Post Data Analysis

In [477]:

*# imports*

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**from** scipy **import** optimize

**import** math

plt.rcParams["figure.figsize"] **=** (20, 10)

# Load Data

## Set start and end points

By examine output data

In [478]:

*# set results path*

results\_file **=** "data\_out\_simple\_nozzle\_run\_without\_firing.csv"

startPoint **=** 37000 *# ms*

endPoint **=** 58000*# ms*

data **=** pd.read\_csv(results\_file)

duration **=** data["timestamp"].isin(range(startPoint, endPoint)) data **=** data[duration]

data\_np **=** data.to\_numpy()

## A quick overview of the raw data, check if any sensors are installed in the wrong orientation

check correlation matrix

In [479]:

corr\_mat **=** data.corr()

mask **=** np.zeros\_like(corr\_mat, dtype**=**bool) mask[np.triu\_indices\_from(mask)] **= True**

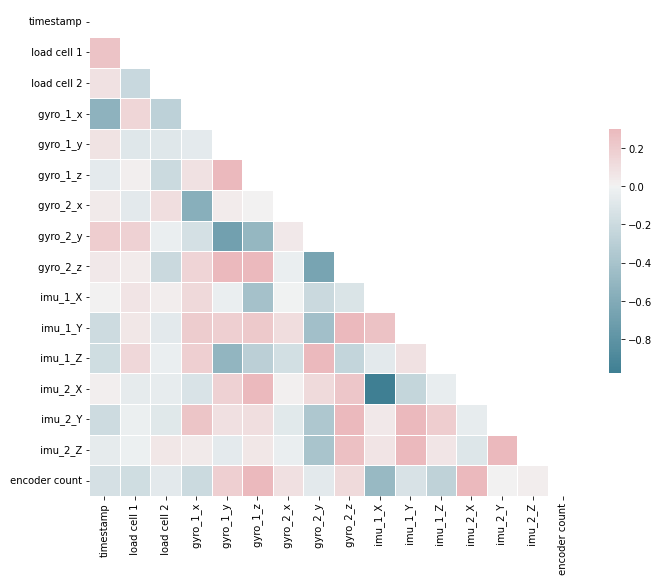
f, ax **=** plt.subplots(figsize**=**(11, 9))

cmap **=** sns.diverging\_palette(220, 10, as\_cmap**=True**)

sns.heatmap(corr\_mat, mask**=**mask, cmap**=**cmap, vmax**=**.3, center**=**0,

square**=True**, linewidths**=**.5, cbar\_kws**=**{"shrink": .5})

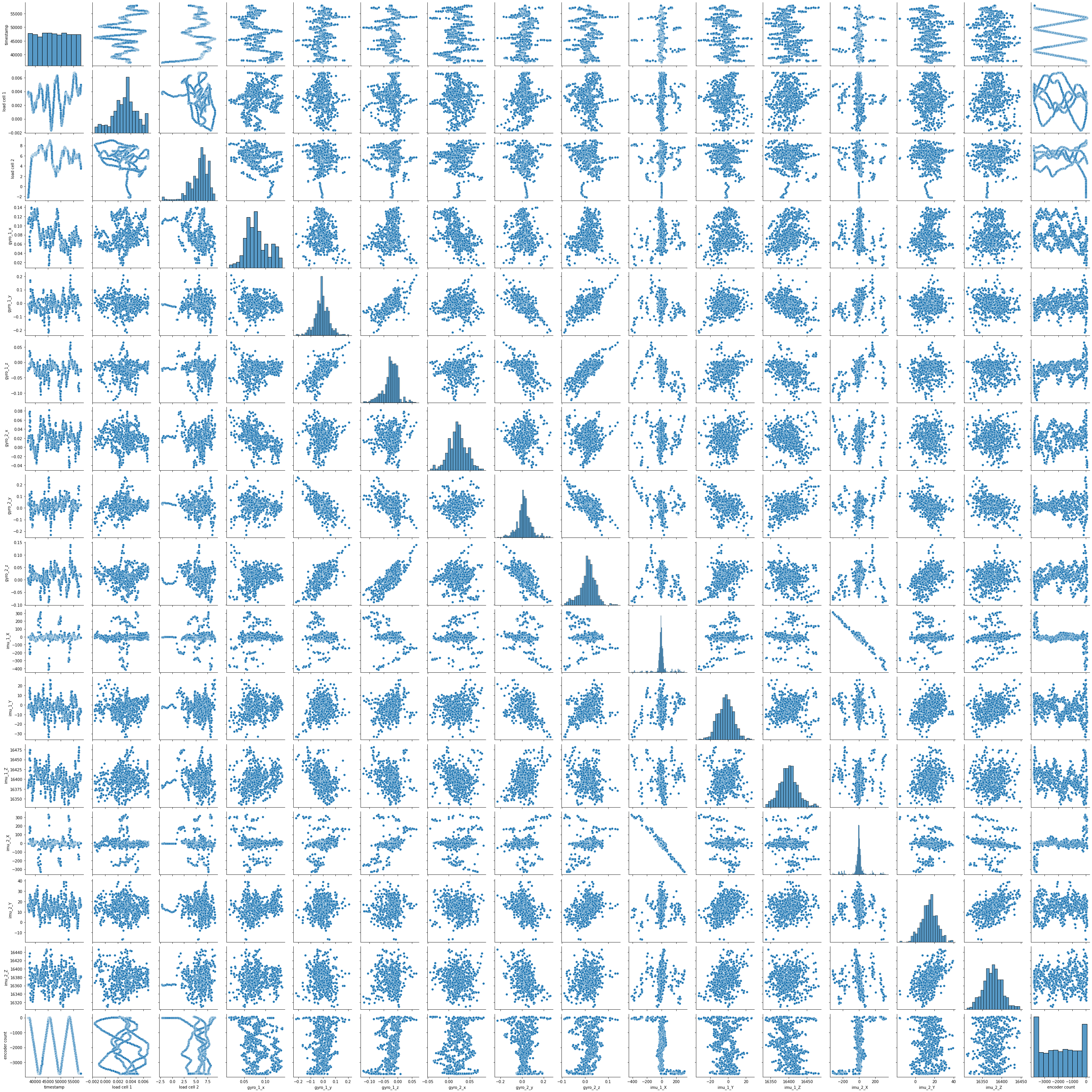
plt.show()



Actual pairwise chart. Takes a long time to render, uncomment and run if want to.

In [480]:

sns.pairplot(data) plt.show()



# Analise Linear Encoder Results

In [481]:

*# plot position*

plt.plot(time, position, 'r')

plt.plot(time, position\_lower\_bound, ':') plt.plot(time, position\_upper\_bound, ':') plt.title("Encoder Results")

plt.xlabel("time (s)")

plt.ylabel("position (m)") plt.show()

*# set specs of encoder*

resolution **=** 35.27777778 **/** 1000000 *# in m*

uncertainty **=** 93.4498 **/** 1000000 *# in m*

time **=** data\_np[:, 0]

time **=** time **-** time[0] *# in ms*

time **=** time **/** 1000 *# in s*

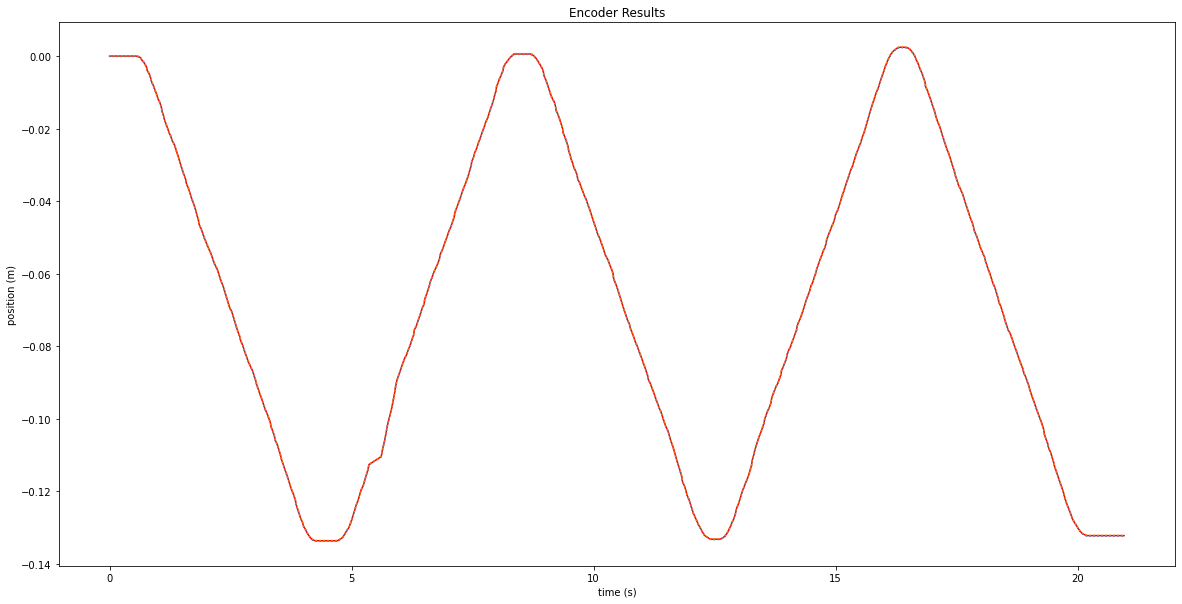
position **=** data\_np[:, 15]

position **=** position **-** position[0] *# zero the measurement*

*# assument starting position as at 0 micron, encoder resolution is 35.27777778 microns and*

position **=** position **\*** resolution

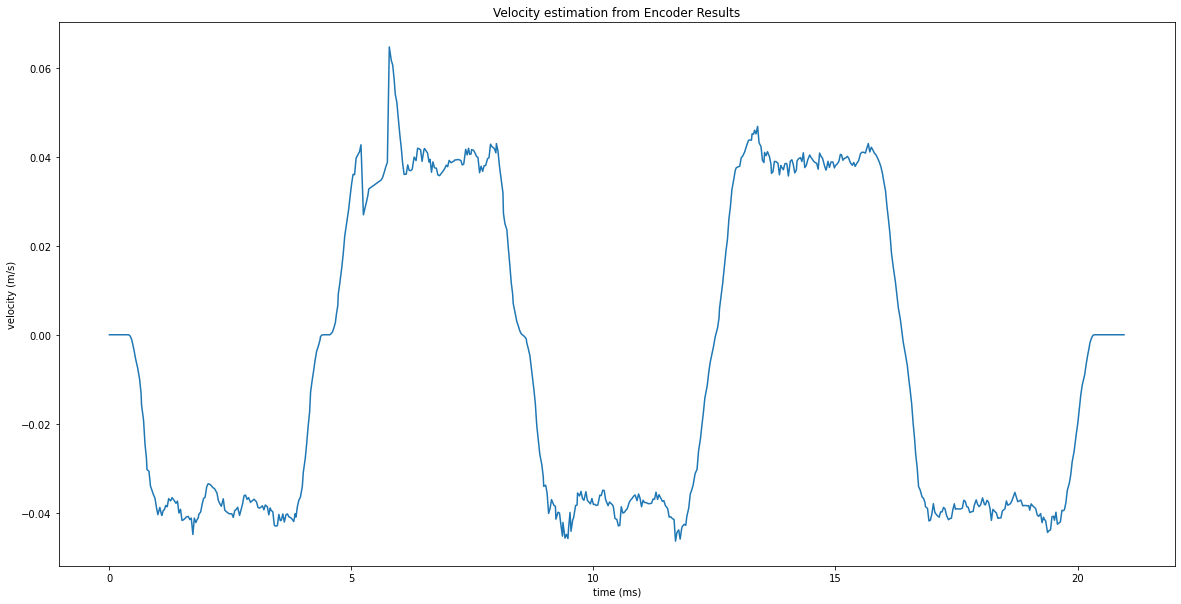
position\_lower\_bound **=** position **-** uncertainty position\_upper\_bound **=** position **+** uncertainty



## Velocity estimation from encoder data

In [482]:

|  |  |  |
| --- | --- | --- |
| velocity **=** np.array([float(0)] **\*** time.size) smooth\_factor **=** 5  **for** i **in** range(smooth\_factor):  **for** j **in** [1**+**i, **-**2**-**i]:  velocity[j] **=** (position[j**+**1**+**i] **-** position[j**-**1**-**i])**/**(time[j**+**1**+**i] **-** time[j**-**1**-**i])  **for** i **in** range(smooth\_factor, time.size**-**1**-**smooth\_factor):  velocity[i] **=** (position[i**+**smooth\_factor] **-** position[i**-**smooth\_factor])**/**(time[i**+**smooth\_fa plt.plot(time, velocity)  plt.title("Velocity estimation from Encoder Results") plt.xlabel("time (ms)")  plt.ylabel("velocity (m/s)") plt.show() | | |
|  |  |  |

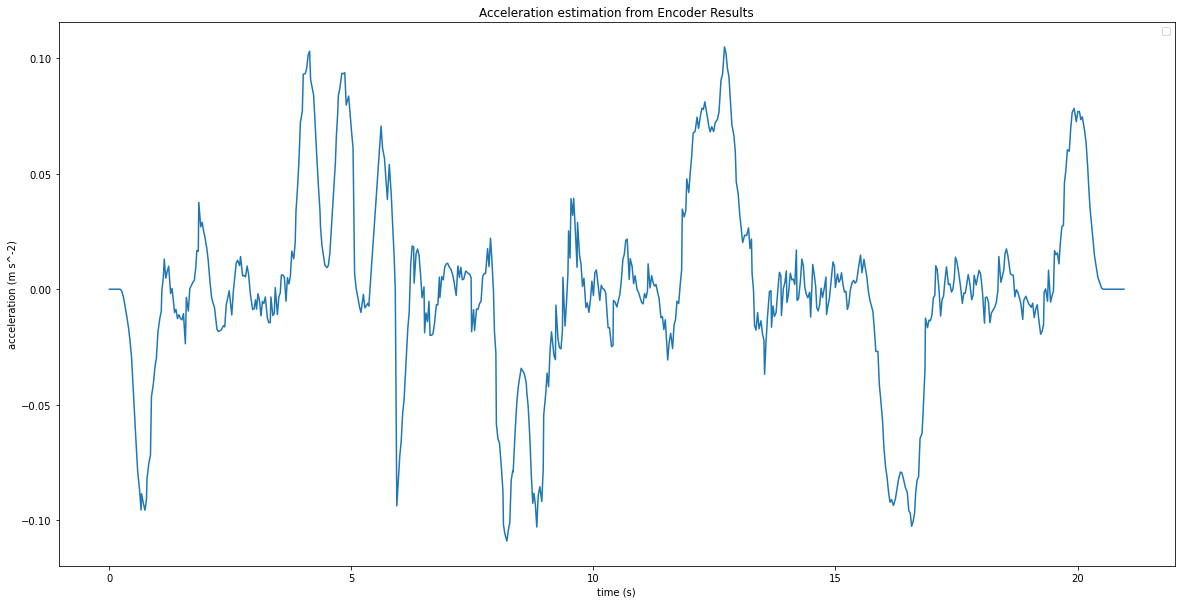


## Acceleration estimation from encoder velocity estimation

In [483]:

|  |  |  |
| --- | --- | --- |
| acceleration **=** np.array([float(0)] **\*** time.size) smooth\_factor **=** 5  **for** i **in** range(smooth\_factor):  **for** j **in** [1**+**i, **-**2**-**i]:  acceleration[j] **=** (velocity[j**+**1**+**i] **-** velocity[j**-**1**-**i])**/**(time[j**+**1**+**i] **-** time[j**-**1**-**i])  **for** i **in** range(smooth\_factor, time.size**-**1**-**smooth\_factor):  acceleration[i] **=** (velocity[i**+**smooth\_factor] **-** velocity[i**-**smooth\_factor])**/**(time[i**+**smoot plt.plot(time, acceleration)  plt.title("Acceleration estimation from Encoder Results") plt.xlabel("time (s)")  plt.ylabel("acceleration (m s^-2)")  *# plt.ylim((-0.1, 0.1))*  plt.legend(loc**=**"upper right") plt.show() | | |
|  |  |  |

No artists with labels found to put in legend. Note that artists whose labe l start with an underscore are ignored when legend() is called with no argum ent.



# IMU Results

## Acceleration on x direction

In [544]:

g **=** 9.8

acc1 **=** data\_np[:, 9] **/** 16384 **\*** g acc1 **=** acc1 **-** acc1[0]

acc2 **=** data\_np[:, 12] **\*** (**-**1) **/** 16384 **\*** g acc2 **=** acc2 **-** acc2[0]

x\_acc **=** (acc1 **+** acc2)**/**2

plt.plot(time, acc1, '.', label**=**"acceleration results from IMU 1") plt.plot(time, acc2, '.', label**=**"acceleration results from IMU 2")

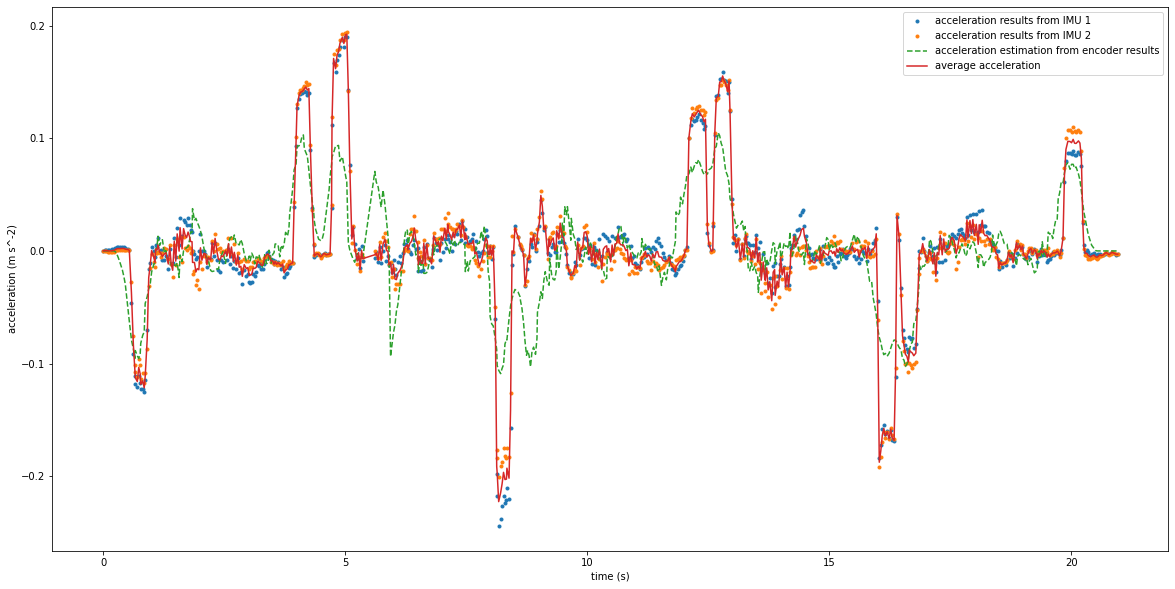
plt.plot(time, acceleration, "--", label**=**"acceleration estimation from encoder results") plt.plot(time, x\_acc, label**=**"average acceleration")

plt.xlabel("time (s)")

plt.ylabel("acceleration (m s^-2)")

*# plt.ylim((-1, 1))*

plt.legend(loc**=**"upper right") plt.show()



In [545]:

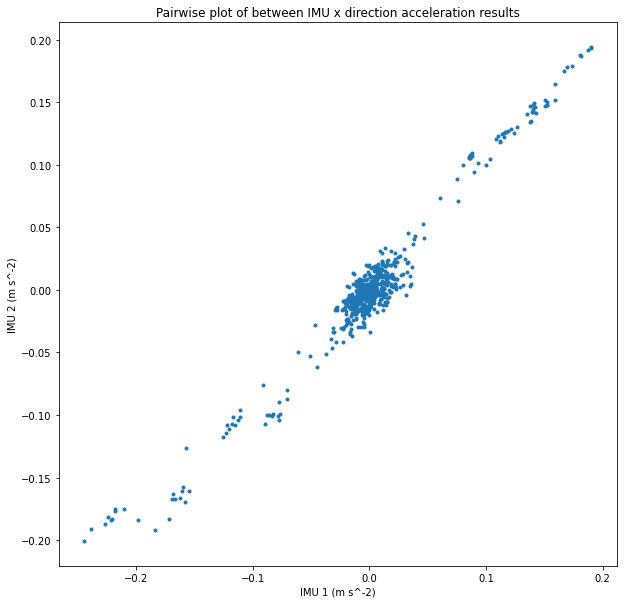
*# pairwise between IMU1 results and IMU2 results*

plt.rcParams["figure.figsize"] **=** (10, 10) plt.plot(acc1, acc2, '.')

plt.title("Pairwise plot of between IMU x direction acceleration results") plt.xlabel("IMU 1 (m s^-2)")

plt.ylabel("IMU 2 (m s^-2)") plt.show()

plt.rcParams["figure.figsize"] **=** (20, 10)



In [546]:

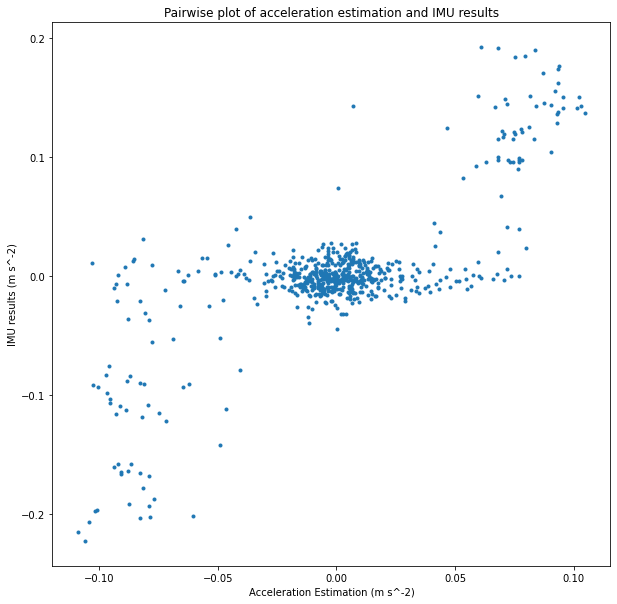
*# pairwise between IMU results and acceleration estimation*

plt.rcParams["figure.figsize"] **=** (10, 10) plt.plot(acceleration, x\_acc, '.')

plt.title("Pairwise plot of acceleration estimation and IMU results") plt.xlabel("Acceleration Estimation (m s^-2)")

plt.ylabel("IMU results (m s^-2)") plt.show()

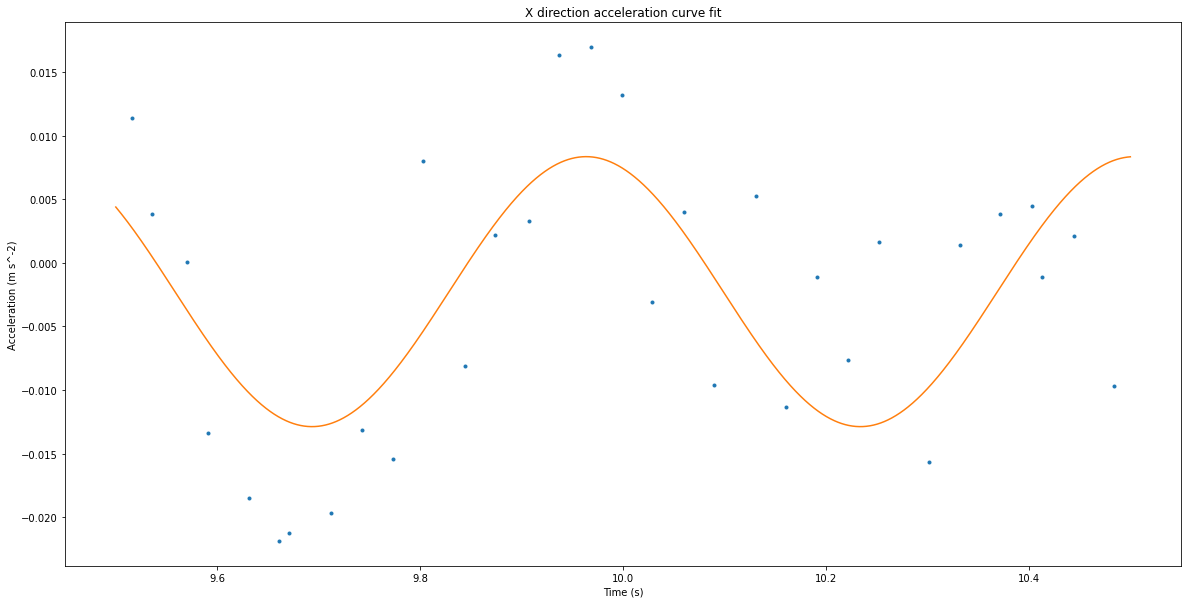
plt.rcParams["figure.figsize"] **=** (20, 10)



In [547]:



|  |  |  |
| --- | --- | --- |
| *# fit sine curve* start **=** 9.5 *# s* end **=** 10.5  *# estimation from graph*  A **=** 0.02  f **=** 1  d **=** 0  *# bounds = ((A[0],f[0], 0.01), (A[1],f[1], 2\*math.pi))*  t **=** time[(start **<=** time) **&** (time **<=** end)] y **=** x\_acc[(start **<=** time) **&** (time **<=** end)]  **def** sine\_func(t, A, f, phi, d):  **return** A**\***np.sin(2**\***math.pi**\***f**\***t **+** phi) **+** d  *# params, params\_covariance = optimize.curve\_fit(sine\_func, t, y, p0=[sum(A)/2, sum(f)/2, 0*  params, params\_covariance **=** optimize.curve\_fit(sine\_func, t, y, p0**=**[A, f, 0.1, d])  A **=** params[0] f **=** params[1] phi **=** params[2] d **=** params[3]  plt.plot(t, y, '.', label**=**"IMU results")  t **=** np.linspace(start, end, num**=**int((start **+** end)**/**0.01), endpoint**=True**) plt.plot(t, A**\***np.sin(2**\***math.pi**\***f**\***t **+** phi) **+** d, label**=**"Estimation")  plt.title("X direction acceleration curve fit") plt.xlabel("Time (s)")  plt.ylabel("Acceleration (m s^-2)") plt.show()  print("X direction vibration estimation")  print(f"Amplitute: {abs(A**/**(2**\***math.pi**\***f)**\*\***2) **\*** (10 **\*\*** 6)} microns") print(f"Frequency: {f} Hz") | | |
|  |  |  |
|  | | |





1. direction vibration estimation

Amplitute: 78.57397842737979 microns

Frequency: 1.8503225638826843 Hz

## accelerometer in y and z direction

In [567]:

acc1 **=** data\_np[:, 10] **/** 16384 **\*** g acc1 **=** acc1 **-** acc1[0]

acc2 **=** data\_np[:, 13] **\*** (**-**1) **/** 16384 **\*** g acc2 **=** acc2 **-** acc2[0]

y\_acc **=** (acc1 **+** acc2)**/**2

plt.title("Acceleration on Y direction")

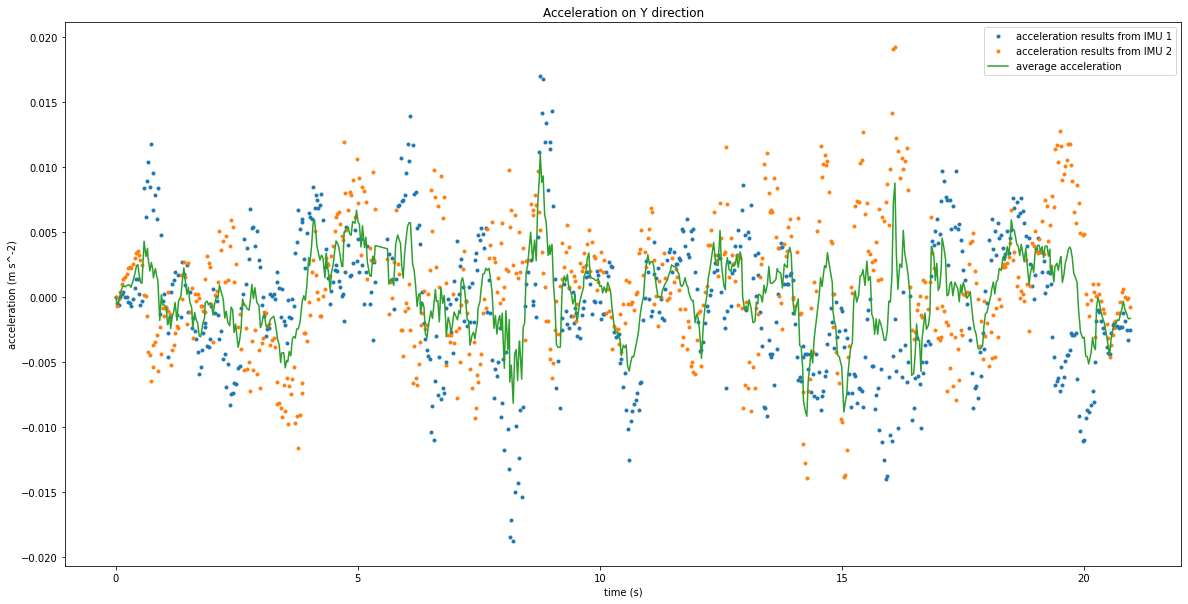
plt.plot(time, acc1, '.', label**=**"acceleration results from IMU 1") plt.plot(time, acc2, '.', label**=**"acceleration results from IMU 2") plt.plot(time, y\_acc, label**=**"average acceleration")

plt.xlabel("time (s)")

plt.ylabel("acceleration (m s^-2)")

*# plt.ylim((-1, 1))*

plt.legend(loc**=**"upper right") plt.show()



In [568]:

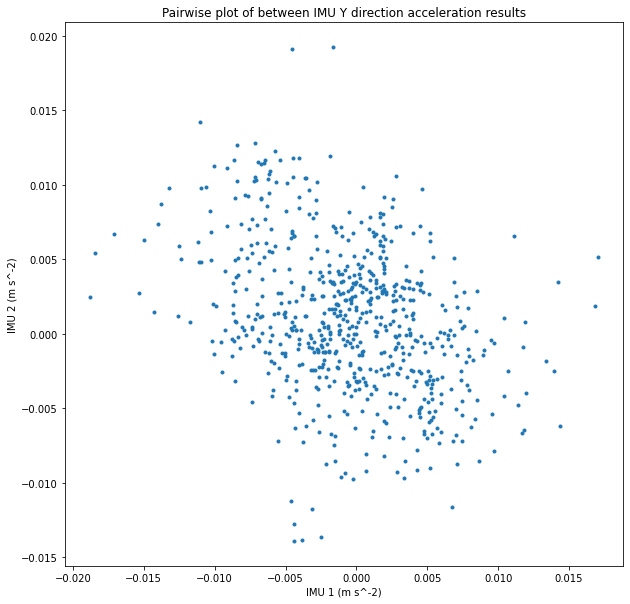
*# pairwise between IMU1 results and IMU2 results*

plt.rcParams["figure.figsize"] **=** (10, 10) plt.plot(acc1, acc2, '.')

plt.title("Pairwise plot of between IMU Y direction acceleration results") plt.xlabel("IMU 1 (m s^-2)")

plt.ylabel("IMU 2 (m s^-2)") plt.show()

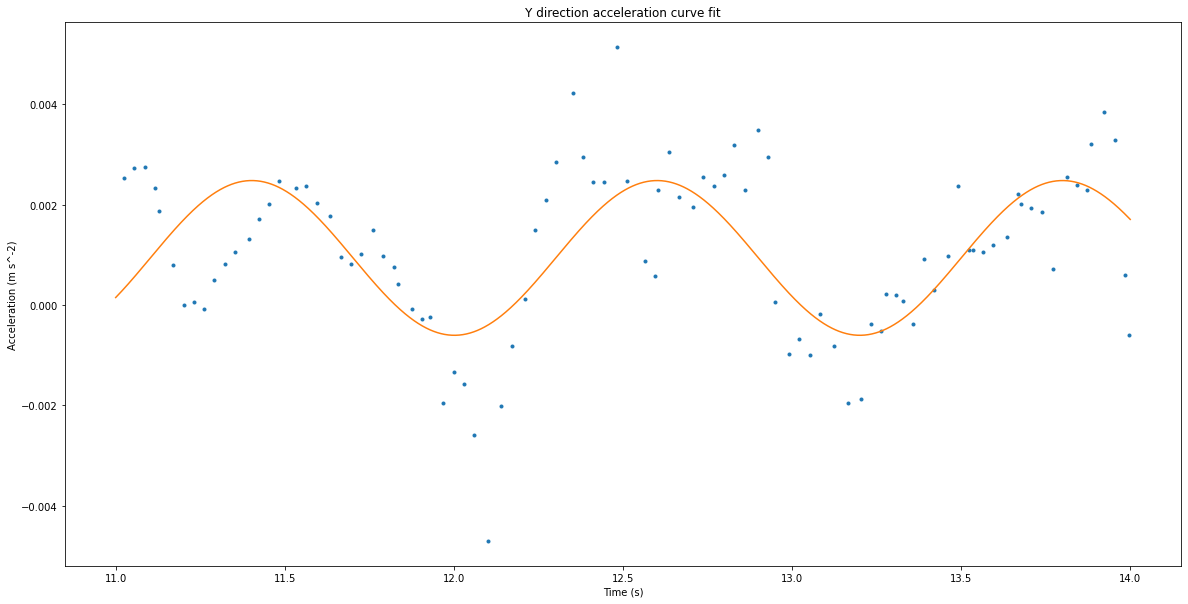
plt.rcParams["figure.figsize"] **=** (20, 10)



In [569]:



|  |  |  |
| --- | --- | --- |
| *# fit sine curve*  start **=** 11 *# s*  end **=** 14  *# estimation from graph*  A **=** 0.02  f **=** 1  d **=** 0  *# bounds = ((A[0],f[0], 0.01), (A[1],f[1], 2\*math.pi))*  t **=** time[(start **<=** time) **&** (time **<=** end)] y **=** y\_acc[(start **<=** time) **&** (time **<=** end)]  **def** sine\_func(t, A, f, phi, d):  **return** A**\***np.sin(2**\***math.pi**\***f**\***t **+** phi) **+** d  *# params, params\_covariance = optimize.curve\_fit(sine\_func, t, y, p0=[sum(A)/2, sum(f)/2, 0*  params, params\_covariance **=** optimize.curve\_fit(sine\_func, t, y, p0**=**[A, f, 0.1, d])  A **=** params[0] f **=** params[1] phi **=** params[2] d **=** params[3]  plt.plot(t, y, '.', label**=**"IMU results")  t **=** np.linspace(start, end, num**=**int((start **+** end)**/**0.01), endpoint**=True**) plt.plot(t, A**\***np.sin(2**\***math.pi**\***f**\***t **+** phi) **+** d, label**=**"Estimation")  plt.title("Y direction acceleration curve fit") plt.xlabel("Time (s)")  plt.ylabel("Acceleration (m s^-2)") plt.show()  print("Y direction vibration estimation")  print(f"Amplitute: {abs(A**/**(2**\***math.pi**\***f)**\*\***2) **\*** (10 **\*\*** 6)} microns") print(f"Frequency: {f} Hz") | | |
|  |  |  |





1. direction vibration estimation

Amplitute: 56.08901733223656 microns

Frequency: 0.8341824856753561 Hz

In [570]:

acc1 **=** data\_np[:, 11] **/** 16384 **\*** g acc1 **=** acc1 **-** acc1[0]

acc2 **=** data\_np[:, 14] **/** 16384 **\*** g acc2 **=** acc2 **-** acc2[0]

z\_acc **=** (acc1 **+** acc2)**/**2

plt.title("Acceleration on Z direction")

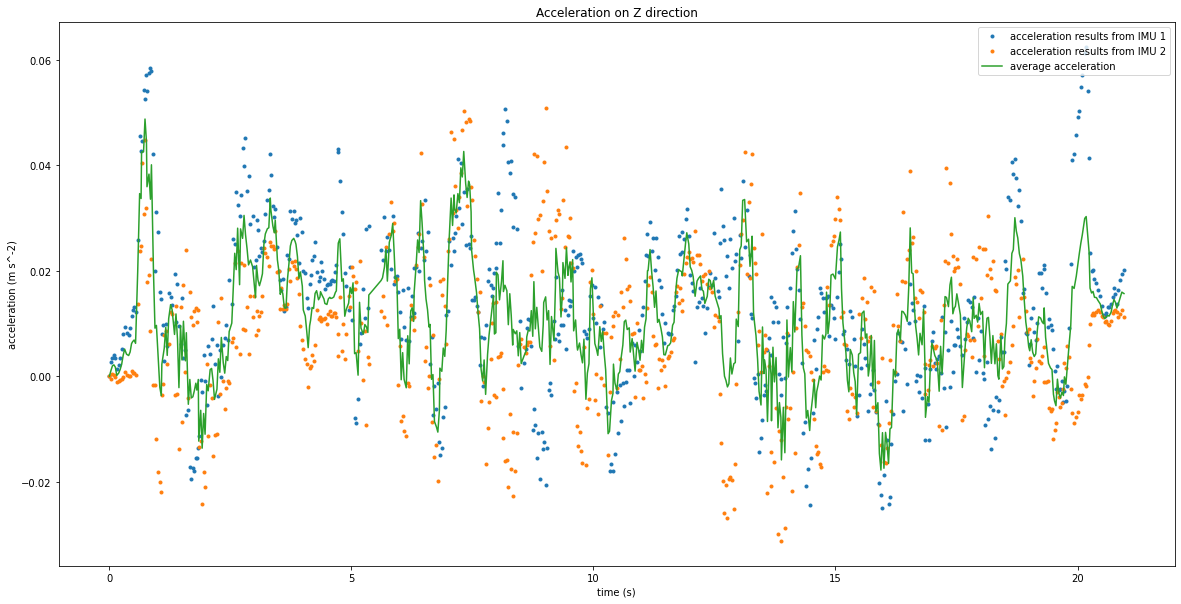
plt.plot(time, acc1, '.', label**=**"acceleration results from IMU 1") plt.plot(time, acc2, '.', label**=**"acceleration results from IMU 2") plt.plot(time, z\_acc, label**=**"average acceleration")

plt.xlabel("time (s)")

plt.ylabel("acceleration (m s^-2)")

*# plt.ylim((-1, 1))*

plt.legend(loc**=**"upper right") plt.show()



In [ ]:

*# pairwise between IMU1 results and IMU2 results*

plt.rcParams["figure.figsize"] **=** (10, 10) plt.plot(acc1, acc2, '.')

plt.title("Pairwise plot of between IMU Z direction acceleration results") plt.xlabel("IMU 1 (m s^-2)")

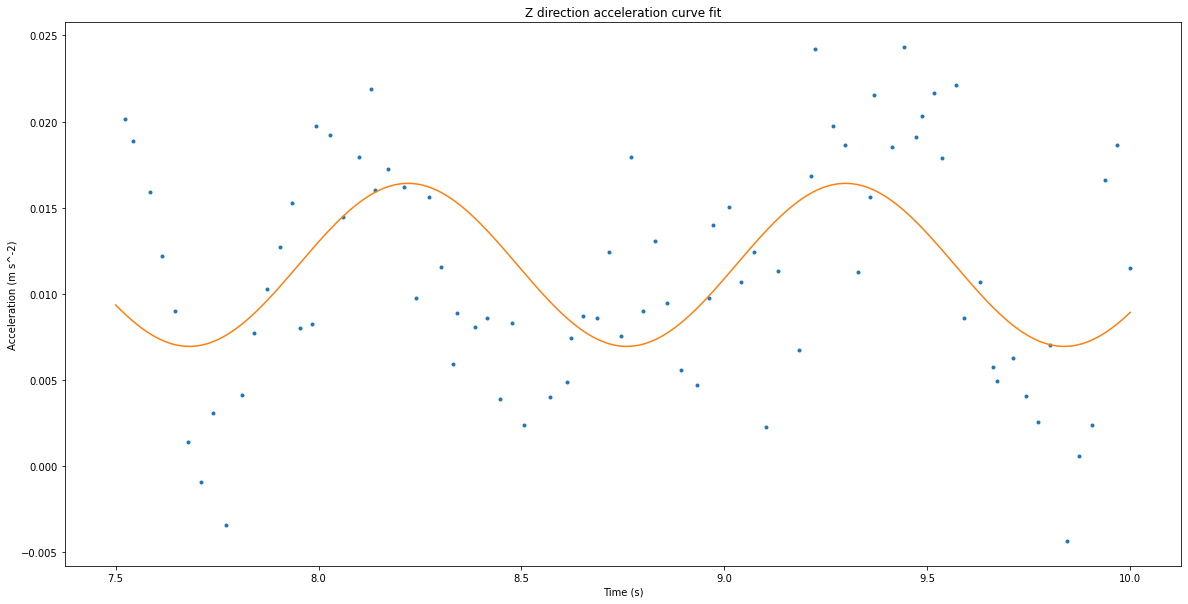
plt.ylabel("IMU 2 (m s^-2)") plt.show()

plt.rcParams["figure.figsize"] **=** (20, 10)

In [571]:



|  |  |  |
| --- | --- | --- |
| *# fit sine curve* start **=** 7.5 *# s* end **=** 10  *# estimation from graph*  A **=** 0.02  f **=** 1  d **=** 0  *# bounds = ((A[0],f[0], 0.01), (A[1],f[1], 2\*math.pi))*  t **=** time[(start **<=** time) **&** (time **<=** end)] y **=** z\_acc[(start **<=** time) **&** (time **<=** end)]  **def** sine\_func(t, A, f, phi, d):  **return** A**\***np.sin(2**\***math.pi**\***f**\***t **+** phi) **+** d  *# params, params\_covariance = optimize.curve\_fit(sine\_func, t, y, p0=[sum(A)/2, sum(f)/2, 0*  params, params\_covariance **=** optimize.curve\_fit(sine\_func, t, y, p0**=**[A, f, 0.1, d])  A **=** params[0] f **=** params[1] phi **=** params[2] d **=** params[3]  plt.plot(t, y, '.', label**=**"IMU results")  t **=** np.linspace(start, end, num**=**int((start **+** end)**/**0.01), endpoint**=True**) plt.plot(t, A**\***np.sin(2**\***math.pi**\***f**\***t **+** phi) **+** d, label**=**"Estimation")  plt.title("Z direction acceleration curve fit") plt.xlabel("Time (s)")  plt.ylabel("Acceleration (m s^-2)") plt.show()  print("Z direction vibration estimation")  print(f"Amplitute: {abs(A**/**(2**\***math.pi**\***f)**\*\***2) **\*** (10 **\*\*** 6)} microns") print(f"Frequency: {f} Hz") | | |
|  |  |  |





1. direction vibration estimation

Amplitute: 139.39622160973454 microns

Frequency: 0.927616956331717 Hz

In [ ]:

# Rotational Results

In [572]:

*# rotation on x direction*

rot1 **=** data\_np[:, 3] rot1 **=** rot1 **-** rot1[0]

rot2 **=** data\_np[:, 6] **\*** (**-**1) rot2 **=** rot2 **-** rot2[0]

x\_rot **=** (rot1 **+** rot2)**/**2

plt.plot(time, x\_rot, label**=**"average") plt.plot(time, rot1, '.', label**=**"IMU 1")

plt.plot(time, rot2, '.', label**=**"IMU 2")

*# plt.ylim((-1, 1))*

plt.xlabel("time (s)")

plt.ylabel("rotational acceleration")

plt.title("x direction rotational vibration results") plt.show()

*# pairwise between IMU1 results and IMU2 results*

plt.rcParams["figure.figsize"] **=** (10, 10) plt.plot(rot1, rot2, '.')

plt.title("Pairwise plot of between IMU x direction rotation acceleration results") plt.xlabel("IMU 1 (degree s^-2)")

plt.ylabel("IMU 2 (degree s^-2)") plt.show()

plt.rcParams["figure.figsize"] **=** (20, 10)

*# rotation on y direction*

rot1 **=** data\_np[:, 4] rot1 **=** rot1 **-** rot1[0]

rot2 **=** data\_np[:, 7] **\*** (**-**1) rot2 **=** rot2 **-** rot2[0]

y\_rot **=** (rot1 **+** rot2)**/**2

plt.plot(time, y\_rot, label**=**"average") plt.plot(time, rot1, '.', label**=**"IMU 1")

plt.plot(time, rot2, '.', label**=**"IMU 2")

*# plt.ylim((-1, 1))*

plt.xlabel("time (s)")

plt.ylabel("rotational acceleration (degree s^-2)")

plt.title("y direction rotational vibration results") plt.show()

*# pairwise between IMU1 results and IMU2 results*

plt.rcParams["figure.figsize"] **=** (10, 10) plt.plot(rot1, rot2, '.')

plt.title("Pairwise plot of between IMU y direction rotation acceleration results") plt.xlabel("IMU 1 (degree s^-2)")

plt.ylabel("IMU 2 (degree s^-2)") plt.show()

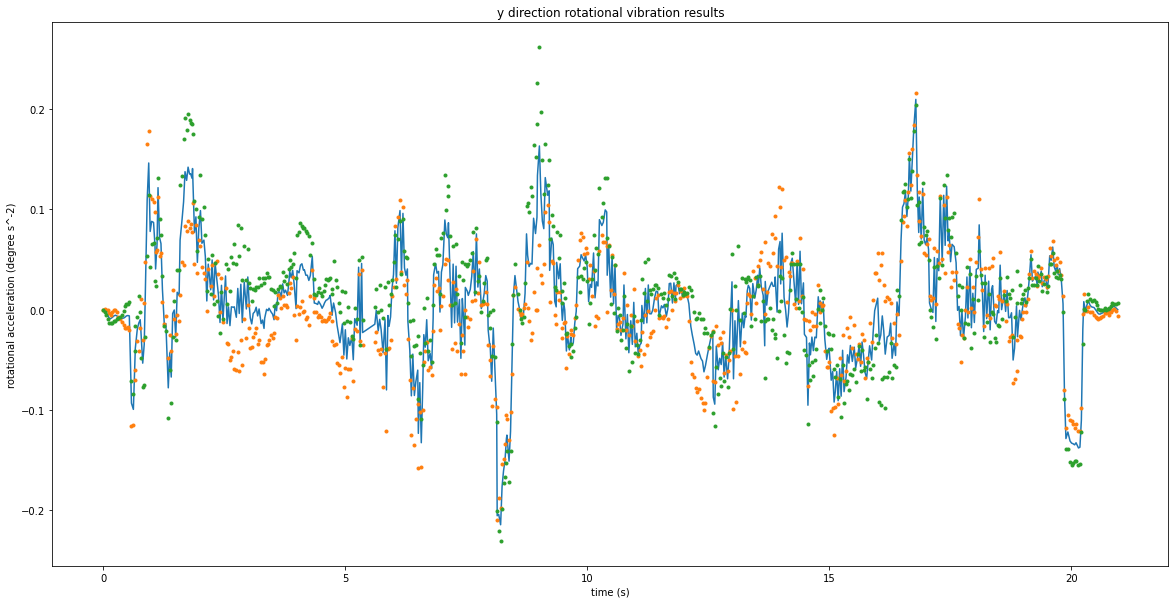
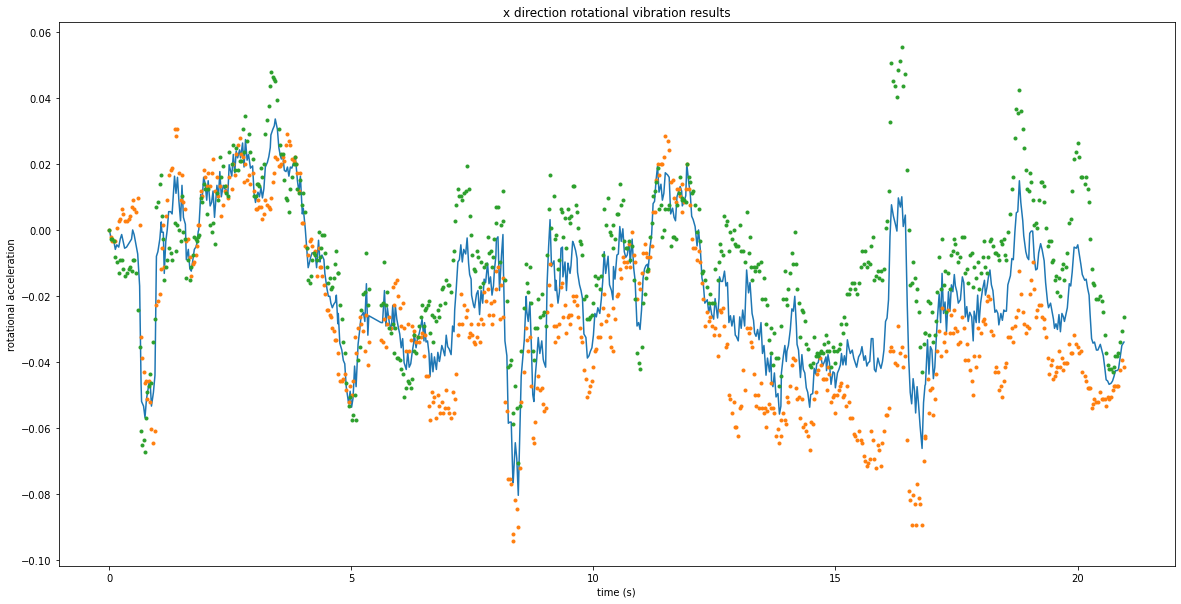
plt.rcParams["figure.figsize"] **=** (20, 10)

*# rotation on z direction*

rot1 **=** data\_np[:, 5] rot1 **=** rot1 **-** rot1[0] rot2 **=** data\_np[:, 8] rot2 **=** rot2 **-** rot2[0]

z\_rot **=** (rot1 **+** rot2)**/**2

plt.plot(time, z\_rot, label**=**"average") plt.plot(time, rot1, '.', label**=**"IMU 1")



plt.plot(time, rot2, '.', label**=**"IMU 2")

*# plt.ylim((-1, 1))*

plt.xlabel("time (s)")

plt.ylabel("rotational acceleration (degree s^-2)")

plt.title("z direction rotational vibration results") plt.show()

*# pairwise between IMU1 results and IMU2 results*

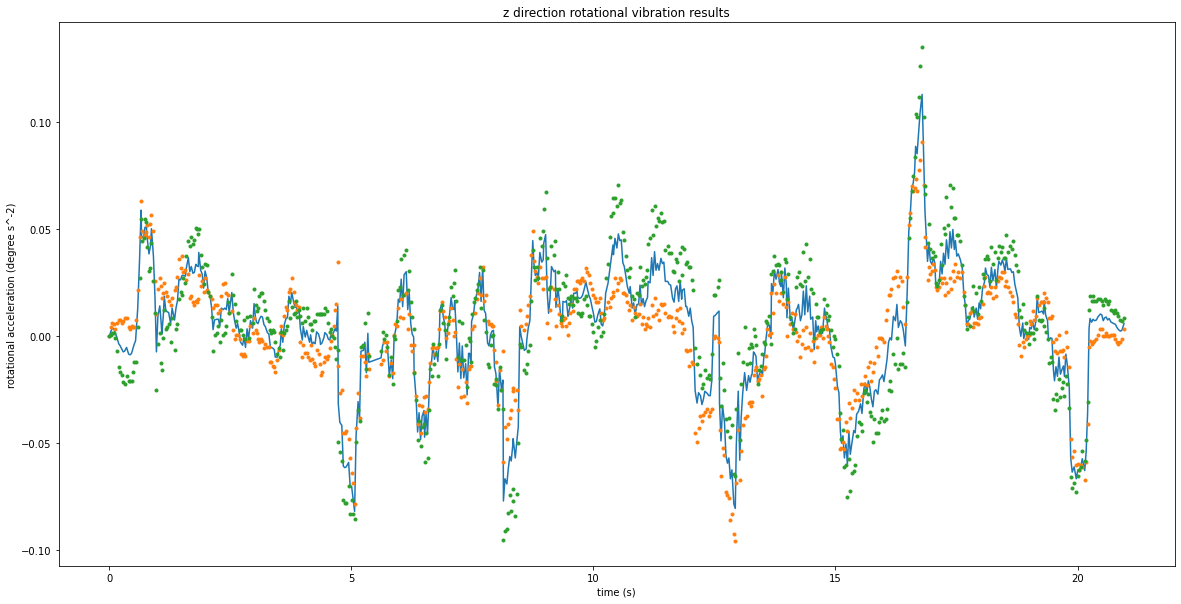
plt.rcParams["figure.figsize"] **=** (10, 10) plt.plot(rot1, rot2, '.')

plt.title("Pairwise plot of between IMU z direction rotation acceleration results") plt.xlabel("IMU 1 (degree s^-2)")

plt.ylabel("IMU 2 (degree s^-2)") plt.show()

plt.rcParams["figure.figsize"] **=** (20, 10)



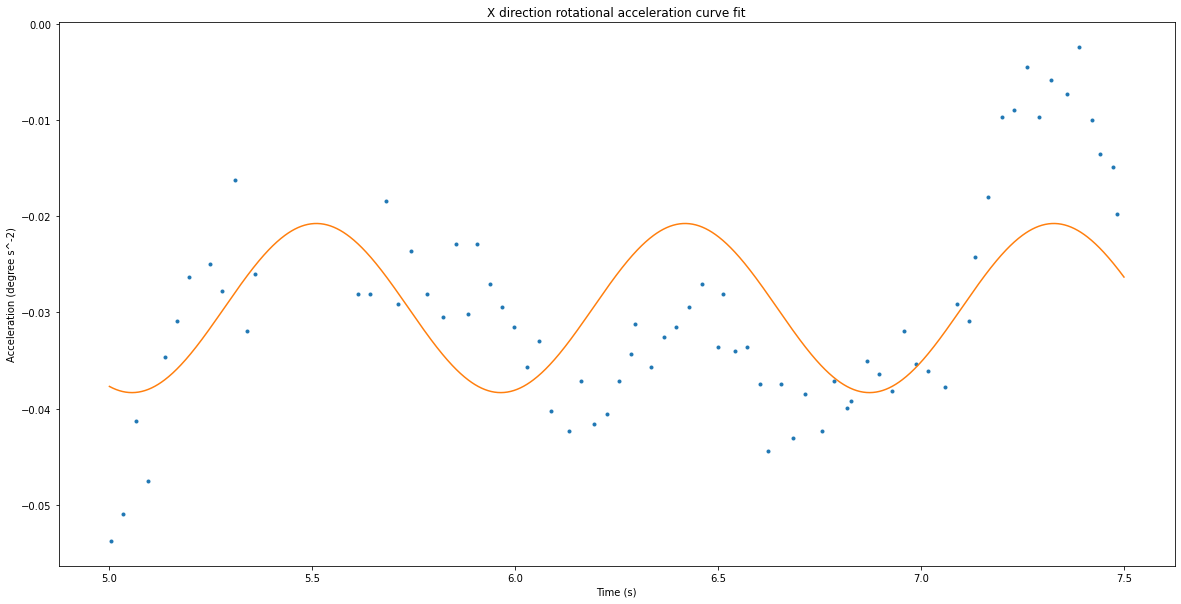




In [573]:



|  |  |  |
| --- | --- | --- |
| *## fit sine curve*  start **=** 5 *# s*  end **=** 7.5  *# estimation from graph*  A **=** 0.02  f **=** 1  d **=** 0  *# bounds = ((A[0],f[0], 0.01), (A[1],f[1], 2\*math.pi))*  t **=** time[(start **<=** time) **&** (time **<=** end)] y **=** x\_rot[(start **<=** time) **&** (time **<=** end)]  **def** sine\_func(t, A, f, phi, d):  **return** A**\***np.sin(2**\***math.pi**\***f**\***t **+** phi) **+** d  *# params, params\_covariance = optimize.curve\_fit(sine\_func, t, y, p0=[sum(A)/2, sum(f)/2, 0*  params, params\_covariance **=** optimize.curve\_fit(sine\_func, t, y, p0**=**[A, f, 0.1, d])  A **=** params[0] f **=** params[1] phi **=** params[2] d **=** params[3]  plt.plot(t, y, '.', label**=**"IMU results")  t **=** np.linspace(start, end, num**=**int((start **+** end)**/**0.01), endpoint**=True**) plt.plot(t, A**\***np.sin(2**\***math.pi**\***f**\***t **+** phi) **+** d, label**=**"Estimation")  plt.title("X direction rotational acceleration curve fit") plt.xlabel("Time (s)")  plt.ylabel("Acceleration (degree s^-2)") plt.show()  print("X direction rotation vibration estimation")  print(f"Amplitute: {abs(A**/**(2**\***math.pi**\***f)**\*\***2)} degree") print(f"Frequency: {f} Hz") | | |
|  |  |  |
|  | | |





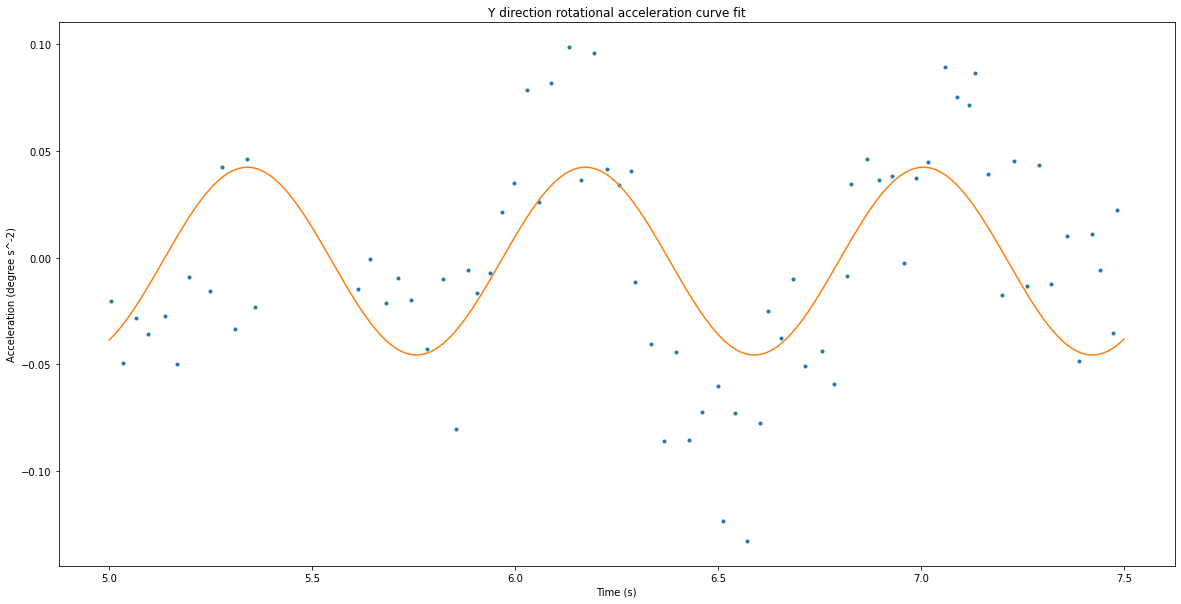
1. direction rotation vibration estimation Amplitute: 0.0001836442756442195 degree

Frequency: 1.1005898997055354 Hz

In [574]:



|  |  |  |
| --- | --- | --- |
| *## fit sine curve*  start **=** 5 *# s*  end **=** 7.5  *# estimation from graph*  A **=** 0.02  f **=** 1  d **=** 0  *# bounds = ((A[0],f[0], 0.01), (A[1],f[1], 2\*math.pi))*  t **=** time[(start **<=** time) **&** (time **<=** end)] y **=** y\_rot[(start **<=** time) **&** (time **<=** end)]  **def** sine\_func(t, A, f, phi, d):  **return** A**\***np.sin(2**\***math.pi**\***f**\***t **+** phi) **+** d  *# params, params\_covariance = optimize.curve\_fit(sine\_func, t, y, p0=[sum(A)/2, sum(f)/2, 0*  params, params\_covariance **=** optimize.curve\_fit(sine\_func, t, y, p0**=**[A, f, 0.1, d])  A **=** params[0] f **=** params[1] phi **=** params[2] d **=** params[3]  plt.plot(t, y, '.', label**=**"IMU results")  t **=** np.linspace(start, end, num**=**int((start **+** end)**/**0.01), endpoint**=True**) plt.plot(t, A**\***np.sin(2**\***math.pi**\***f**\***t **+** phi) **+** d, label**=**"Estimation")  plt.title("Y direction rotational acceleration curve fit") plt.xlabel("Time (s)")  plt.ylabel("Acceleration (degree s^-2)") plt.show()  print("Y direction rotation vibration estimation")  print(f"Amplitute: {abs(A**/**(2**\***math.pi**\***f)**\*\***2)} degree") print(f"Frequency: {f} Hz") | | |
|  |  |  |
|  | | |





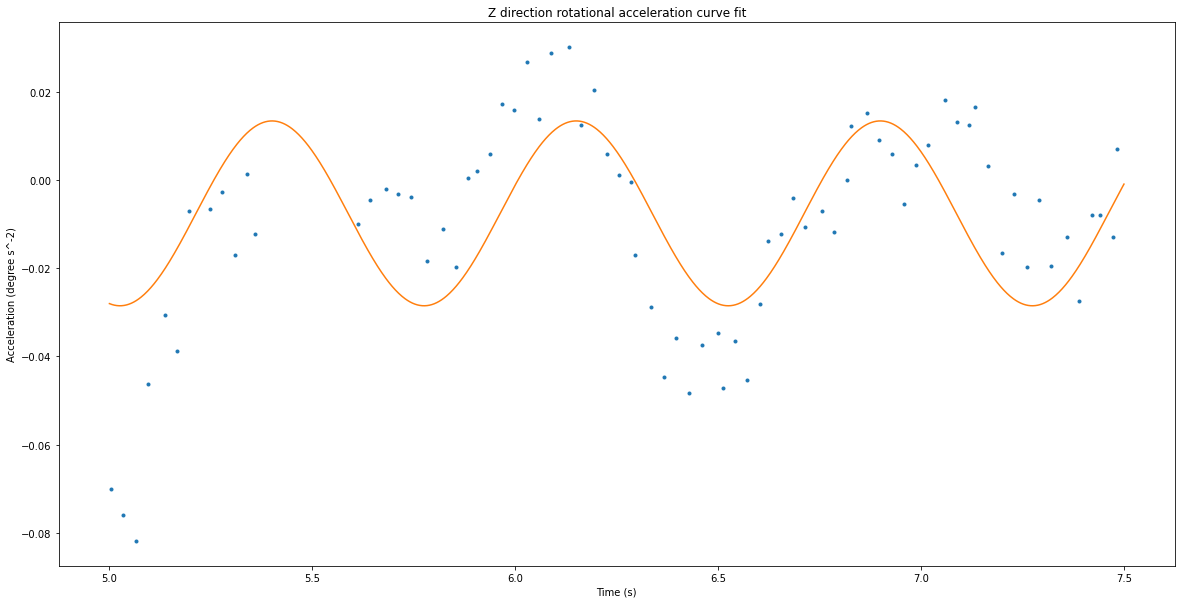
1. direction rotation vibration estimation Amplitute: 0.0007729214965654529 degree

Frequency: 1.20073125655508 Hz

In [575]:



|  |  |  |
| --- | --- | --- |
| *## fit sine curve*  start **=** 5 *# s*  end **=** 7.5  *# estimation from graph*  A **=** 0.02  f **=** 1  d **=** 0  *# bounds = ((A[0],f[0], 0.01), (A[1],f[1], 2\*math.pi))*  t **=** time[(start **<=** time) **&** (time **<=** end)] y **=** z\_rot[(start **<=** time) **&** (time **<=** end)]  **def** sine\_func(t, A, f, phi, d):  **return** A**\***np.sin(2**\***math.pi**\***f**\***t **+** phi) **+** d  *# params, params\_covariance = optimize.curve\_fit(sine\_func, t, y, p0=[sum(A)/2, sum(f)/2, 0*  params, params\_covariance **=** optimize.curve\_fit(sine\_func, t, y, p0**=**[A, f, 0.1, d])  A **=** params[0] f **=** params[1] phi **=** params[2] d **=** params[3]  plt.plot(t, y, '.', label**=**"IMU results")  t **=** np.linspace(start, end, num**=**int((start **+** end)**/**0.01), endpoint**=True**) plt.plot(t, A**\***np.sin(2**\***math.pi**\***f**\***t **+** phi) **+** d, label**=**"Estimation")  plt.title("Z direction rotational acceleration curve fit") plt.xlabel("Time (s)")  plt.ylabel("Acceleration (degree s^-2)") plt.show()  print("Z direction rotation vibration estimation")  print(f"Amplitute: {abs(A**/**(2**\***math.pi**\***f)**\*\***2)} degree") print(f"Frequency: {f} Hz") | | |
|  |  |  |
|  | | |





1. direction rotation vibration estimation Amplitute: 0.00029802670813540476 degree

Frequency: 1.334359258401142 Hz

# Tension Results

In [576]:

'''

load1 = data\_np[:, 1]

load2 = data\_np[:, 2] \* (-1) tension = (load1 + load2) / 2

plt.plot(time, tension, label="average tension")

plt.plot(time, load1, '.', label="tension from load cell 1") plt.plot(time, load2, '.', label="tension from load cell 2") plt.xlabel("time (ms)")

plt.ylabel("tension")

plt.title("tension results from load cells") plt.legend(loc="upper right")

plt.show() '''

Out[576]:

'\nload1 = data\_np[:, 1]\nload2 = data\_np[:, 2] \* (-1)\ntension = (load1 + l oad2) / 2\nplt.plot(time, tension, label="average tension")\nplt.plot(time, load1, \'.\', label="tension from load cell 1")\nplt.plot(time, load2,

\'.\', label="tension from load cell 2")\nplt.xlabel("time (ms)")\nplt.ylabe l("tension")\nplt.title("tension results from load cells")\nplt.legend(loc

="upper right")\nplt.show()\n'

# pairwise chart for everything

In [566]:

|  |  |  |
| --- | --- | --- |
| *# final data*  final\_data **=** pd.DataFrame({"time": time, "encoder": position, "velocity estimate": velocity corr\_mat **=** final\_data.corr()  mask **=** np.zeros\_like(corr\_mat, dtype**=**bool) mask[np.triu\_indices\_from(mask)] **= True**  f, ax **=** plt.subplots(figsize**=**(11, 9))  cmap **=** sns.diverging\_palette(220, 10, as\_cmap**=True**)  sns.heatmap(corr\_mat, mask**=**mask, cmap**=**cmap, vmax**=**.3, center**=**0,  square**=True**, linewidths**=**.5, cbar\_kws**=**{"shrink": .5})  plt.show()  *# sns.pairplot(final\_data) # plt.show()* | | |
|  |  |  |
|  | | |



