,z_0]
popt,pcov = opt.curve_fit(f,z,grid_counts,p0 = guess)
plt.plot(z,grid_counts)
plt.plot(z,f(z,*popt))
plt.show()

guess = [rho_0,z_0]
popt,pcov = opt.curve_fit(f,z,phi_part,p0 = guess)
plt.plot(z,phi_part)
plt.plot(z,f(z,*popt))
plt.show()

def g(z,*p):
    return p[0]*np.exp(-z**2 / p[1])

guess = [-rho_0,z_0]
popt,pcov = opt.curve_fit(g,z,phi_part,p0 = guess)
plt.plot(z,phi_part)
plt.plot(z,g(z,*popt))
plt.show()
```


400
350
300
250
200
150
100
50
0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


0
−5000
−10000
−15000
−20000
−25000
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


0
−5000
−10000
−15000
−20000
−25000
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

In [ ]: