Dataset_Z_Exploration

February 25, 2021

0.1 Dataset Prueba 1 - Tesis Javier-Uriel

0.1.1 Importamos algunas librerías que nos serán útiles más adelante

[1]:
```
import os
import time
import random

import pandas as pd # for dataframe operations.
import numpy as np #for linear algebra operations.
import seaborn as sns # data visualization library
import matplotlib.pyplot as plt # for plotting

from scipy.fftpack import fft, fftfreq

from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.stattools import pacf

pd.set_option('display.max_columns', None) #Para mostrar todas las columnas
random.seed(1)
```

0.1.2 Leemos el Dataset

[2]:
```
#Dataset solo movimientos en Z
rpm_list = ['RPM0', 'RPM1', 'RPM2', 'RPM3']
# states_list_org = ["vz","az", "uvz",
#                    "p", "q",
#                    "wp", "wq",
#                    "ap", "aq"]
states_list_org = ["vz","az", "uvz"]
states_list_min = ["vz","az", "uvz"]
dataset_name = "Dataset_Z1"
directory = "../logs/Datasets/"+dataset_name
ORDER = 3
dfs = []
states_list=states_list_org.copy()
```

0.1.3 Corregir la salida

El estado que entrega Pybullet de RPMs es la salida anterior, en este dataset se tomará RPMs como la salida actual. Si el primer elemento de RPMs es 0, es necesario hacer el shift

[4]:
```
for filename in os.listdir(directory):
    if not filename.endswith(".csv"):
        continue
    df = pd.read_csv(os.path.join(directory, filename))
    if any(df['z']<=1): #Eliminar si el dron se cae
        print(filename)
    else:
        if any(df[rpm_list].loc[0]==0): #Desplazar los estados de RPM si es␣
 ↪necesario
            df[rpm_list] = df[rpm_list].shift(periods=-1)
            df = df.dropna()
            df.to_csv(os.path.join(directory, filename), index=False)

        a = []
        ## Desplazamos estados anteriores
        for column in states_list:
            for n in range(1,ORDER+1):
                df[column+str(n)] = df[column].shift(periods=n, fill_value=0)
                a.append(column+str(n))
        dfs.append(df)
states_list+=a

dataset = pd.concat(dfs)
dataset.describe()
```

[4]:

	timestamps	x	y	z	Q1	Q2	\
count	4.463814e+06	4463814.0	4463814.0	4.463814e+06	4463814.0	4463814.0	
mean	4.999583e+01	0.0	0.0	3.275286e+01	0.0	0.0	
std	2.886631e+01	0.0	0.0	1.841391e+01	0.0	0.0	
min	0.000000e+00	0.0	0.0	3.295215e+00	0.0	0.0	
25%	2.499583e+01	0.0	0.0	2.462073e+01	0.0	0.0	
50%	4.999583e+01	0.0	0.0	2.562761e+01	0.0	0.0	
75%	7.499583e+01	0.0	0.0	3.432266e+01	0.0	0.0	
max	9.999167e+01	0.0	0.0	2.019521e+02	0.0	0.0	

	Q3	Q4	p	q	r	vx	\
count	4463814.0	4463814.0	4463814.0	4463814.0	4463814.0	4463814.0	
mean	0.0	1.0	0.0	0.0	0.0	0.0	
std	0.0	0.0	0.0	0.0	0.0	0.0	
min	0.0	1.0	0.0	-0.0	0.0	0.0	
25%	0.0	1.0	0.0	-0.0	0.0	0.0	
50%	0.0	1.0	0.0	-0.0	0.0	0.0	
75%	0.0	1.0	0.0	-0.0	0.0	0.0	

```
max          0.0        1.0        0.0       -0.0        0.0        0.0

              vy            vz         wp         wq         wr         ax  \
count  4463814.0  4.463814e+06  4463814.0  4463814.0  4463814.0  4463814.0
mean         0.0  1.584920e-01        0.0        0.0        0.0        0.0
std          0.0  1.417739e+00        0.0        0.0        0.0        0.0
min          0.0 -8.101928e+00        0.0        0.0        0.0        0.0
25%          0.0 -2.722415e-01        0.0        0.0        0.0        0.0
50%          0.0  1.811155e-07        0.0        0.0        0.0        0.0
75%          0.0  4.815810e-01        0.0        0.0        0.0        0.0
max          0.0  8.893730e+00        0.0        0.0        0.0        0.0

              ay            az         ap         aq         ar          RPM0  \
count  4463814.0  4.463814e+06  4463814.0  4463814.0  4463814.0  4.463814e+06
mean         0.0  1.273032e-03        0.0        0.0        0.0  1.440522e+04
std          0.0  2.153462e+00        0.0        0.0        0.0  1.585793e+03
min          0.0 -9.800000e+00        0.0        0.0        0.0  9.440300e+03
25%          0.0 -1.356594e-03        0.0        0.0        0.0  1.437433e+04
50%          0.0  0.000000e+00        0.0        0.0        0.0  1.447453e+04
75%          0.0  1.828387e-01        0.0        0.0        0.0  1.470020e+04
max          0.0  1.512624e+01        0.0        0.0        0.0  2.166645e+04

              RPM1          RPM2          RPM3         ux          uy  \
count  4.463814e+06  4.463814e+06  4.463814e+06  4463814.0  4463814.0
mean   1.440522e+04  1.440522e+04  1.440522e+04        0.0        0.0
std    1.585793e+03  1.585793e+03  1.585793e+03        0.0        0.0
min    9.440300e+03  9.440300e+03  9.440300e+03        0.0        0.0
25%    1.437433e+04  1.437433e+04  1.437433e+04        0.0        0.0
50%    1.447453e+04  1.447453e+04  1.447453e+04        0.0        0.0
75%    1.470020e+04  1.470020e+04  1.470020e+04        0.0        0.0
max    2.166645e+04  2.166645e+04  2.166645e+04        0.0        0.0

              uz        uvx        uvy           uvz         up         uq  \
count  4463814.0  4463814.0  4463814.0  4.463814e+06  4463814.0  4463814.0
mean        25.0        0.0        0.0  8.579590e-02        0.0        0.0
std          0.0        0.0        0.0  1.602419e+00        0.0        0.0
min         25.0        0.0        0.0 -9.083793e+00        0.0        0.0
25%         25.0        0.0        0.0 -3.294446e-01        0.0        0.0
50%         25.0        0.0        0.0  0.000000e+00        0.0        0.0
75%         25.0        0.0        0.0  4.401539e-01        0.0        0.0
max         25.0        0.0        0.0  9.083793e+00        0.0        0.0

              ur        uwp        uwq        uwr           vz1           vz2  \
count  4463814.0  4463814.0  4463814.0  4463814.0  4.463814e+06  4.463814e+06
mean         0.0        0.0        0.0        0.0  1.584867e-01  1.584814e-01
std          0.0        0.0        0.0        0.0  1.417718e+00  1.417698e+00
min          0.0        0.0        0.0        0.0 -8.101928e+00 -8.101928e+00
```

```
25%          0.0        0.0        0.0        0.0 -2.722415e-01 -2.722415e-01
50%          0.0        0.0        0.0        0.0  1.773966e-07  1.727081e-07
75%          0.0        0.0        0.0        0.0  4.815321e-01  4.814689e-01
max          0.0        0.0        0.0        0.0  8.893730e+00  8.893730e+00

                vz3           az1           az2           az3           uvz1  \
count  4.463814e+06  4.463814e+06  4.463814e+06  4.463814e+06  4.463814e+06
mean   1.584760e-01  1.277244e-03  1.281459e-03  1.285675e-03  8.582277e-02
std    1.417678e+00  2.153439e+00  2.153416e+00  2.153393e+00  1.602379e+00
min   -8.101928e+00 -9.800000e+00 -9.800000e+00 -9.800000e+00 -9.083793e+00
25%   -2.722397e-01 -1.354283e-03 -1.353126e-03 -1.350632e-03 -3.293318e-01
50%    1.680023e-07  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
75%    4.814005e-01  1.828131e-01  1.827909e-01  1.827674e-01  4.401539e-01
max    8.893730e+00  1.512624e+01  1.512624e+01  1.512624e+01  9.083793e+00

               uvz2          uvz3
count  4.463814e+06  4.463814e+06
mean   8.584965e-02  8.587652e-02
std    1.602339e+00  1.602299e+00
min   -9.083793e+00 -9.083793e+00
25%   -3.292443e-01 -3.291735e-01
50%    0.000000e+00  0.000000e+00
75%    4.401539e-01  4.401334e-01
max    9.083793e+00  9.083793e+00
```

0.1.4 Estados repetidos

En este caso se eliminan estados repetidos y estados que se encuentren en estado transitorio mientras el dron despega o se estabiliza antes de introducir la señal de control.

[5]:
```
shape_b4 = dataset.drop(["timestamps"], axis=1).shape
shape_drop= dataset.drop(["timestamps"], axis=1).drop_duplicates().shape
print(f'shape (b4 drop) = {shape_b4}')
print(f'shape = {shape_drop}')
print(f'len (b4 drop) - len = {shape_b4[0]-shape_drop[0]}')
```

```
shape (b4 drop) = (4463814, 47)
shape = (4279971, 47)
len (b4 drop) - len = 183843
```

[6]:
```
states = dataset.drop(["timestamps"], axis=1).drop_duplicates()[states_list]
print(f'columns = {states.columns}')
print(f'shape = {states.shape}')
states.head()
```

```
columns = Index(['vz', 'az', 'uvz', 'vz1', 'vz2', 'vz3', 'az1', 'az2', 'az3',
'uvz1',
       'uvz2', 'uvz3'],
```

```
      dtype='object')
shape = (4279971, 12)
```

[6]:

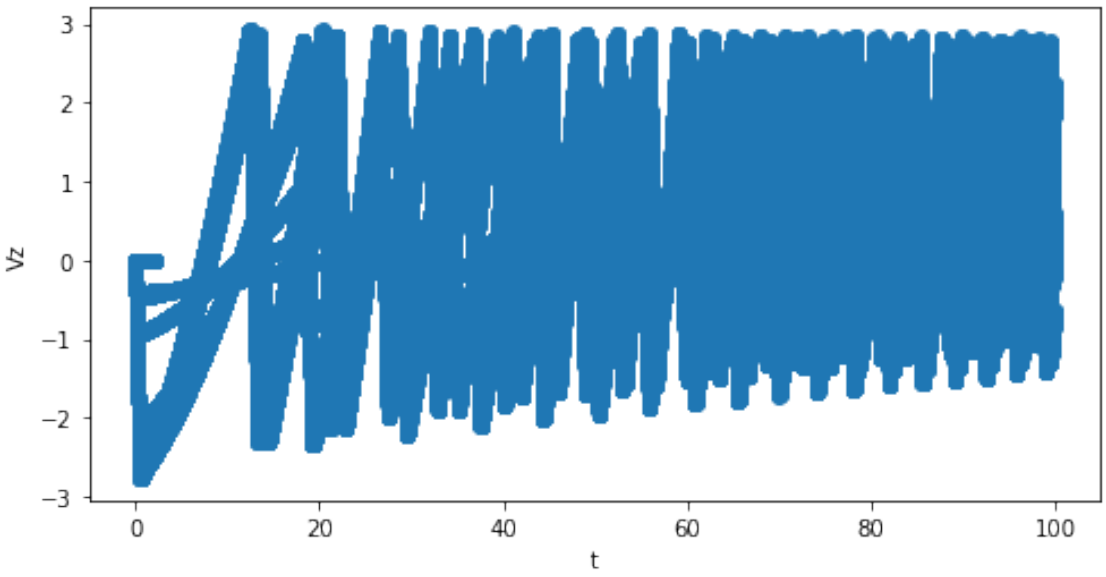
	vz	az	uvz	vz1	vz2	vz3	az1	az2	\
0	-0.040833	-9.800000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	
1	-0.037676	0.757873	0.0	-0.040833	0.000000	0.000000	-9.800000	0.000000	
2	-0.034518	0.757737	0.0	-0.037676	-0.040833	0.000000	0.757873	-9.800000	
3	-0.031362	0.757601	0.0	-0.034518	-0.037676	-0.040833	0.757737	0.757873	
4	-0.028206	0.757467	0.0	-0.031362	-0.034518	-0.037676	0.757601	0.757737	

	az3	uvz1	uvz2	uvz3
0	0.000000	0.0	0.0	0.0
1	0.000000	0.0	0.0	0.0
2	0.000000	0.0	0.0	0.0
3	-9.800000	0.0	0.0	0.0
4	0.757873	0.0	0.0	0.0

[7]:
```
states_duplicates = dataset[dataset.duplicated(keep='last')]
states_duplicates = states_duplicates.dropna()
```

[8]:
```
fig = plt.figure(figsize=(8, 4))
t = states_duplicates['timestamps']
y = states_duplicates['vz']
#y_ref = states_duplicates['uvz']
plt.scatter(t, y)
#plt.scatter(t, y_ref)
plt.ylabel('Vz')
plt.xlabel('t')
```

[8]:
```
Text(0.5, 0, 't')
```



[9]:
```
#Eliminar del dataset los estados repetidos entre 20 y 25 segundos
# for filename in os.listdir(directory):
#     if not filename.endswith(".csv"):
#         continue
#     df = pd.read_csv(os.path.join(directory, filename))
#     df = df[(df['timestamps']>20) & (df['timestamps']<25)]
#     df = pd.concat([df, states_duplicates])
#     df = df.reset_index(drop=True)
#     df_gpby = df.groupby(list(df.columns))
#     idx = [x[0] for x in df_gpby.groups.values() if len(x) > 1]
#     if len(idx)>1:
#         print(filename)
```

[10]:
```
df = pd.read_csv(os.path.join(directory, filename))
df = df[df['timestamps']>5]
df.head()
```

[10]:

	timestamps	x	y	z	Q1	Q2	Q3	Q4	p	q	r	vx
1201	5.004167	0.0	0.0	24.932919	0.0	0.0	0.0	1.0	0.0	-0.0	0.0	0.0
1202	5.008333	0.0	0.0	24.931309	0.0	0.0	0.0	1.0	0.0	-0.0	0.0	0.0
1203	5.012500	0.0	0.0	24.929685	0.0	0.0	0.0	1.0	0.0	-0.0	0.0	0.0
1204	5.016667	0.0	0.0	24.928048	0.0	0.0	0.0	1.0	0.0	-0.0	0.0	0.0
1205	5.020833	0.0	0.0	24.926396	0.0	0.0	0.0	1.0	0.0	-0.0	0.0	0.0

	vy	vz	wp	wq	wr	ax	ay	az	ap	aq	ar
1201	0.0	-0.383138	0.0	0.0	0.0	0.0	0.0	-0.790783	0.0	0.0	0.0
1202	0.0	-0.386432	0.0	0.0	0.0	0.0	0.0	-0.790551	0.0	0.0	0.0
1203	0.0	-0.389725	0.0	0.0	0.0	0.0	0.0	-0.790318	0.0	0.0	0.0
1204	0.0	-0.393017	0.0	0.0	0.0	0.0	0.0	-0.790084	0.0	0.0	0.0
1205	0.0	-0.396308	0.0	0.0	0.0	0.0	0.0	-0.789849	0.0	0.0	0.0

	RPM0	RPM1	RPM2	RPM3	ux	uy	uz
1201	13856.257814	13856.257814	13856.257814	13856.257814	0.0	0.0	25.0
1202	13856.257814	13856.257814	13856.257814	13856.257814	0.0	0.0	25.0
1203	13856.257814	13856.257814	13856.257814	13856.257814	0.0	0.0	25.0
1204	13856.257814	13856.257814	13856.257814	13856.257814	0.0	0.0	25.0
1205	14253.918732	14253.918732	14253.918732	14253.918732	0.0	0.0	25.0

	uvx	uvy	uvz	up	uq	ur	uwp	uwq	uwr
1201	0.0	0.0	-0.411884	0.0	0.0	0.0	0.0	0.0	0.0
1202	0.0	0.0	-0.411884	0.0	0.0	0.0	0.0	0.0	0.0
1203	0.0	0.0	-0.411884	0.0	0.0	0.0	0.0	0.0	0.0
1204	0.0	0.0	-0.411884	0.0	0.0	0.0	0.0	0.0	0.0
1205	0.0	0.0	-0.390044	0.0	0.0	0.0	0.0	0.0	0.0

0.1.5 Se grafican los datos

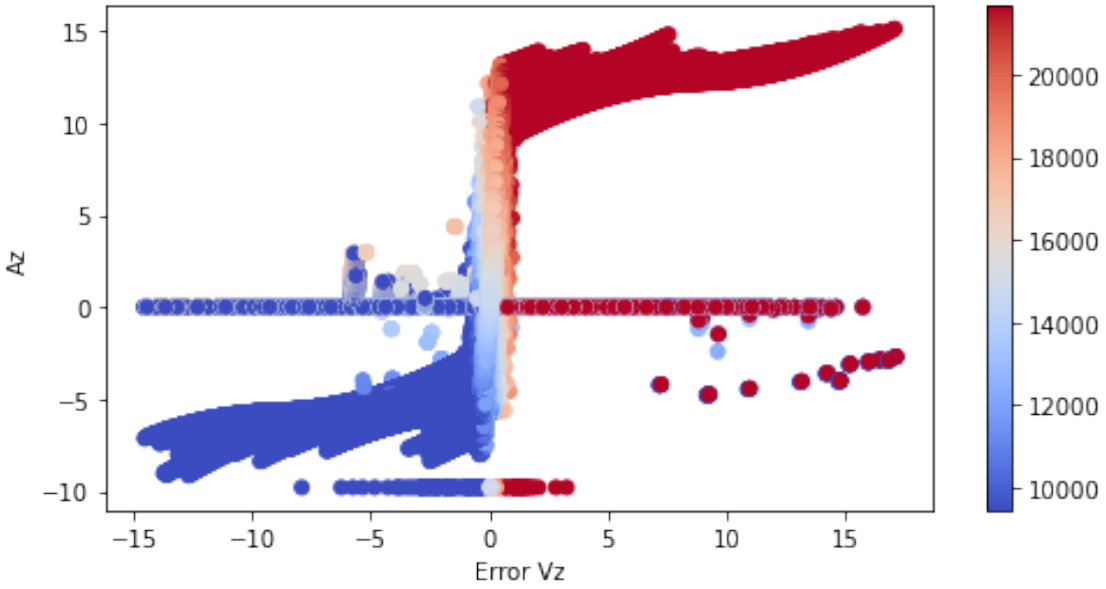
Se grafica un histograma de cada una de las propiedades los datos analizados individualmente por columnas, en el cual se observa que todos tienen distribuciones altamente apuntadas (curosis) y en algunos casos bimodales, pero de cualquier manera, no son uniformes

[11]:
```
n_bins = 50
#_ = dataset.hist(bins=n_bins, figsize=(30,30))
```

0.1.6 Análisis de estados

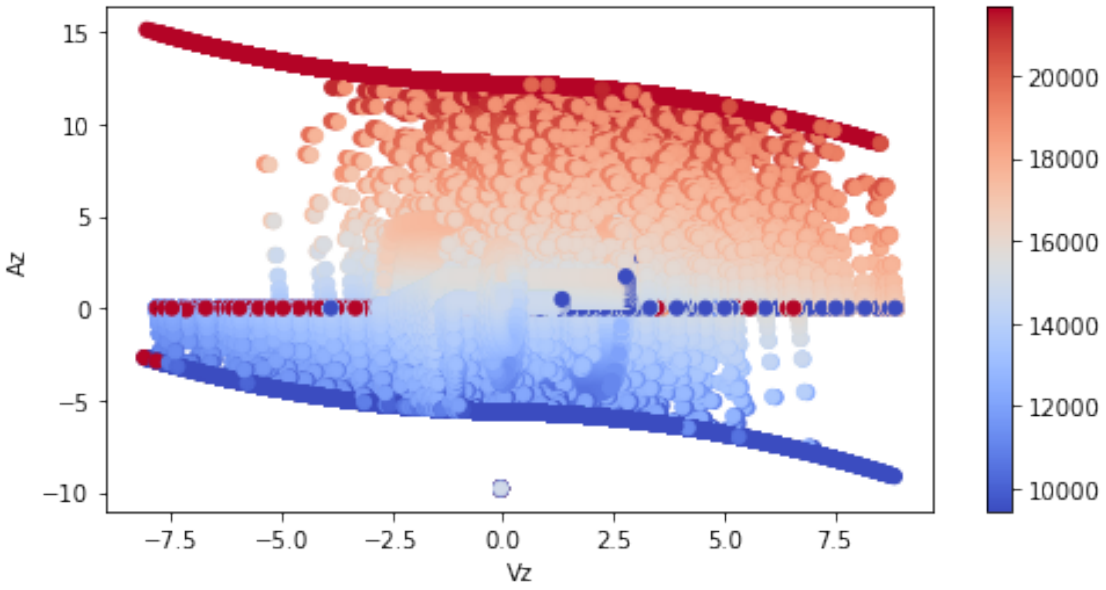
[12]:
```
fig = plt.figure(figsize=(8, 4))
x = dataset['uvz']-dataset['vz']
y = dataset['az']
c = dataset['RPMO']
plt.scatter(x, y, c=c, cmap='coolwarm')
plt.colorbar()
plt.ylabel('Az')
plt.xlabel('Error Vz')
```

[12]:
```
Text(0.5, 0, 'Error Vz')
```



[13]:
```
fig = plt.figure(figsize=(8, 4))
x = dataset['vz']
plt.scatter(x, y, c=c, cmap='coolwarm')
plt.colorbar()
plt.ylabel('Az')
plt.xlabel('Vz')
```

[13]: Text(0.5, 0, 'Vz')



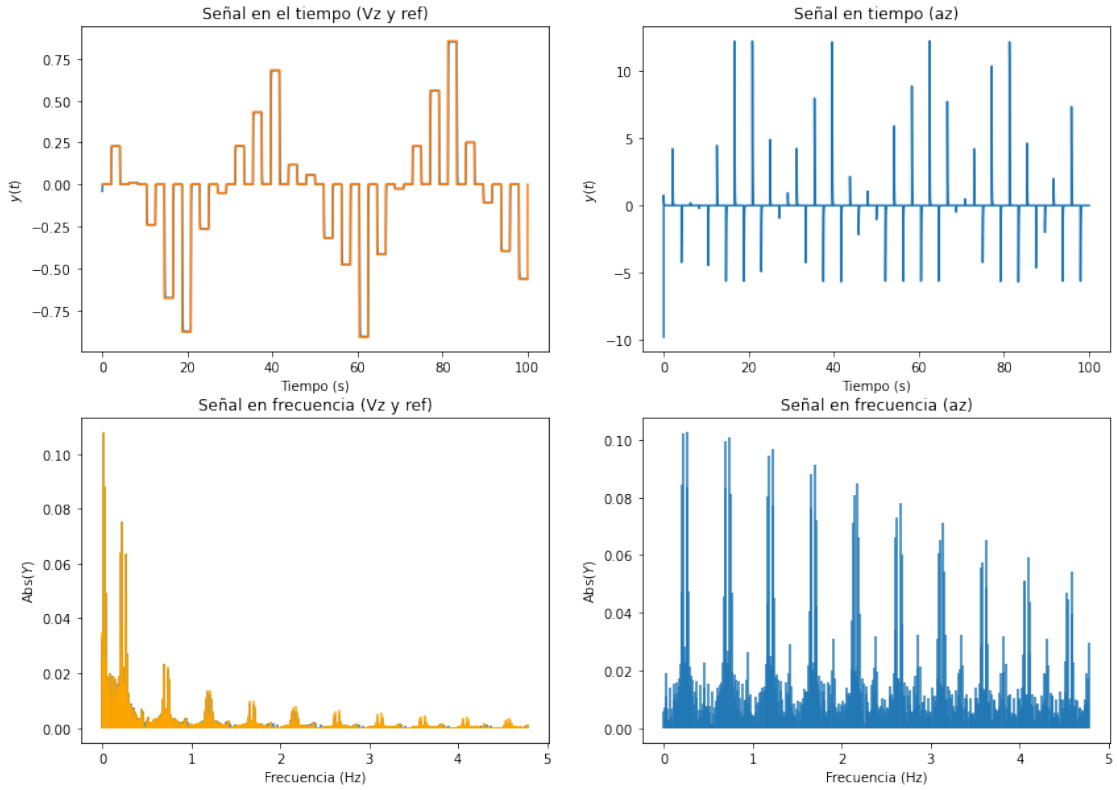
0.1.7 Análisis de Fourier

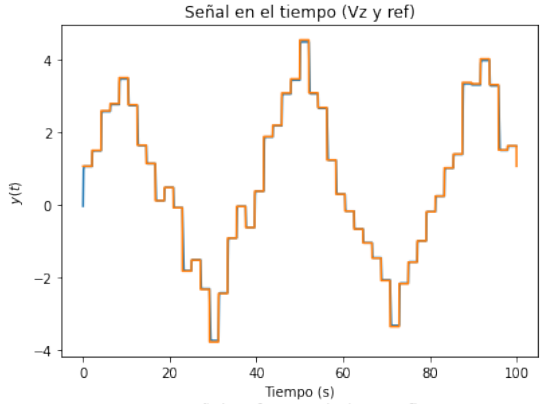
Gráfica de algunas señales

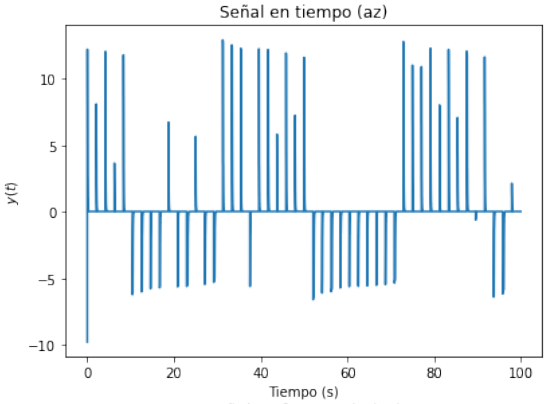
[14]:
```
def plot_fourier(df, states=['vz', 'uvz', 'az']):
    dt = df['timestamps'][1]-df['timestamps'][0]
    n = len(df['timestamps'])
    Y = fft(df[states[0]].to_numpy()) / n # Transformada normalizada
    Y_ref = fft(df[states[1]].to_numpy()) / n
    frq = fftfreq(n, dt)
    fig = plt.figure(figsize=(14, 10))
    ax1 = fig.add_subplot(221)
    ax1.plot(df['timestamps'], df[states[0]], df['timestamps'], df[states[1]])
    ax1.set_xlabel('Tiempo (s)')
    ax1.set_ylabel('$y(t)$')
    ax1.set_title('Señal en el tiempo (Vz y ref)')
    ax2 = fig.add_subplot(223)
    ax2.set_title('Señal en frecuencia (Vz y ref)')
    ax2.vlines(frq[0:int(n/50)], 0, abs(Y[0:int(n/50)]))
    ax2.vlines(frq[0:int(n/50)], 0, abs(Y_ref[0:int(n/50)]),  color='orange')
    plt.xlabel('Frecuencia (Hz)')
    plt.ylabel('Abs($Y$)')

    Y = fft(df[states[2]].to_numpy()) / n # Transformada normalizada
    ax1 = fig.add_subplot(222)
    ax1.plot(df['timestamps'], df[states[2]])
```

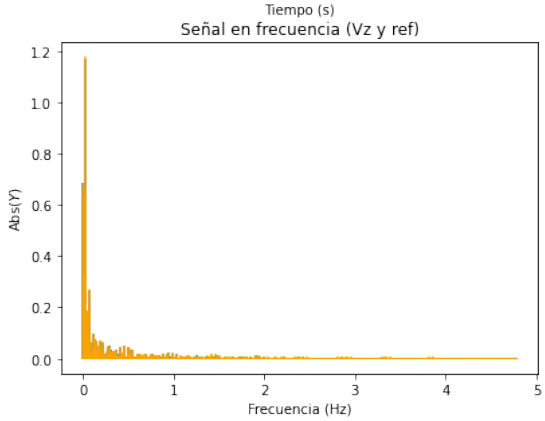
```
    ax1.set_xlabel('Tiempo (s)')
    ax1.set_ylabel('$y(t)$')
    ax1.set_title('Señal en tiempo (az)')
    ax2 = fig.add_subplot(224)
    ax2.set_title('Señal en frecuencia (az)')
    ax2.vlines(frq[0:int(n/50)], 0, abs(Y[0:int(n/50)]))
    plt.xlabel('Frecuencia (Hz)')
    plt.ylabel('Abs($Y$)')
```

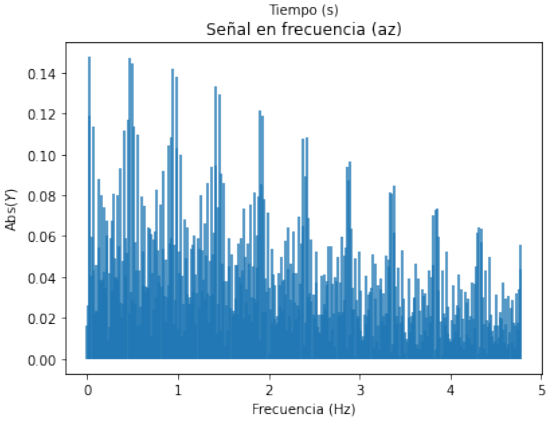
[15]:
```
for df in random.choices(dfs, k = 3):
    plot_fourier(df)
```

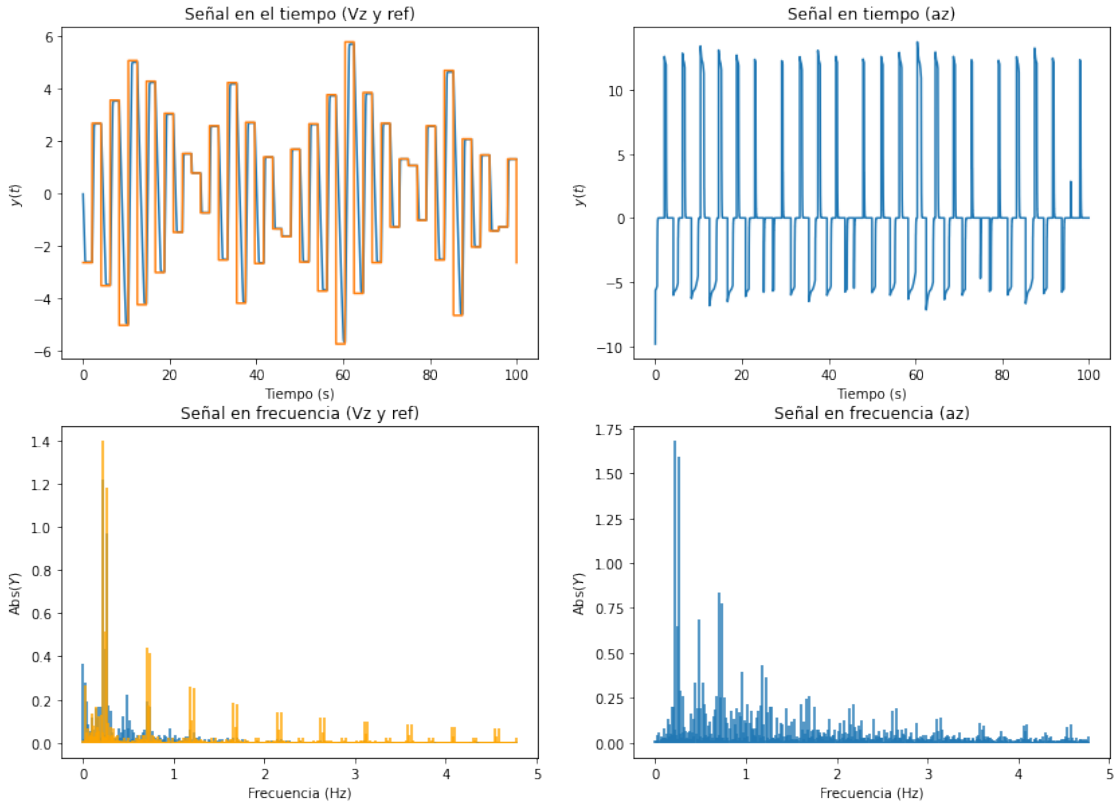



Señal en el tiempo (Vz y ref)
y(t)
Tiempo (s)


Señal en tiempo (az)
y(t)
Tiempo (s)


Señal en frecuencia (Vz y ref)
Abs(Y)
Frecuencia (Hz)


Señal en frecuencia (az)
Abs(Y)
Frecuencia (Hz)



Histograma

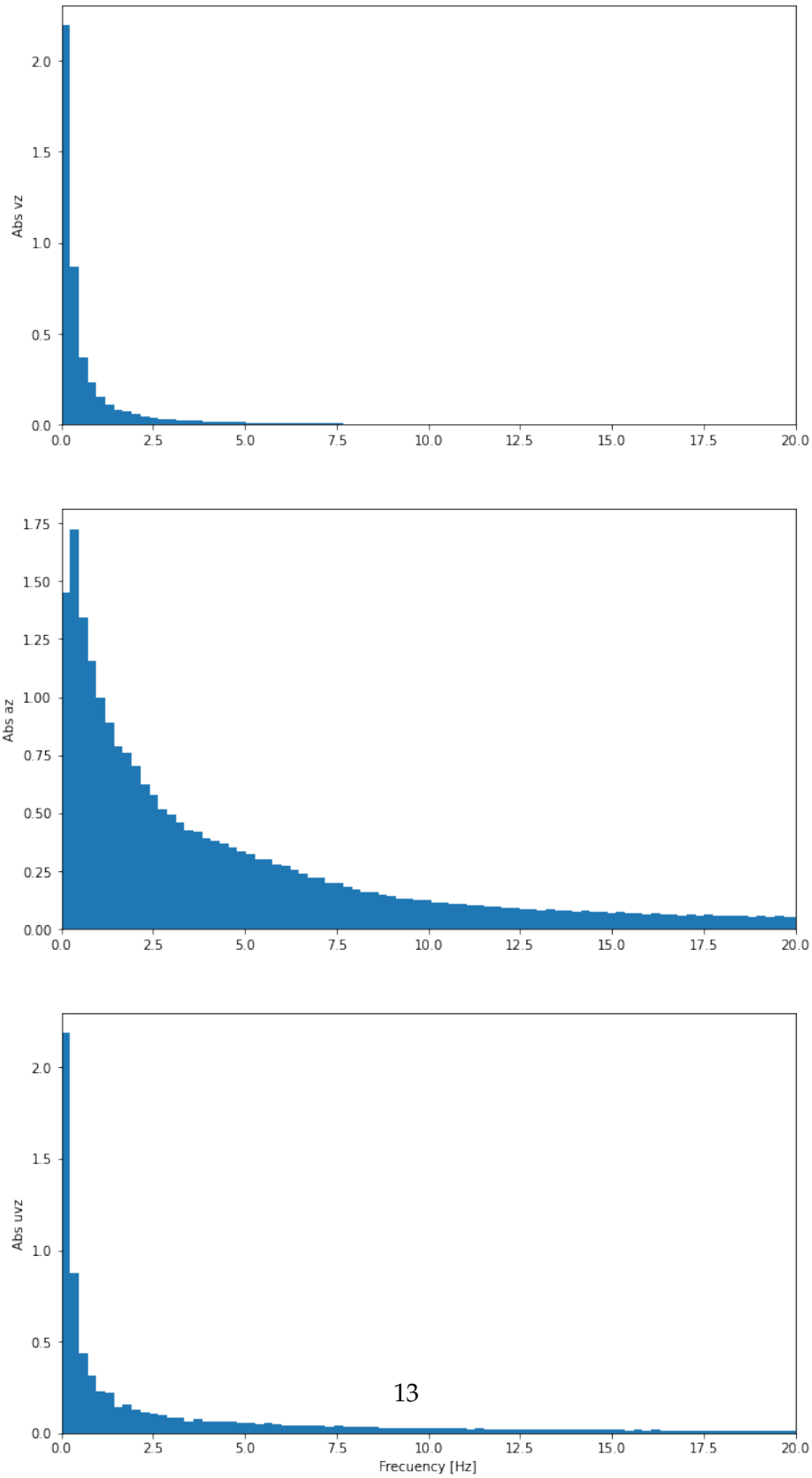
[16]:
```
Fourier = []
for i, df in enumerate(dfs):
    dt = df['timestamps'][1]-df['timestamps'][0]
    n = len(df['timestamps'])
    Fourier.append({})
    for state in states_list_org:
        Fourier[i][state]={}
        Fourier[i][state]['Y'] = abs(fft(df[state].to_numpy())/n)[0:int(n/2)] #␣
 ↪Transformada normalizada
        Fourier[i][state]['X'] = fftfreq(n, dt)[0:int(n/2)]
```

[17]:
```
F = {}
for state in states_list_org:
    F[state]={}
    F[state]['X'] = []
    F[state]['Y'] = []
    for f in Fourier:
        F[state]['X'] = np.concatenate([F[state]['X'], f[state]['X']])
        F[state]['Y'] = np.concatenate([F[state]['Y'], f[state]['Y']])
```

[18]:
```
fig, axs = plt.subplots(len(states_list_org), 1, figsize=(10, 20))
fig.suptitle('Fourier Transform Histogram per State')
for i, state in enumerate(states_list_org):
    axs[i].hist(F[state]['X'], bins=10*n_bins, weights=((F[state]['Y']+1e-7)/
 ↪len(Fourier)))
    axs[i].set_ylabel(f'Abs {state}')
    axs[i].set_xlim(0, 20)
axs[i].set_xlabel('Frecuency [Hz]')
```

[18]:
```
Text(0.5, 0, 'Frecuency [Hz]')
```

Fourier Transform Histogram per State



0.1.8 Análisis de Características - Método Estático

[19]:
```
dataset.describe()
```

[19]:

	timestamps	x	y	z	Q1	Q2
count	4.463814e+06	4463814.0	4463814.0	4.463814e+06	4463814.0	4463814.0
mean	4.999583e+01	0.0	0.0	3.275286e+01	0.0	0.0
std	2.886631e+01	0.0	0.0	1.841391e+01	0.0	0.0
min	0.000000e+00	0.0	0.0	3.295215e+00	0.0	0.0
25%	2.499583e+01	0.0	0.0	2.462073e+01	0.0	0.0
50%	4.999583e+01	0.0	0.0	2.562761e+01	0.0	0.0
75%	7.499583e+01	0.0	0.0	3.432266e+01	0.0	0.0
max	9.999167e+01	0.0	0.0	2.019521e+02	0.0	0.0

	Q3	Q4	p	q	r	vx
count	4463814.0	4463814.0	4463814.0	4463814.0	4463814.0	4463814.0
mean	0.0	1.0	0.0	0.0	0.0	0.0
std	0.0	0.0	0.0	0.0	0.0	0.0
min	0.0	1.0	0.0	-0.0	0.0	0.0
25%	0.0	1.0	0.0	-0.0	0.0	0.0
50%	0.0	1.0	0.0	-0.0	0.0	0.0
75%	0.0	1.0	0.0	-0.0	0.0	0.0
max	0.0	1.0	0.0	-0.0	0.0	0.0

	vy	vz	wp	wq	wr	ax
count	4463814.0	4.463814e+06	4463814.0	4463814.0	4463814.0	4463814.0
mean	0.0	1.584920e-01	0.0	0.0	0.0	0.0
std	0.0	1.417739e+00	0.0	0.0	0.0	0.0
min	0.0	-8.101928e+00	0.0	0.0	0.0	0.0
25%	0.0	-2.722415e-01	0.0	0.0	0.0	0.0
50%	0.0	1.811155e-07	0.0	0.0	0.0	0.0
75%	0.0	4.815810e-01	0.0	0.0	0.0	0.0
max	0.0	8.893730e+00	0.0	0.0	0.0	0.0

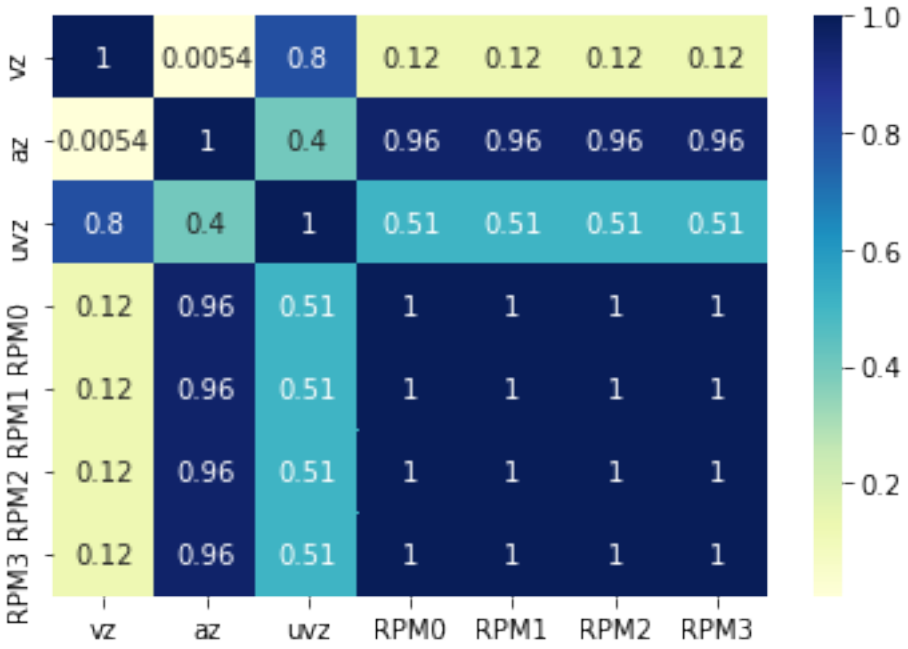
	ay	az	ap	aq	ar	RPM0
count	4463814.0	4.463814e+06	4463814.0	4463814.0	4463814.0	4.463814e+06
mean	0.0	1.273032e-03	0.0	0.0	0.0	1.440522e+04
std	0.0	2.153462e+00	0.0	0.0	0.0	1.585793e+03
min	0.0	-9.800000e+00	0.0	0.0	0.0	9.440300e+03
25%	0.0	-1.356594e-03	0.0	0.0	0.0	1.437433e+04
50%	0.0	0.000000e+00	0.0	0.0	0.0	1.447453e+04
75%	0.0	1.828387e-01	0.0	0.0	0.0	1.470020e+04
max	0.0	1.512624e+01	0.0	0.0	0.0	2.166645e+04

```
              RPM1          RPM2          RPM3         ux         uy  \
count  4.463814e+06  4.463814e+06  4.463814e+06  4463814.0  4463814.0
mean   1.440522e+04  1.440522e+04  1.440522e+04        0.0        0.0
std    1.585793e+03  1.585793e+03  1.585793e+03        0.0        0.0
min    9.440300e+03  9.440300e+03  9.440300e+03        0.0        0.0
25%    1.437433e+04  1.437433e+04  1.437433e+04        0.0        0.0
50%    1.447453e+04  1.447453e+04  1.447453e+04        0.0        0.0
75%    1.470020e+04  1.470020e+04  1.470020e+04        0.0        0.0
max    2.166645e+04  2.166645e+04  2.166645e+04        0.0        0.0

             uz        uvx        uvy           uvz         up         uq  \
count  4463814.0  4463814.0  4463814.0  4.463814e+06  4463814.0  4463814.0
mean        25.0        0.0        0.0  8.579590e-02        0.0        0.0
std          0.0        0.0        0.0  1.602419e+00        0.0        0.0
min         25.0        0.0        0.0 -9.083793e+00        0.0        0.0
25%         25.0        0.0        0.0 -3.294446e-01        0.0        0.0
50%         25.0        0.0        0.0  0.000000e+00        0.0        0.0
75%         25.0        0.0        0.0  4.401539e-01        0.0        0.0
max         25.0        0.0        0.0  9.083793e+00        0.0        0.0

             ur        uwp        uwq        uwr           vz1           vz2  \
count  4463814.0  4463814.0  4463814.0  4463814.0  4.463814e+06  4.463814e+06
mean         0.0        0.0        0.0        0.0  1.584867e-01  1.584814e-01
std          0.0        0.0        0.0        0.0  1.417718e+00  1.417698e+00
min          0.0        0.0        0.0        0.0 -8.101928e+00 -8.101928e+00
25%          0.0        0.0        0.0        0.0 -2.722415e-01 -2.722415e-01
50%          0.0        0.0        0.0        0.0  1.773966e-07  1.727081e-07
75%          0.0        0.0        0.0        0.0  4.815321e-01  4.814689e-01
max          0.0        0.0        0.0        0.0  8.893730e+00  8.893730e+00

              vz3           az1           az2           az3           uvz1  \
count  4.463814e+06  4.463814e+06  4.463814e+06  4.463814e+06  4.463814e+06
mean   1.584760e-01  1.277244e-03  1.281459e-03  1.285675e-03  8.582277e-02
std    1.417678e+00  2.153439e+00  2.153416e+00  2.153393e+00  1.602379e+00
min   -8.101928e+00 -9.800000e+00 -9.800000e+00 -9.800000e+00 -9.083793e+00
25%   -2.722397e-01 -1.354283e-03 -1.353126e-03 -1.350632e-03 -3.293318e-01
50%    1.680023e-07  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
75%    4.814005e-01  1.828131e-01  1.827909e-01  1.827674e-01  4.401539e-01
max    8.893730e+00  1.512624e+01  1.512624e+01  1.512624e+01  9.083793e+00

             uvz2          uvz3
count  4.463814e+06  4.463814e+06
mean   8.584965e-02  8.587652e-02
std    1.602339e+00  1.602299e+00
min   -9.083793e+00 -9.083793e+00
25%   -3.292443e-01 -3.291735e-01
50%    0.000000e+00  0.000000e+00
```

```
75%     4.401539e-01   4.401334e-01
max     9.083793e+00   9.083793e+00
```

Mapa de Correlación

[20]:
```
correlation = dataset[states_list_org + rpm_list].corr() #corr() method of␣
 ↪pandas library calculates correlation between columns of dataframe
sns.heatmap(correlation,cmap="YlGnBu",annot=True)
plt.show()
```



Análisis de Correlaciones

[21]:
```
# Comentado porque se demora mucho procesando
# for i in states_list_org:
#     sns.lmplot(x=i, y=rpm_list[0], data=dataset,line_kws={'color': 'red'})
#     text="Relation between RPM0 and " + i
#     plt.title(text)
#     plt.show()
```

[22]:
```
corr_df = pd.DataFrame()
for i in rpm_list:
    correlation = dataset.corr()[i] # convert series to dataframe so it can be␣
 ↪sorted
    correlation_df = pd.DataFrame(correlation) # correct column label from␣
 ↪Points to correlation
    correlation_df.columns = [f"Correlation_{i}"] # sort correlation
```

```
    corr_df = pd.concat([correlation_df, corr_df], axis=1)
corr_df = corr_df.dropna(how='all')
corr_df = corr_df.sort_values(by=[f'Correlation_{rpm_list[0]}'], ascending=False)
corr_df.head(30)
```

[22]:

	Correlation_RPM3	Correlation_RPM2	Correlation_RPM1	\
RPM0	1.000000	1.000000	1.000000	
RPM1	1.000000	1.000000	1.000000	
RPM2	1.000000	1.000000	1.000000	
RPM3	1.000000	1.000000	1.000000	
az	0.963329	0.963329	0.963329	
az1	0.951930	0.951930	0.951930	
az2	0.940524	0.940524	0.940524	
az3	0.929113	0.929113	0.929113	
uvz	0.509812	0.509812	0.509812	
uvz1	0.509636	0.509636	0.509636	
uvz2	0.509459	0.509459	0.509459	
uvz3	0.509283	0.509283	0.509283	
vz	0.124640	0.124640	0.124640	
vz1	0.118545	0.118545	0.118545	
vz2	0.112522	0.112522	0.112522	
vz3	0.106571	0.106571	0.106571	
timestamps	-0.005450	-0.005450	-0.005450	
z	-0.045928	-0.045928	-0.045928	

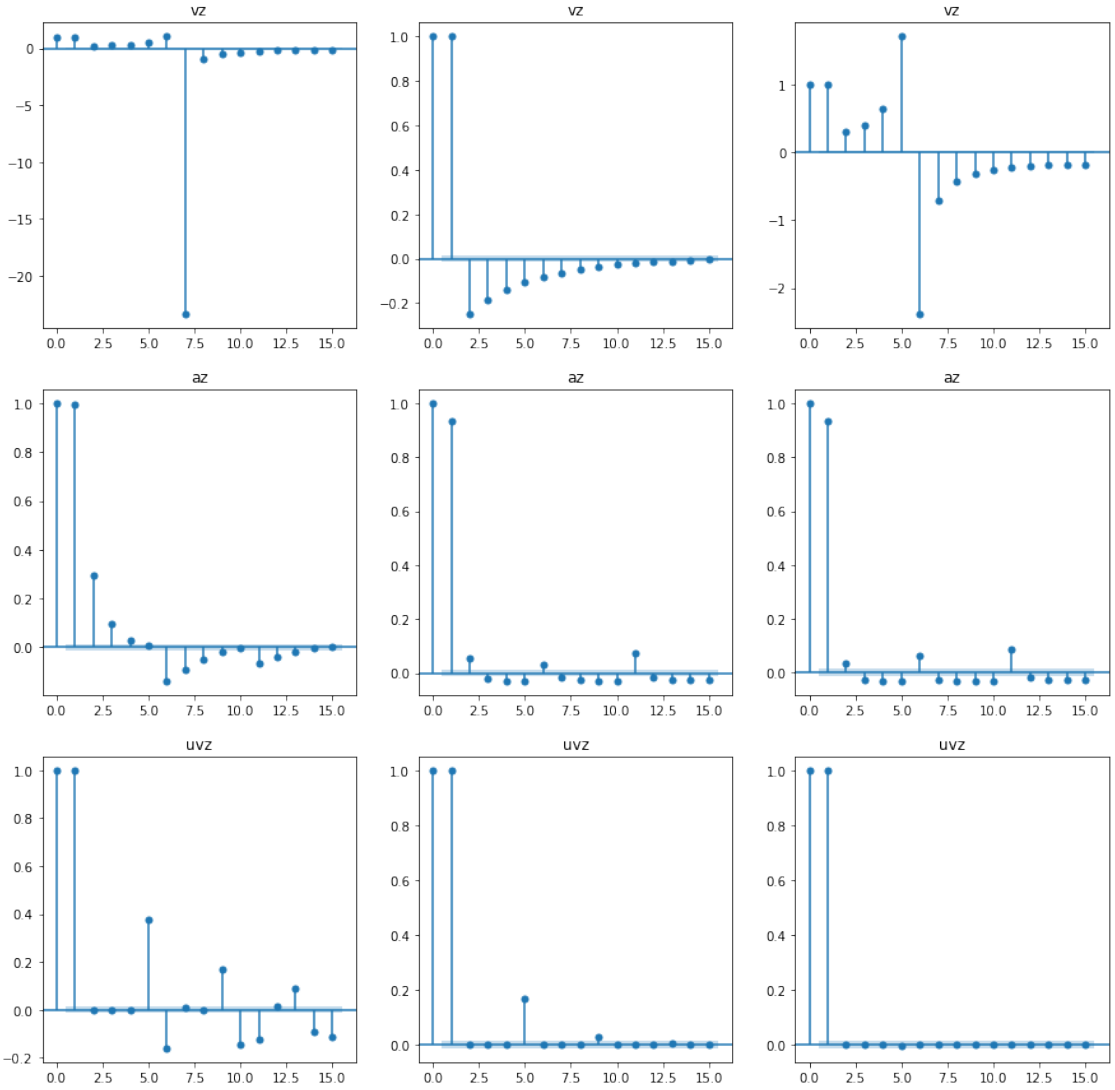
	Correlation_RPM0
RPM0	1.000000
RPM1	1.000000
RPM2	1.000000
RPM3	1.000000
az	0.963329
az1	0.951930
az2	0.940524
az3	0.929113
uvz	0.509812
uvz1	0.509636
uvz2	0.509459
uvz3	0.509283
vz	0.124640
vz1	0.118545
vz2	0.112522
vz3	0.106571
timestamps	-0.005450
z	-0.045928

0.1.9 Análisis de Características - Método Dinámico

Autocorrelación Parcial

[23]:
```
N_df = 3
nlags = 15
fig, axs = plt.subplots(N_df, len(states_list_min), figsize=(15, 15))
for k, df in enumerate(random.choices(dfs, k = N_df)):
    for j, i in enumerate(states_list_min):
        plot_pacf(df[i], lags=nlags, ax = axs[j, k])
        axs[j, k].set_title(i)
```

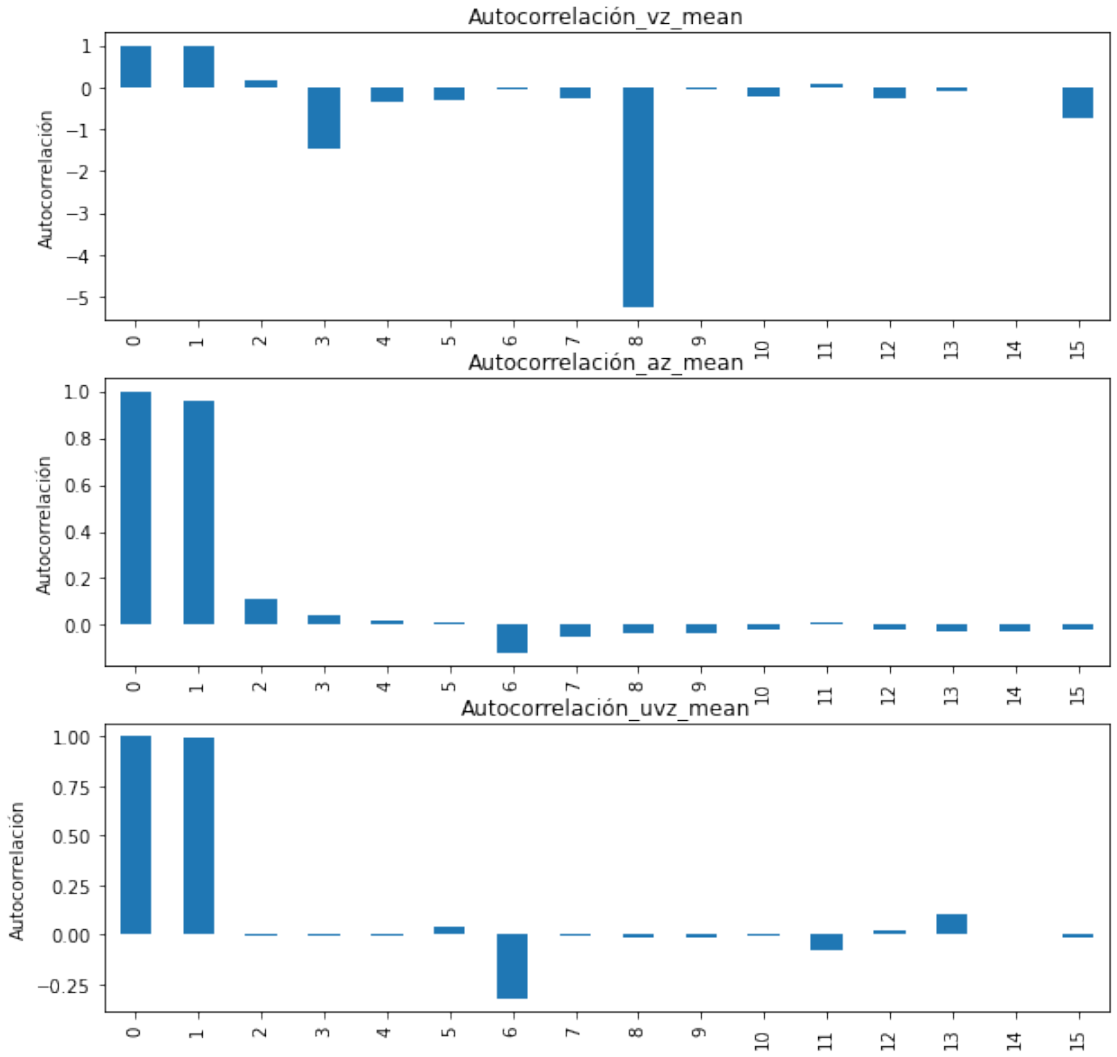
```
C:\Users\mrjar\.conda\envs\tesis\lib\site-
packages\statsmodels\regression\linear_model.py:1434: RuntimeWarning: invalid
value encountered in sqrt
  return rho, np.sqrt(sigmasq)
C:\Users\mrjar\.conda\envs\tesis\lib\site-
packages\statsmodels\regression\linear_model.py:1434: RuntimeWarning: invalid
value encountered in sqrt
  return rho, np.sqrt(sigmasq)
```



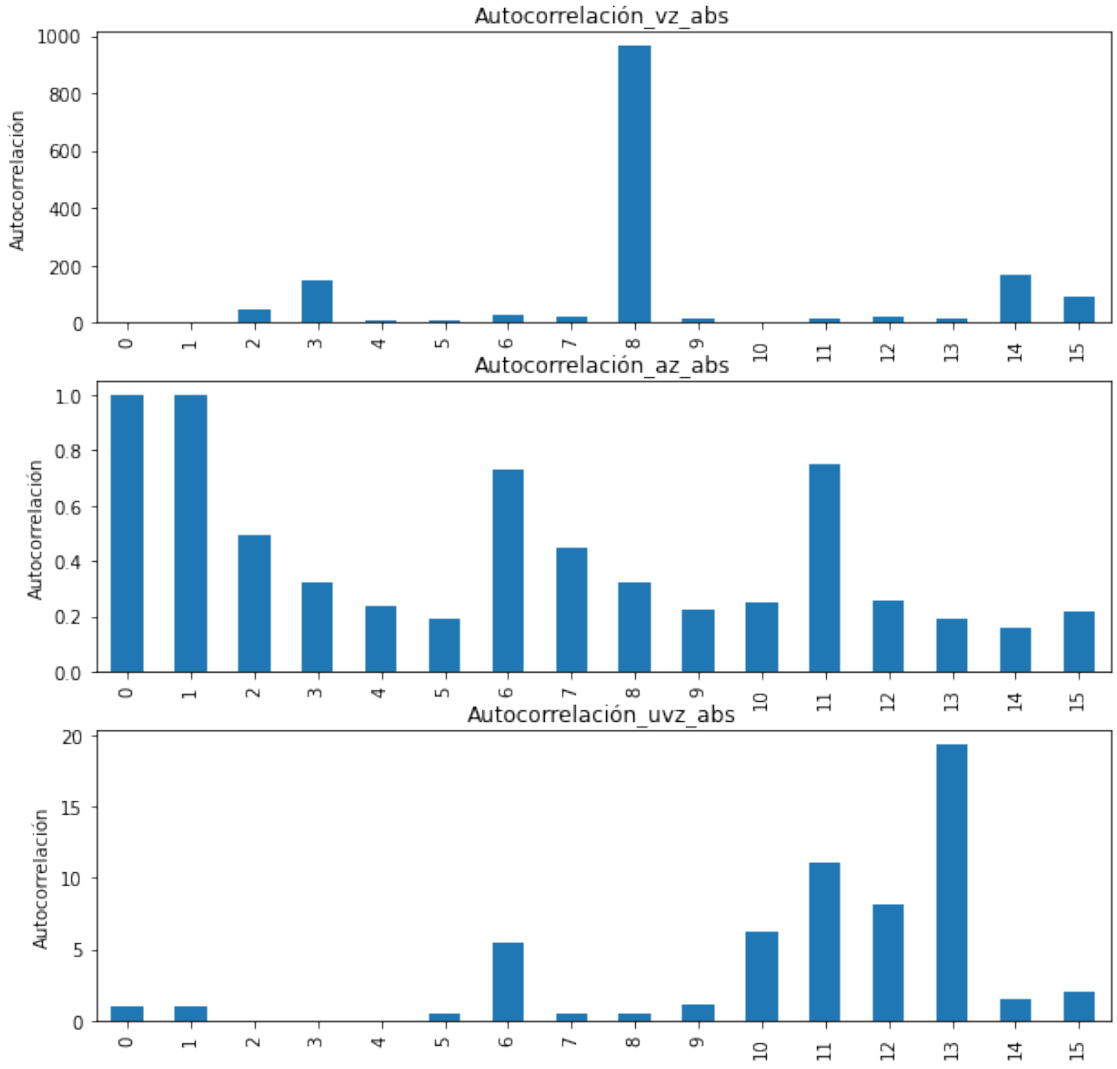
[24]:
```
pacf_df = [pd.DataFrame()]*len(states_list_min)
for k, df in enumerate(dfs):
    for j, i in enumerate(states_list_min):
        tmp = pd.DataFrame(pacf(df[i], nlags=nlags), columns=[str(k)])
        pacf_df[j] = pd.concat([pacf_df[j], tmp], axis=1)
pacf_df_dict = {}
for j, i in enumerate(states_list_min):
    pacf_df_dict[i] =  pd.DataFrame()
    pacf_df_dict[i]['mean'] = pacf_df[j].T.mean()
    pacf_df_dict[i]['min'] = pacf_df[j].T.min()
    pacf_df_dict[i]['max'] = pacf_df[j].T.max()
    pacf_df_dict[i]['abs'] = np.maximum(pacf_df[j].T.max(), abs(pacf_df[j].T.
 ↪min()))
```

```
C:\Users\mrjar\.conda\envs\tesis\lib\site-
packages\statsmodels\regression\linear_model.py:1434: RuntimeWarning: invalid
value encountered in sqrt
  return rho, np.sqrt(sigmasq)
```

[25]:
```
fig, axes = plt.subplots(nrows=len(states_list_min), ncols=1, figsize=(10, 10))
crt = 'mean'
for j, i in enumerate(states_list_min):
    pacf_df_dict[i][crt].plot(kind="bar", ax=axes[j])
    axes[j].set_ylabel('Autocorrelación')
    axes[j].set_title(f'Autocorrelación_{i}_{crt}')
```



[26]:
```
fig, axes = plt.subplots(nrows=len(states_list_min), ncols=1, figsize=(10, 10))
crt = 'abs'
for j, i in enumerate(states_list_min):
    pacf_df_dict[i][crt].plot(kind="bar", ax=axes[j])
    axes[j].set_ylabel('Autocorrelación')
    axes[j].set_title(f'Autocorrelación_{i}_{crt}')
```


Autocorrelación_vz_abs
Autocorrelación
1000
800
600
400
200
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Autocorrelación_az_abs
Autocorrelación
1.0
0.8
0.6
0.4
0.2
0.0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Autocorrelación_uvz_abs
Autocorrelación
20
15
10
5
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15