Autor: Diego Hinojosa Córdova

Fecha: 23-Nov-2020

Objetivo:

Determine cuál de las puntas de cohete es la más segura.

Suponiendo que todas las demás variables (velocidad del cohete antes del impacto, masa del cohete, etc.) son iguales, definimos que la punta más segura es **la que imparte la fuerza media y/o la fuerza máxima más bajas** al impactar contra un objeto.

Las dos cosas que debe determinar de manera concisa son:

- La fuerza máxima impartida por el cohete durante el impacto de cada punta.
- La fuerza media impartida por el cohete durante el impacto de cada punta.

Hay dos tareas secundarias asociadas con este objetivo.

1. **Filtrado**: para cada punta de cohete, calcule los perfiles de velocidad y aceleración a partir de los datos de posición.
2. Determine el tiempo total de impacto y las fuerzas impartidas

Las muestras de la posicion se tomaron a una velocidad de Velocidad de video: **6250 fotogramas / segundo**

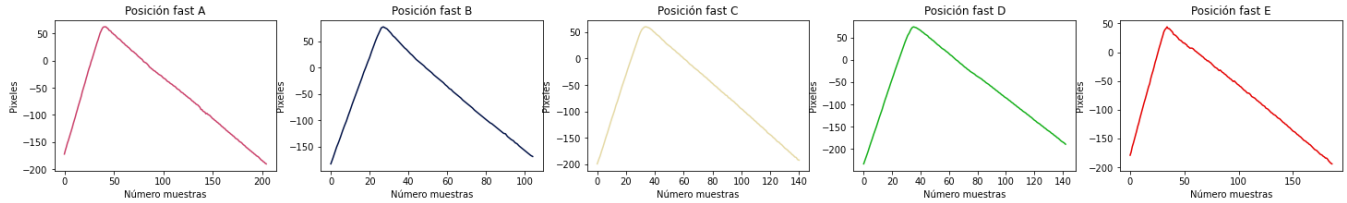
La calibracion de pixeles a dimensiones esta dada por Escala fisica: **47.0027 pixeles/pulgadas**

La prueba lenta se realizó a 15.5 m/s y la rápida a 28,4 m/s.

```
import pandas as pd
import numpy  as np
import psycopg2 as ps
import matplotlib.pyplot as plt
import seaborn as sns
```

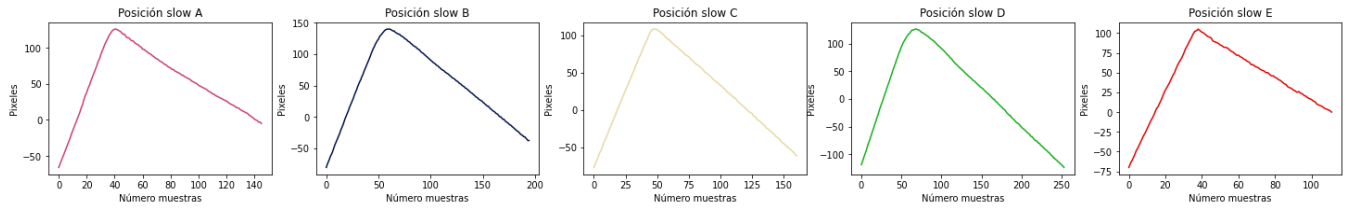
```
#Cargamos los datos
p_fast_a = pd.read_csv('https://raw.githubusercontent.com/dhinojosac/mit_beyond_iot/master/mo
p_fast_b = pd.read_csv('https://raw.githubusercontent.com/dhinojosac/mit_beyond_iot/master/mo
p_fast_c = pd.read_csv('https://raw.githubusercontent.com/dhinojosac/mit_beyond_iot/master/mo
p_fast_d = pd.read_csv('https://raw.githubusercontent.com/dhinojosac/mit_beyond_iot/master/mo
p_fast_e = pd.read_csv('https://raw.githubusercontent.com/dhinojosac/mit_beyond_iot/master/mo

p_slow_a = pd.read_csv('https://raw.githubusercontent.com/dhinojosac/mit_beyond_iot/master/mo
p_slow_b = pd.read_csv('https://raw.githubusercontent.com/dhinojosac/mit_beyond_iot/master/mo
p_slow_c = pd.read_csv('https://raw.githubusercontent.com/dhinojosac/mit_beyond_iot/master/mo
p_slow_d = pd.read_csv('https://raw.githubusercontent.com/dhinojosac/mit_beyond_iot/master/mo
p_slow_e = pd.read_csv('https://raw.githubusercontent.com/dhinojosac/mit_beyond_iot/master/mo
```

```
# Graficamos
#Graficamos las posiciones rapidas
fig = plt.figure(figsize=(25, 3))

ax1 = fig.add_subplot(1, 5, 1)
ax2 = fig.add_subplot(1, 5, 2)
ax3 = fig.add_subplot(1, 5, 3)
ax4 = fig.add_subplot(1, 5, 4)
ax5 = fig.add_subplot(1, 5, 5)

ax1.plot(p_fast_a, color='xkcd:dark pink')
ax2.plot(p_fast_b, color='xkcd:navy blue')
ax3.plot(p_fast_c, color='xkcd:beige')
ax4.plot(p_fast_d, color='xkcd:green')
ax5.plot(p_fast_e, color='xkcd:red')

ax1.set_title("Posición fast A")
ax2.set_title("Posición fast B")
ax3.set_title("Posición fast C")
ax4.set_title("Posición fast D")
ax5.set_title("Posición fast E")

# recorremos todos los gráficos para agregarles etiquetas a los ejes
for ax in [ax1, ax2, ax3, ax4, ax5]:
    ax.set_ylabel("Pixeles")
    ax.set_xlabel("Número muestras")

plt.show()
```



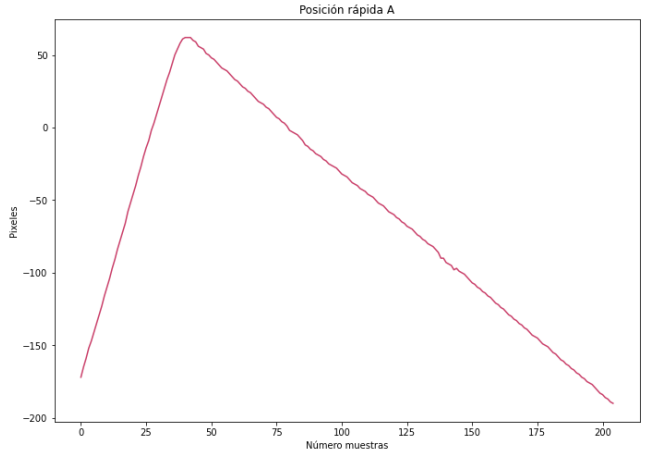

```
# Graficamos
#Graficamos las posiciones lentas
fig = plt.figure(figsize=(25, 3))

ax1 = fig.add_subplot(1, 5, 1)
ax2 = fig.add_subplot(1, 5, 2)
ax3 = fig.add_subplot(1, 5, 3)
ax4 = fig.add_subplot(1, 5, 4)
```

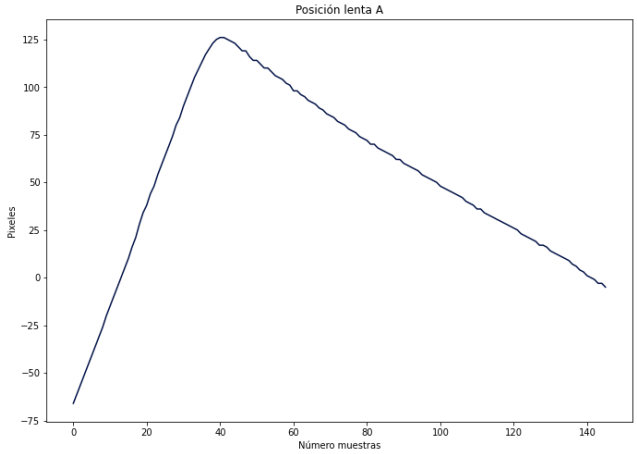
```
ax5 = fig.add_subplot(1, 5, 5)

ax1.plot(p_slow_a, color='xkcd:dark pink')
ax2.plot(p_slow_b, color='xkcd:navy blue')
ax3.plot(p_slow_c, color='xkcd:beige')
ax4.plot(p_slow_d, color='xkcd:green')
ax5.plot(p_slow_e, color='xkcd:red')

ax1.set_title("Posición slow A")
ax2.set_title("Posición slow B")
ax3.set_title("Posición slow C")
ax4.set_title("Posición slow D")
ax5.set_title("Posición slow E")

# recorremos todos los gráficos para agregarles etiquetas a los ejes
for ax in [ax1, ax2, ax3, ax4, ax5]:
    ax.set_ylabel("Pixeles")
    ax.set_xlabel("Número muestras")

plt.show()
```




```
# comparamos una rápida con una lenta
fig = plt.figure(figsize=(25, 8))

ax1 = fig.add_subplot(1, 2, 1)
ax2 = fig.add_subplot(1, 2, 2)

ax1.plot(p_fast_a, color='xkcd:dark pink')
ax2.plot(p_slow_a, color='xkcd:navy blue')

ax1.set_title("Posición rápida A")
ax2.set_title("Posición lenta A")

# recorremos todos los gráficos para agregarles etiquetas a los ejes
for ax in [ax1, ax2]:
    ax.set_ylabel("Pixeles")
    ax.set_xlabel("Número muestras")

plt.show()
```

```
plt.show()
```





Como sabemos, para calcular la velocidad debemos sacar la razon de cambio de la distancia en el tiempo.

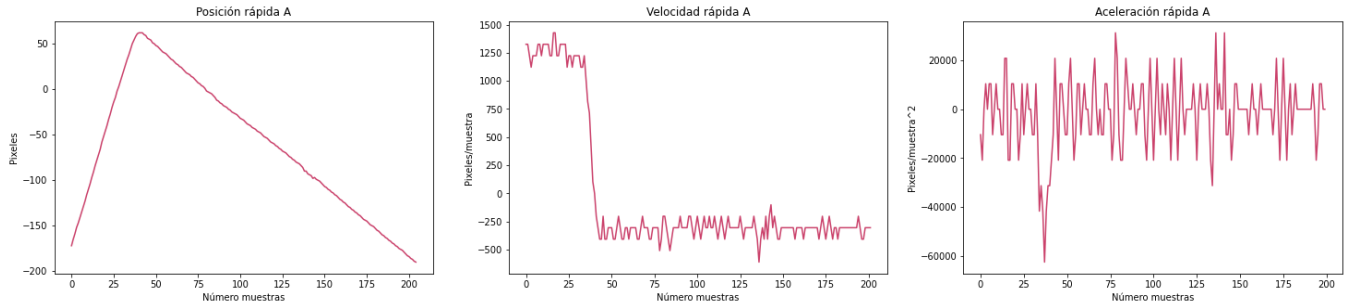
En estos ejemplos la distancia es equivalente a los pixeles y el tiempo al número de muestras.

Utilizando los datos antes dados:

- Velocidad de video: 6250 fotogramas / segundo
- La calibracion de pixeles: 47.0027 pixeles/pulgada

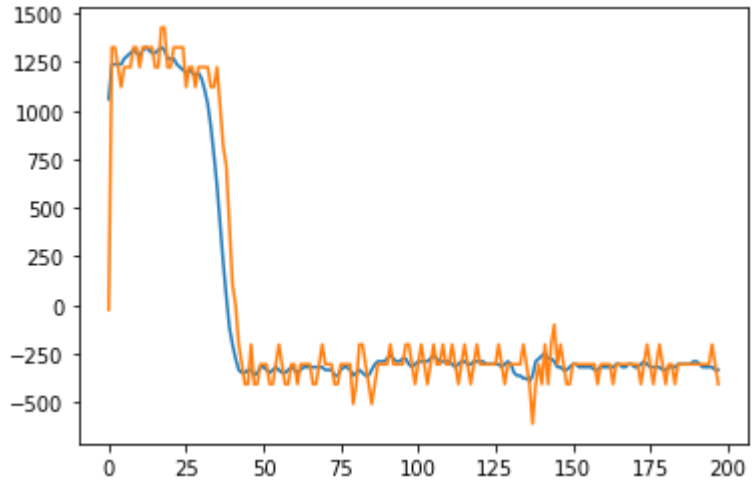
```
def numerical_derivative(y):
  x = np.zeros((y.shape[0],1))
  x[:,0] = np.linspace(0,1 , y.shape[0])
  res = (np.roll(y, -1) - np.roll(y, 1)) / (np.roll(x, -1) - np.roll(x, 1))
  return res

y = p_fast_a


v = numerical_derivative(y)

y2 = v
```

```
a = numerical_derivative(y2)

# comparamos una rápida con una lenta
fig = plt.figure(figsize=(25, 5))

ax0 = fig.add_subplot(1, 3, 1)
ax1 = fig.add_subplot(1, 3, 2)
ax2 = fig.add_subplot(1, 3, 3)

ax0.plot(y, color='xkcd:dark pink')
ax1.plot(v[1:-2], color='xkcd:dark pink')
ax2.plot(a[2:-3], color='xkcd:dark pink')

ax0.set_title("Posición rápida A")
ax0.set_ylabel("Pixeles")
ax0.set_xlabel("Número muestras")

ax1.set_title("Velocidad rápida A")
ax1.set_ylabel("Pixeles/muestra")
ax1.set_xlabel("Número muestras")

ax2.set_title("Aceleración rápida A")
ax2.set_ylabel("Pixeles/muestra^2")
ax2.set_xlabel("Número muestras")

plt.show()
```



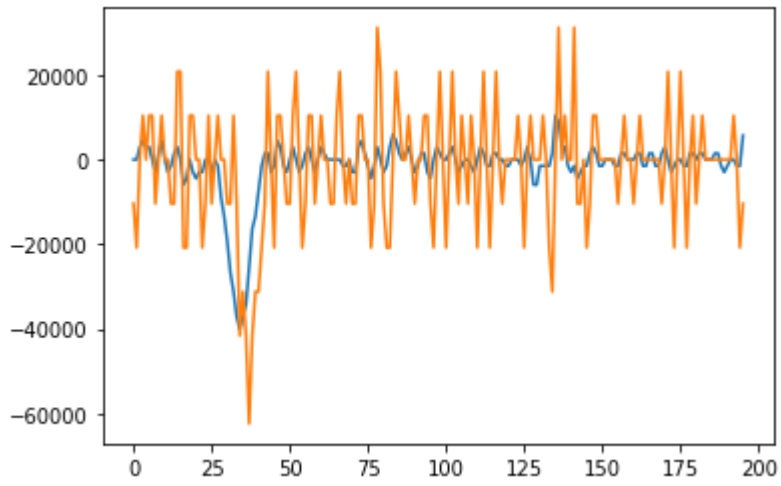

```
# Creamos un filtro de ventana movil, que lo que hace es sauvizar los datos
def smoothData(data, num):
  f = np.zeros(data.shape[0])

  for i in range(data.shape[0]-num):
```

```
    a=0
    for j in range(num):
      a+=data[i+j]
    f[i] = a/num
  return f[:-num]

num_w = 7
v_filter = smoothData(v,num_w)
plt.plot(v_filter)
plt.plot(v[:-num_w])
plt.show()
```



Posteriormente se obtiene la aceleración filtrada a partir de la velocidad sin filtrar

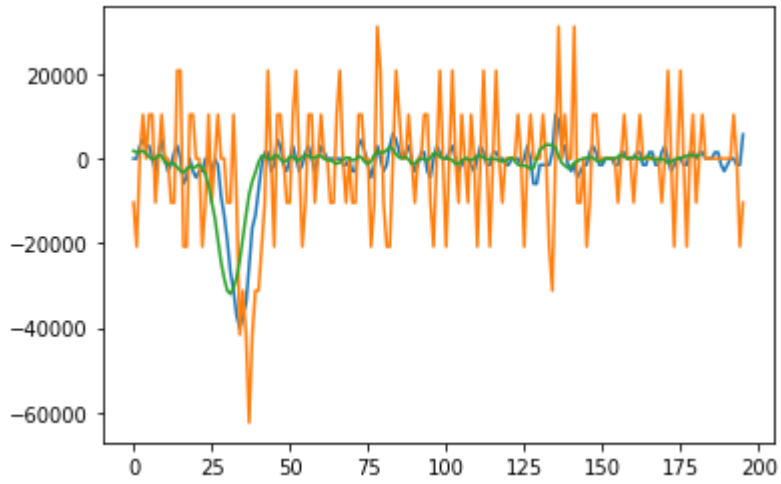
```
a_filter = smoothData(a,num_w)
plt.plot(a_filter[2:])
plt.plot(a[2:-num_w])
plt.show()
```



En el siguiente gráfico se obtiene a aceleración filtrada a partir de la velocidad filtrada.

```
print(v_filter.shape)
vff=np.zeros((v_filter.shape[0],1))
print(vff.shape)
vff[:,0] = v_filter
aff = numerical_derivative(vff)
afff = smoothData(aff,num_w)
plt.plot(a_filter[2:])
plt.plot(a[2:-num_w])
plt.plot(afff[2:-num_w])
plt.show()
```

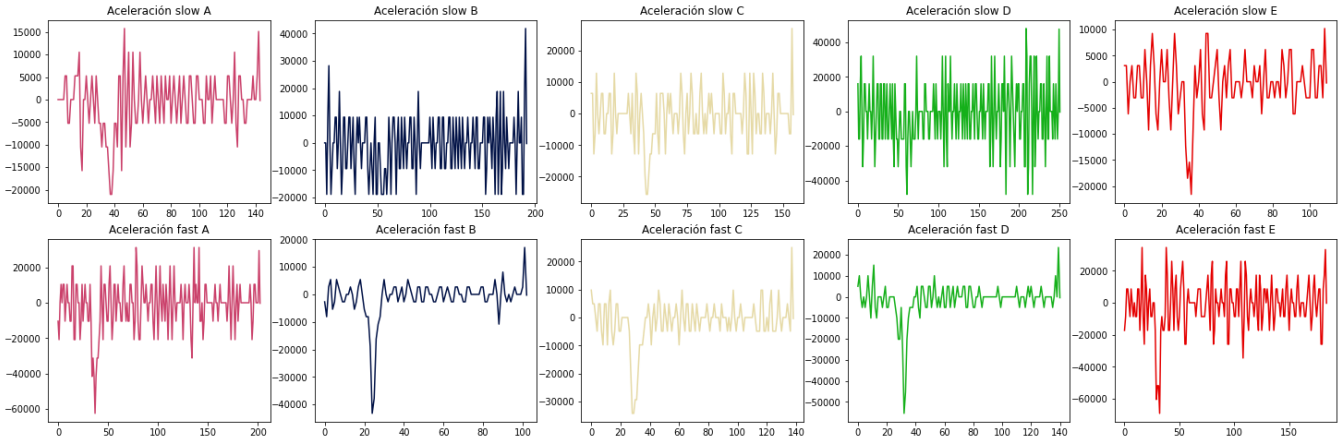
```
(198,)
(198, 1)
```



Se sabe que la fuerza sobre un objeto es igual a la tasa de cambio de la cantidad de movimiento de un objeto, el producto de la masa y la velocidad. Y la fuerza máxima instantánea se puede calcular con el máximo del producto entre la masa y la derivada de la velocidad en el tiempo.

```
v_fast_a = numerical_derivative(p_fast_a)
v_fast_b = numerical_derivative(p_fast_b)
v_fast_c = numerical_derivative(p_fast_c)
v_fast_d = numerical_derivative(p_fast_d)
v_fast_e = numerical_derivative(p_fast_e)

v_slow_a = numerical_derivative(p_slow_a)
v_slow_b = numerical_derivative(p_slow_b)
v_slow_c = numerical_derivative(p_slow_c)
v_slow_d = numerical_derivative(p_slow_d)
v_slow_e = numerical_derivative(p_slow_e)

a_fast_a = numerical_derivative(v_fast_a)
a_fast_b = numerical_derivative(v_fast_b)
a_fast_c = numerical_derivative(v_fast_c)
a_fast_d = numerical_derivative(v_fast_d)
a_fast_e = numerical_derivative(v_fast_e)
```

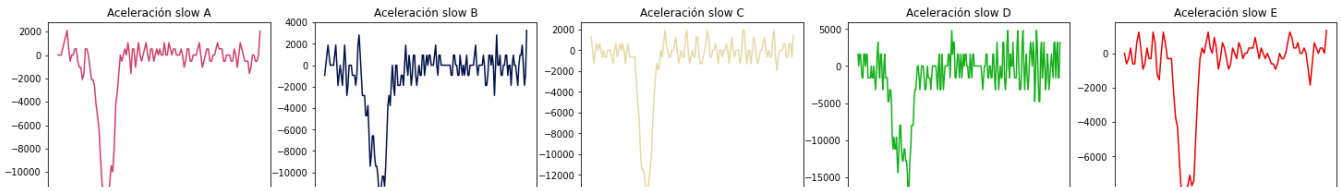
```
a_slow_a = numerical_derivative(v_slow_a)
a_slow_b = numerical_derivative(v_slow_b)
a_slow_c = numerical_derivative(v_slow_c)
a_slow_d = numerical_derivative(v_slow_d)
a_slow_e = numerical_derivative(v_slow_e)


# Graficamos las aceleraciones
fig = plt.figure(figsize=(25, 8))

ax1 = fig.add_subplot(2, 5, 1)
ax2 = fig.add_subplot(2, 5, 2)
ax3 = fig.add_subplot(2, 5, 3)
ax4 = fig.add_subplot(2, 5, 4)
ax5 = fig.add_subplot(2, 5, 5)

ax6 = fig.add_subplot(2, 5, 6)
ax7 = fig.add_subplot(2, 5, 7)
ax8= fig.add_subplot(2, 5, 8)
ax9 = fig.add_subplot(2, 5, 9)
ax10 = fig.add_subplot(2, 5, 10)

ax1.plot(a_slow_a[2:], color='xkcd:dark pink')
ax2.plot(a_slow_b[2:], color='xkcd:navy blue')
ax3.plot(a_slow_c[2:], color='xkcd:beige')
ax4.plot(a_slow_d[2:], color='xkcd:green')
ax5.plot(a_slow_e[2:], color='xkcd:red')

ax6.plot(a_fast_a[2:], color='xkcd:dark pink')
ax7.plot(a_fast_b[2:], color='xkcd:navy blue')
ax8.plot(a_fast_c[2:], color='xkcd:beige')
ax9.plot(a_fast_d[2:], color='xkcd:green')
ax10.plot(a_fast_e[2:], color='xkcd:red')

ax1.set_title("Aceleración slow A")
ax2.set_title("Aceleración slow B")
ax3.set_title("Aceleración slow C")
ax4.set_title("Aceleración slow D")
ax5.set_title("Aceleración slow E")

ax6.set_title("Aceleración fast A")
ax7.set_title("Aceleración fast B")
ax8.set_title("Aceleración fast C")
ax9.set_title("Aceleración fast D")
ax10.set_title("Aceleración fast E")


plt.show()
```



Filtramos todas las aceleraciones con una ventana móvil.

```
wn=10
a_slow_a_f = smoothData(a_slow_a[2:],wn)
a_slow_b_f = smoothData(a_slow_b[2:],wn)
a_slow_c_f = smoothData(a_slow_c[2:],wn)
a_slow_d_f = smoothData(a_slow_d[2:],wn)
a_slow_e_f = smoothData(a_slow_e[2:],wn)

a_fast_a_f = smoothData(a_fast_a[2:],wn)
a_fast_b_f = smoothData(a_fast_b[2:],wn)
a_fast_c_f = smoothData(a_fast_c[2:],wn)
a_fast_d_f = smoothData(a_fast_d[2:],wn)
a_fast_e_f = smoothData(a_fast_e[2:],wn)

# Graficamos las aceleraciones
fig = plt.figure(figsize=(25, 8))

ax1 = fig.add_subplot(2, 5, 1)
ax2 = fig.add_subplot(2, 5, 2)
ax3 = fig.add_subplot(2, 5, 3)
ax4 = fig.add_subplot(2, 5, 4)
ax5 = fig.add_subplot(2, 5, 5)

ax6 = fig.add_subplot(2, 5, 6)
ax7 = fig.add_subplot(2, 5, 7)
ax8= fig.add_subplot(2, 5, 8)
ax9 = fig.add_subplot(2, 5, 9)
```

```
ax9 = fig.add_subplot(2, 5, 9)
ax10 = fig.add_subplot(2, 5, 10)



ax1.plot(a_slow_a_f, color='xkcd:dark pink')
ax2.plot(a_slow_b_f, color='xkcd:navy blue')
ax3.plot(a_slow_c_f, color='xkcd:beige')
ax4.plot(a_slow_d_f, color='xkcd:green')
ax5.plot(a_slow_e_f, color='xkcd:red')

ax6.plot(a_fast_a_f, color='xkcd:dark pink')
ax7.plot(a_fast_b_f, color='xkcd:navy blue')
ax8.plot(a_fast_c_f, color='xkcd:beige')
ax9.plot(a_fast_d_f, color='xkcd:green')
ax10.plot(a_fast_e_f, color='xkcd:red')

ax1.set_title("Aceleración slow A")
ax2.set_title("Aceleración slow B")
ax3.set_title("Aceleración slow C")
ax4.set_title("Aceleración slow D")
ax5.set_title("Aceleración slow E")

ax6.set_title("Aceleración fast A")
ax7.set_title("Aceleración fast B")
ax8.set_title("Aceleración fast C")
ax9.set_title("Aceleración fast D")
ax10.set_title("Aceleración fast E")


plt.show()
```



Cálculos de las fuerzas máximas instantáneas

```
# Obtenemos el máximo de la aceleración
print( max(abs(a_filter)) )
```

```
40129.714285714304
```