Final Presentation

Here we show some emperical evidence on the proof we just presented

In [144…

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

We take 100 measurements and let the features of a signal be 50

In [145…

```
m = 100
n = 50
sim = 50000
A = np.random.rand(sim, m,n)
x = np.linspace(-1,1, n)
x = x.reshape(-1,1)
```

We create our signal using sin waves and cos waves, with noise be standrd gaussian

In [146…

```
signal = 2 * (x ** 2 + np.sin(0.8 * np.pi * 2*x) + np.cos(2.3 * 2 * np.pi *
epsilon =  np.random.rand(m,1)/np.sqrt(m)
y = A @ signal + epsilon
```

Meanwhile for comparison, we take 51 measurements and see how it recovers the signal

In [146…

```
A_60 = np.random.rand(51, n)
epsilon = np.random.rand(51,1)/np.sqrt(51)
y_60 = A_60 @ signal + epsilon
A_60_inverse = np.linalg.pinv(A_60)
recovery_60 = A_60_inverse @ y_60
```

In [147…

```
A_pseudoInverse = np.linalg.pinv(A)
recovery = np.matmul(A_pseudoInverse,y)
```

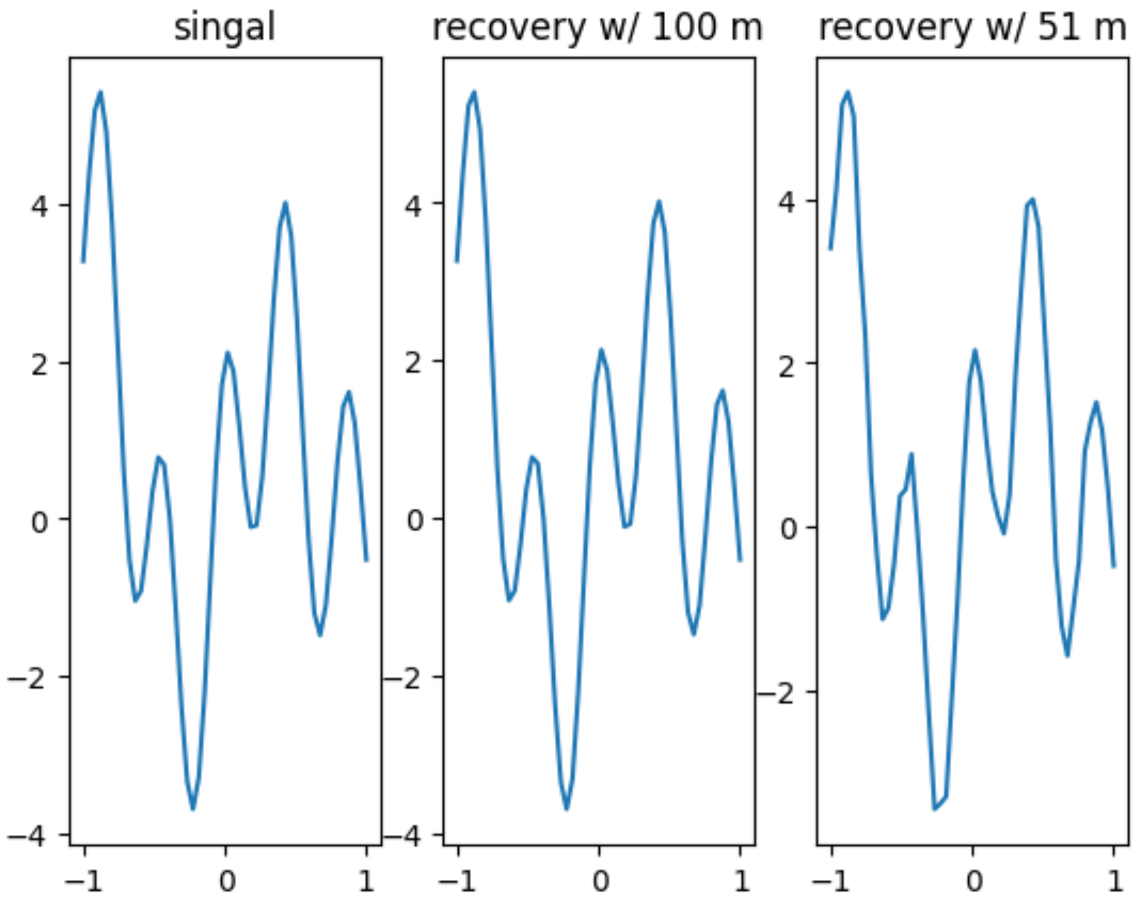
Here is the result we have

In [164…

```
plt.subplot(1,3,1)
plt.plot(x, signal)
plt.title("singal")
plt.subplot(1,3,2)
plt.title("recovery w/ 100 m")
plt.plot(x, recovery[1])
plt.subplot(1,3,3)
plt.title("recovery w/ 51 m")
plt.plot(x, recovery_60)
```

Out[164]:

```
[<matplotlib.lines.Line2D at 0x114500880>]
```



In [149…

```
diff = recovery - signal
diff = diff.reshape(sim,-1)
diff = diff**2
expectation = np.sum(np.sum(diff, axis = 1))/sim
print(expectation)
print(np.sum(np.sqrt(epsilon**2)))
```

```
0.009472988427881196
3.6003892844087004
```

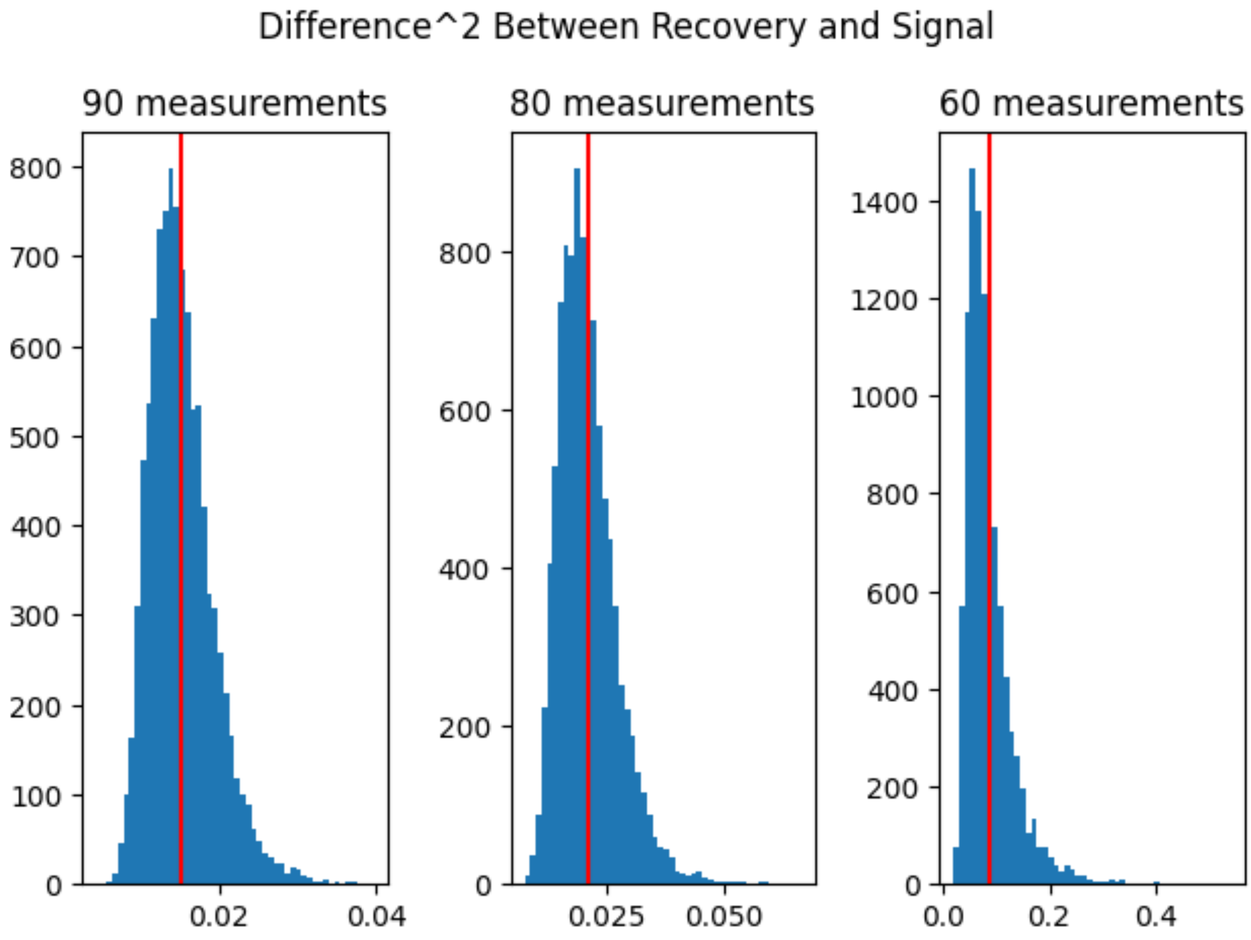
We also conduct experiments to verify our hypothesis

In [150…

```
def solve(m, sim, record ,signal, n = 50):

    epsilon = np.random.rand(m,1)/np.sqrt(m)
    A = np.random.rand(sim, m, n)
    y = A @ signal + epsilon
    A_pseudoInverse = np.linalg.pinv(A)
    recovery = A_pseudoInverse @ y
    diff = recovery - signal
    diff = diff.reshape(sim,-1)
    diff = diff**2
    record.append(np.sum(diff, axis = 1))
    expectation = np.sum((np.sum(diff, axis = 1)))/sim

    return expectation

record = []
results = []
for m in [50,60,80,90,100,110,120,150,160,200,250,300,500,1000]:
    results.append(solve(m, 10000, record, signal))
    print(results[-1])
```

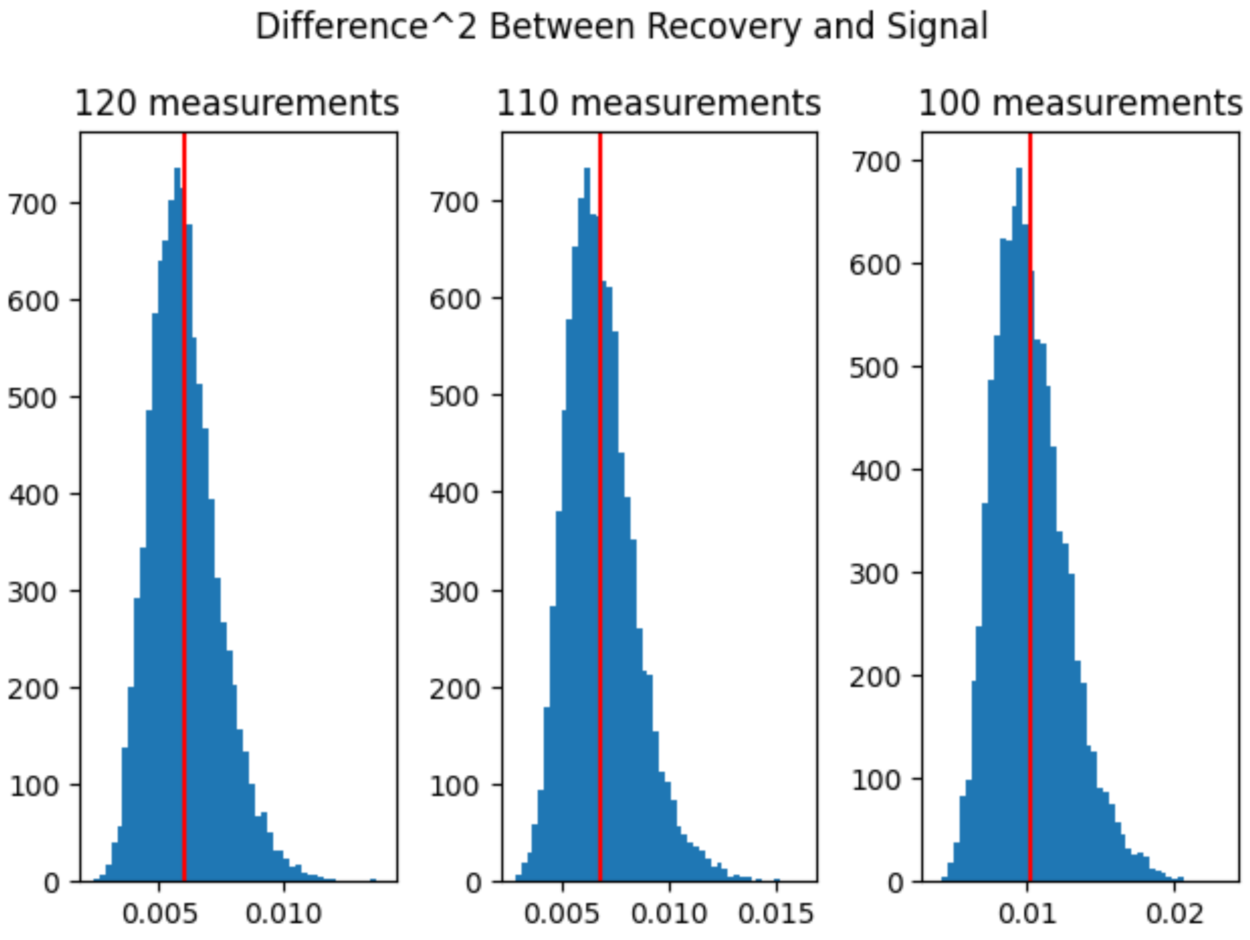
```
257009.12277709684
0.08538199912353575
0.021008263983959507
0.015272690506147945
0.010265080848013575
0.006828986663067629
0.006026646489020288
0.003683763434027587
0.0030899108753765434
0.001649621516173248
0.0010572607740816876
0.000700013676935496
0.00027084997375831127
7.040221897061117e-05
```

```
In [166…  plt.subplot(1,3,1)
          plt.hist(record[3],bins = 50)
          plt.axvline(results[3],color ="red")
          plt.title("90 measurements")
          plt.subplot(1,3,2)
          plt.hist(record[2],bins=50)
          plt.axvline(results[2],color ="red")
          plt.title("80 measurements")

          plt.subplot(1,3,3)
          plt.hist(record[1],bins = 50)
          plt.axvline(results[1],color ="red")
          plt.title("60 measurements")
          # plt.subplot(1,3,4)
          # plt.hist(record[0])
          # plt.axvline(results[0],color ="red")
          # plt.title("50 measurements")
          plt.suptitle("Difference^2 Between Recovery and Signal")
          plt.tight_layout()
```



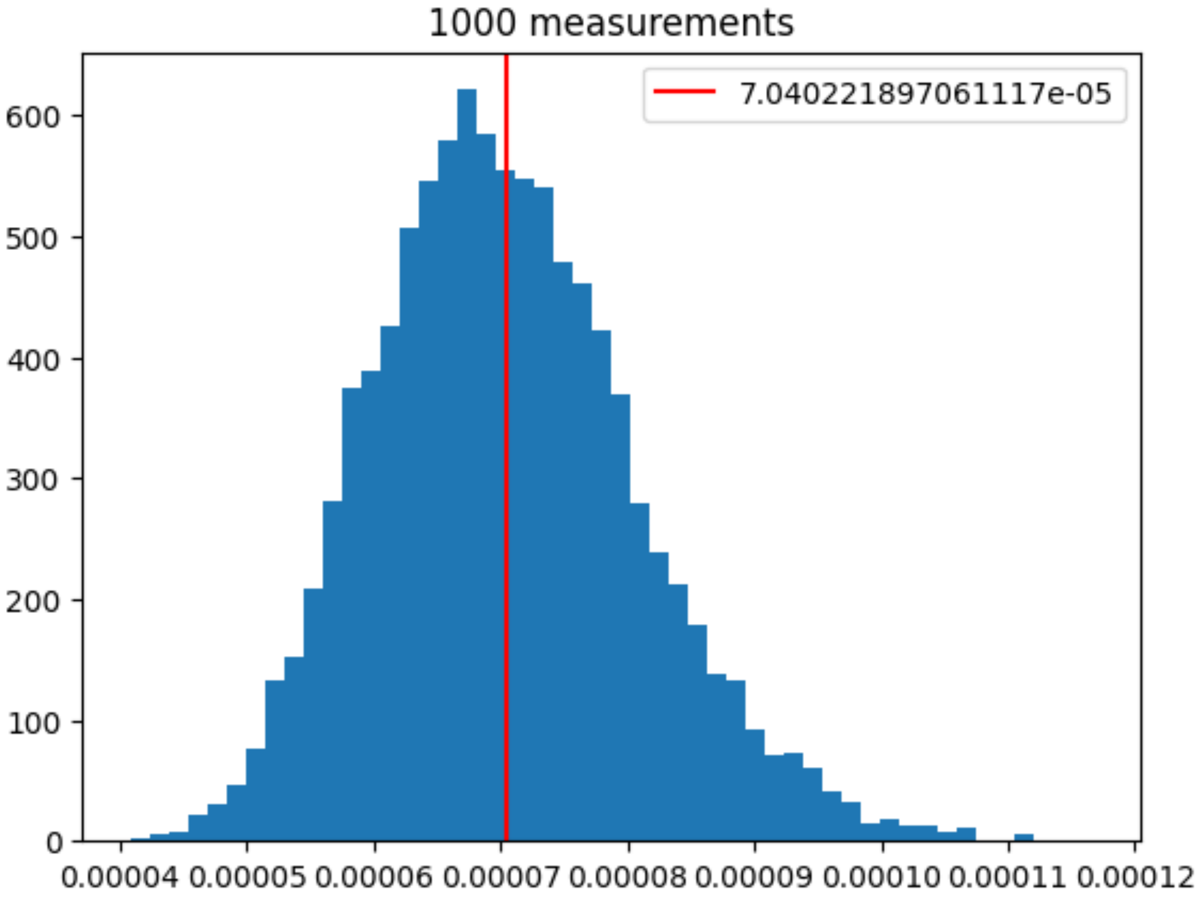
In [167…

```
plt.subplot(1,3,1)
plt.hist(record[6],bins = 50)
plt.axvline(results[6],color ="red")
plt.title("120 measurements")
plt.subplot(1,3,2)
plt.hist(record[5],bins=50)
plt.axvline(results[5],color ="red")
plt.title("110 measurements")

plt.subplot(1,3,3)
plt.hist(record[4],bins = 50)
plt.axvline(results[4],color ="red")
plt.title("100 measurements")
# plt.subplot(1,3,4)
# plt.hist(record[0])
# plt.axvline(results[0],color ="red")
# plt.title("50 measurements")
plt.suptitle("Difference^2 Between Recovery and Signal")
plt.tight_layout()
```



In [173…

```
plt.hist(record[-1],bins = 50)
plt.axvline(results[-1],color ="red",label= '{}'.format(results[-1]))
plt.title("1000 measurements")
leg = plt.legend(loc='upper right')
plt.show()
```

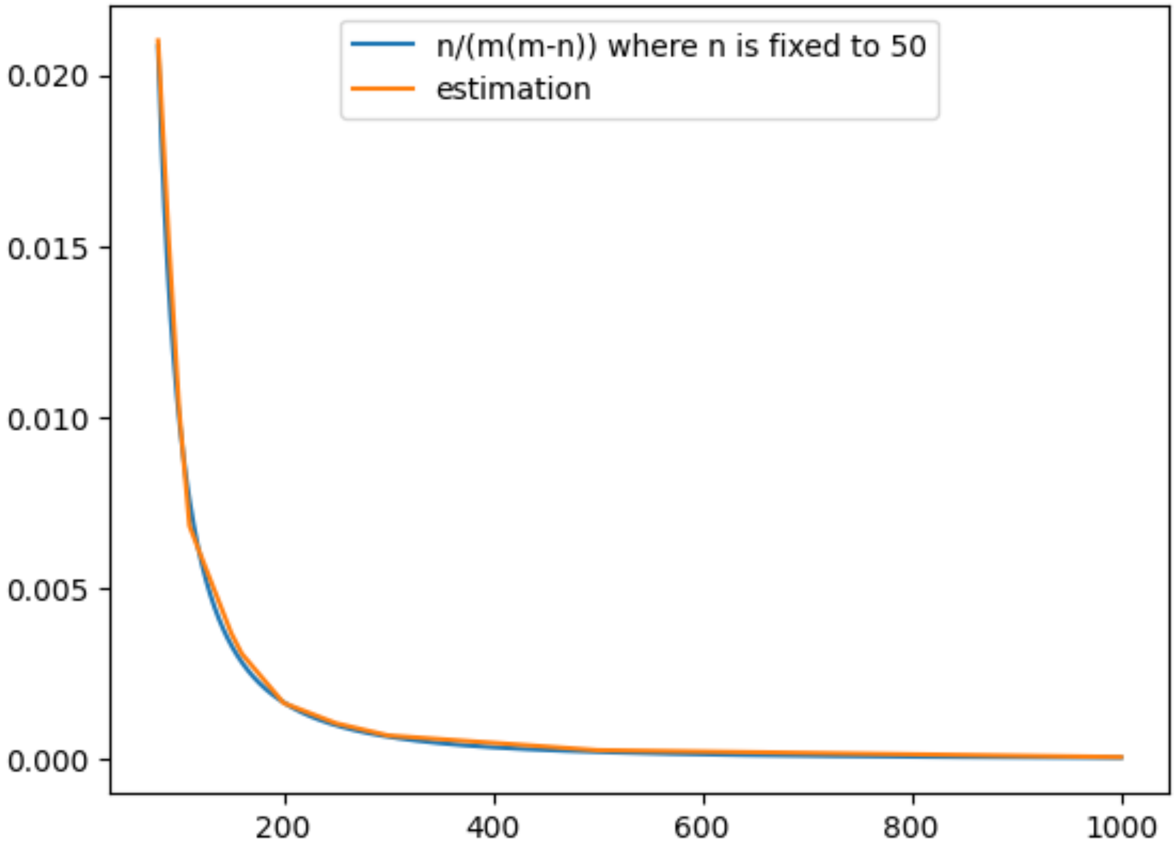


One Discovery

Also, after several experiments and some concultation with ChatGPT, we see that the $E\|\hat{x} - x\|_2^2\asymp \frac{n}{m}\frac{1}{m-n}$

In [177…

```
m = np.linspace(80,1000,921)
y = 50/m * (1/(m-50))
plt.plot(m,y,label='n/(m(m-n)) where n is fixed to 50')
plt.plot([80,90,100,110,120,150,160,200,250,300,500,1000],results[2:],label=
leg = plt.legend(loc='upper center')
plt.show()
```


n/(m(m-n)) where n is fixed to 50
estimation
0.020
0.015
0.010
0.005
0.000
200
400
600
800
1000

```
In [ ]:
```