smsr_osc_compare_Svet1

May 7, 2024

1 -3, -1

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
```

[ ]:
```
#
#                    ,   :
Pnom1=107

#                csv
df_main = pd.read_csv('./Svet/Svet_main_data.csv', sep=';', encoding="ansi",␣
 ↪decimal=',', skiprows=1050)
df_main.columns = ["TimeUTC",
                   "G1_Fa", "G1_U1", "G1_P", "G1_Q", "G1_If","G1_Uf",
                   "G2_Fa", "G2_U1", "G2_P", "G2_Q", "G2_If","G2_Uf",
                   "G3_Fa", "G3_U1", "G3_P", "G3_Q", "G3_If","G3_Uf",]
#                      datetime
df_main['TimeUTC'] = pd.to_datetime(df_main['TimeUTC'])
#                        ,
mask = df_main['TimeUTC'].dt.microsecond == 0
#            DataFrame,
df_main = df_main[mask]

#                csv
df_sk = pd.read_csv('./Svet/Svet_results_data.csv', sep=';', encoding="ansi",␣
 ↪decimal=',')
df_sk.columns = ["TimeUTC",
                   "G1_def", "G1_def_P", "G1_def_Q", "G1_def_P_amp",␣
 ↪"G1_def_P_freq","G1_def_Q_amp", "G1_def_Q_freq",
                   "G2_def", "G2_def_P", "G2_def_Q", "G2_def_P_amp",␣
 ↪"G2_def_P_freq","G2_def_Q_amp", "G2_def_Q_freq",
                   "G3_def", "G3_def_P", "G3_def_Q", "G3_def_P_amp",␣
 ↪"G3_def_P_freq","G3_def_Q_amp", "G3_def_Q_freq"]
```

```
df_sk["time"] = ((pd.to_datetime(df_sk["TimeUTC"]).astype('int64') / 1e6 -␣
 ↪float(pd.to_datetime(df_sk["TimeUTC"][0]).to_datetime64()) / 1e6).
 ↪astype('int64')) / 1e3
#                csv
df_ext_SMSR = pd.read_csv('./Svet/SMSR_SIG_EXT_data_Svet_1.csv', sep=';',␣
 ↪encoding="windows-1251")
#           diagno                        (               df_ext)
diagno_SMSR = df_ext[["osh_ARV1_ext[6]", "osh_ARV1_ext[7]", "osh_ARV1_ext[8]",␣
 ↪"osh_ARV1_ext[9]", "osh_ARV1_ext[10]"]]
diagno_SMSR.columns = ["bug1", "bug2", "bug3", "bug4", "bug5"]
#           result_osc                                  (               df_ext)
results_SMSR = df_ext[["osh_ARV1_ext[{:d}]".format(i) for i in range(11, 25)]]
results_SMSR.columns = ["osc1_f", "osc1_phi", "osc1_np", "osc2_f", "osc2_phi",␣
 ↪"osc2_np", "osc3_f", "osc3_phi", "osc3_np", "osc4_f", "osc4_phi", "osc4_np",␣
 ↪"osc5_f", "osc5_amp"]
#           osh                                (               df_ext)
osh_SMSR = df_ext[["osh_ARV1_ext[1]", "osh_ARV1_ext[2]", "osh_ARV1_ext[3]",␣
 ↪"osh_ARV1_ext[4]", "osh_ARV1_ext[5]"]]
osh_SMSR.columns = ["osc1", "osc2", "osc3", "osc4", "osc5"]
```

```
C:\Users\das\AppData\Local\Temp\ipykernel_23876\1869202340.py:12: UserWarning:
Parsing dates in %d.%m.%Y %H:%M:%S.%f format when dayfirst=False (the default)
was specified. Pass `dayfirst=True` or specify a format to silence this warning.
  df_main['TimeUTC'] = pd.to_datetime(df_main['TimeUTC'])
C:\Users\das\AppData\Local\Temp\ipykernel_23876\1869202340.py:24: UserWarning:
Parsing dates in %d.%m.%Y %H:%M:%S.%f format when dayfirst=False (the default)
was specified. Pass `dayfirst=True` or specify a format to silence this warning.
  df_sk["time"] = ((pd.to_datetime(df_sk["TimeUTC"]).astype('int64') / 1e6 -
float(pd.to_datetime(df_sk["TimeUTC"][0]).to_datetime64()) /
1e6).astype('int64')) / 1e3
C:\Users\das\AppData\Local\Temp\ipykernel_23876\1869202340.py:24: UserWarning:
Parsing dates in %d.%m.%Y %H:%M:%S.%f format when dayfirst=False (the default)
was specified. Pass `dayfirst=True` or specify a format to silence this warning.
  df_sk["time"] = ((pd.to_datetime(df_sk["TimeUTC"]).astype('int64') / 1e6 -
float(pd.to_datetime(df_sk["TimeUTC"][0]).to_datetime64()) /
1e6).astype('int64')) / 1e3
```

2

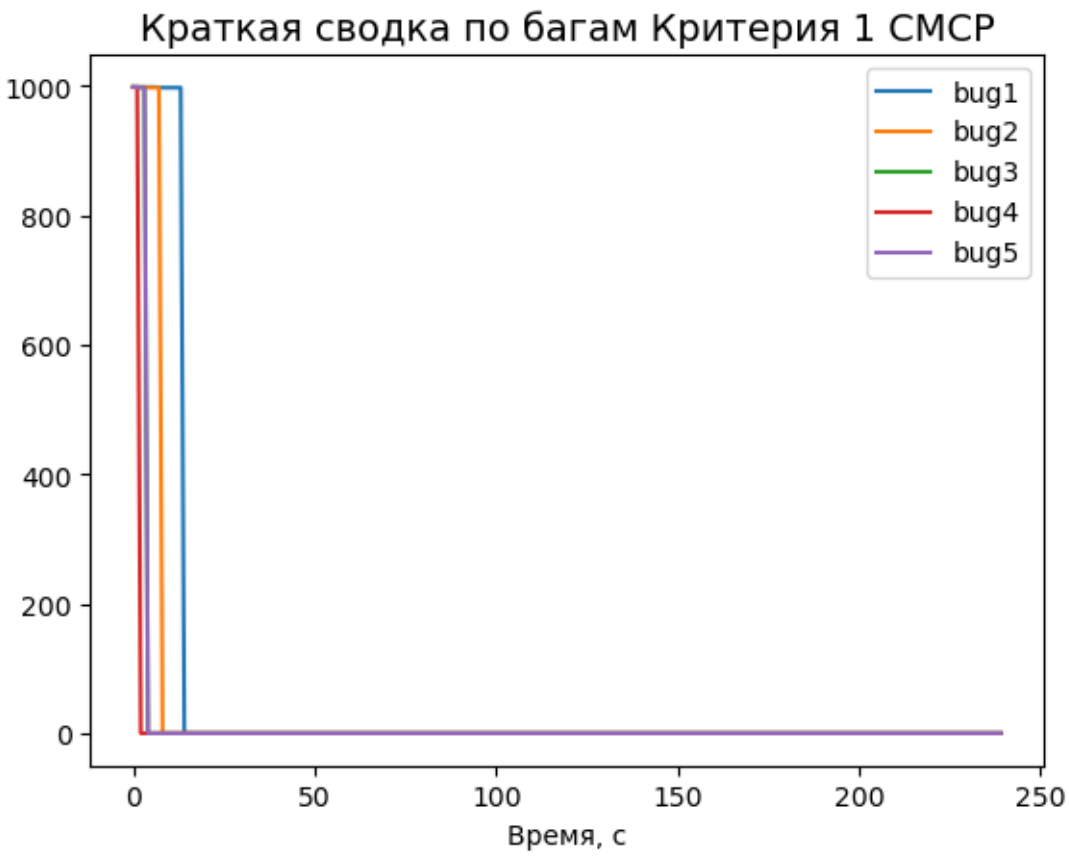
[ ]:
```
#
plt.title('          ', fontsize=14)
plt.xlabel('   ,  ')
plt.ylabel('     ,   ')
x=df_sk['time']
y=df_main["G1_P"]/1000000
plt.plot(x,y,'firebrick', label='          ')
plt.legend()
```

```
plt.show()
```

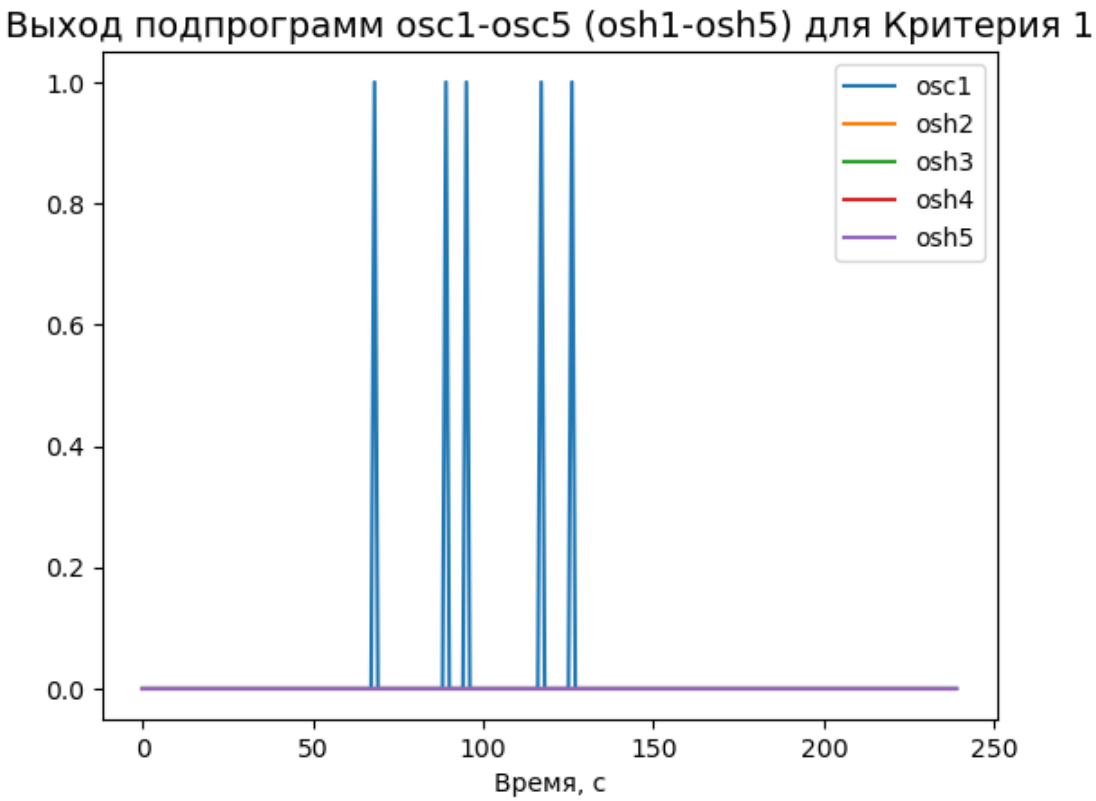
[ ]:
```
#
plt.title('           ', fontsize=14)
plt.xlabel('   ,  ')
plt.ylabel('     ,    ')
x=df_sk['time']
y=df_main["G1_Q"]/1000000
plt.plot(x,y,'mediumpurple', label='           ')
plt.legend()
plt.show()
```

3

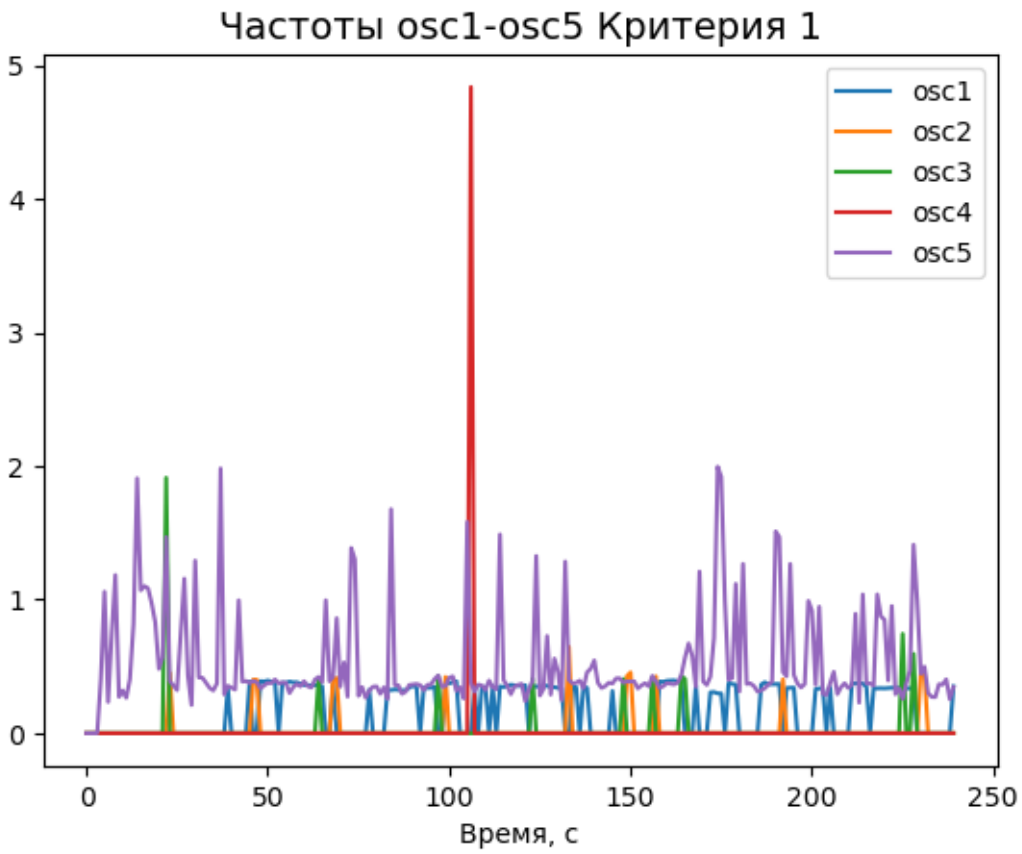
[ ]:
```
#                    1
plt.title('                    1    ', fontsize=14)
plt.xlabel('   ,  ')
plt.plot(diagno_SMSR["bug1"], label='bug1')
plt.plot(diagno_SMSR["bug2"], label='bug2')
plt.plot(diagno_SMSR["bug3"], label='bug3')
plt.plot(diagno_SMSR["bug4"], label='bug4')
plt.plot(diagno_SMSR["bug5"], label='bug5')
plt.legend()
plt.show()
```



[ ]:
```
#           osc1-osc5 (osh1-osh5)      1
plt.title('           osc1-osc5 (osh1-osh5)        1', fontsize=14)
plt.xlabel('   ,  ')
plt.plot(osh_SMSR["osc1"], label='osc1')
plt.plot(osh_SMSR["osc2"], label='osh2')
plt.plot(osh_SMSR["osc3"], label='osh3')
plt.plot(osh_SMSR["osc4"], label='osh4')
plt.plot(osh_SMSR["osc5"], label='osh5')
plt.legend()
plt.show()
```

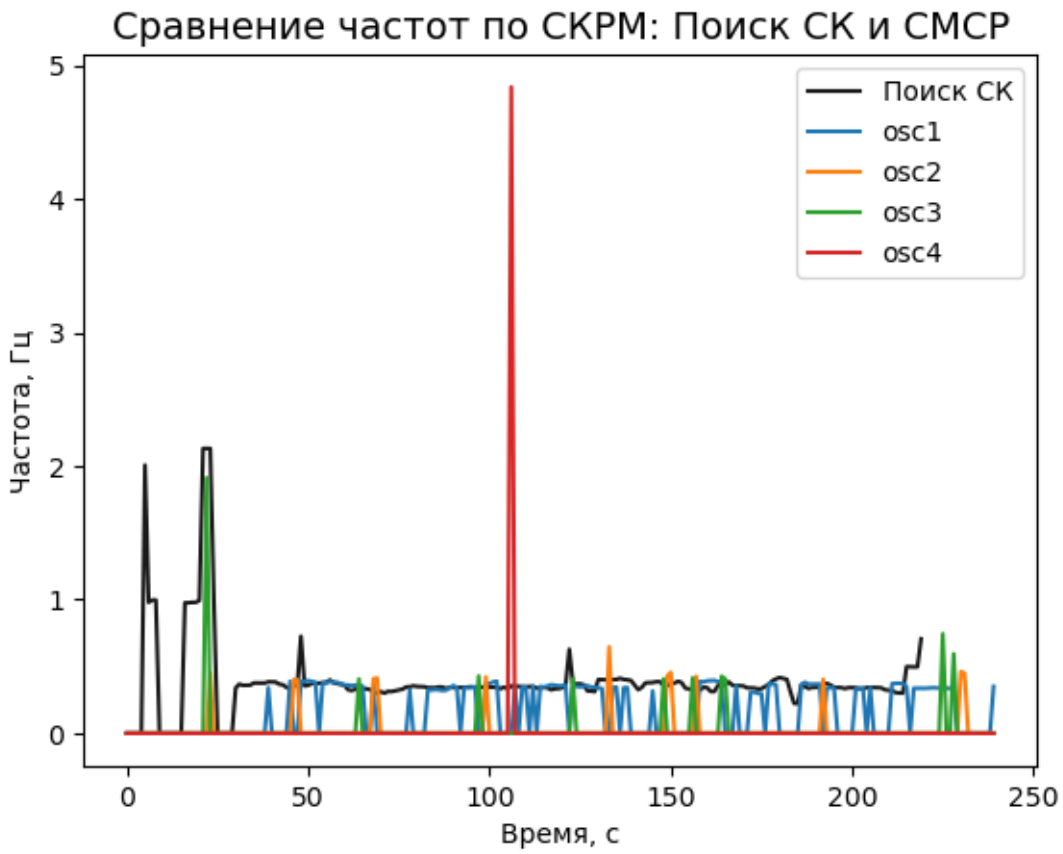


[ ]:
```
#      osc1-osc5       1
plt.title('     osc1-osc5       1', fontsize=14)
plt.xlabel('   ,  ')
plt.plot(results_SMSR["osc1_f"], label='osc1')
plt.plot(results_SMSR["osc2_f"], label='osc2')
plt.plot(results_SMSR["osc3_f"], label='osc3')
plt.plot(results_SMSR["osc4_f"], label='osc4')
plt.plot(results_SMSR["osc5_f"], label='osc5')
plt.legend()
plt.show()
```



[ ]:
```
#           -     (            )
x=df_sk["time"]
y1=df_sk["G1_def_P_freq"]
y1_filled = pd.Series(y1).fillna(method='ffill')
plt.title('               :          ', fontsize=14)
plt.xlabel('   ,  ')
plt.ylabel('    ,  ')
plt.plot(x,y1_filled,'0.1', label='     ')
plt.plot(results_SMSR["osc1_f"], label='osc1')
plt.plot(results_SMSR["osc2_f"], label='osc2')
plt.plot(results_SMSR["osc3_f"], label='osc3')
plt.plot(results_SMSR["osc4_f"], label='osc4')
plt.legend()
plt.show()
```

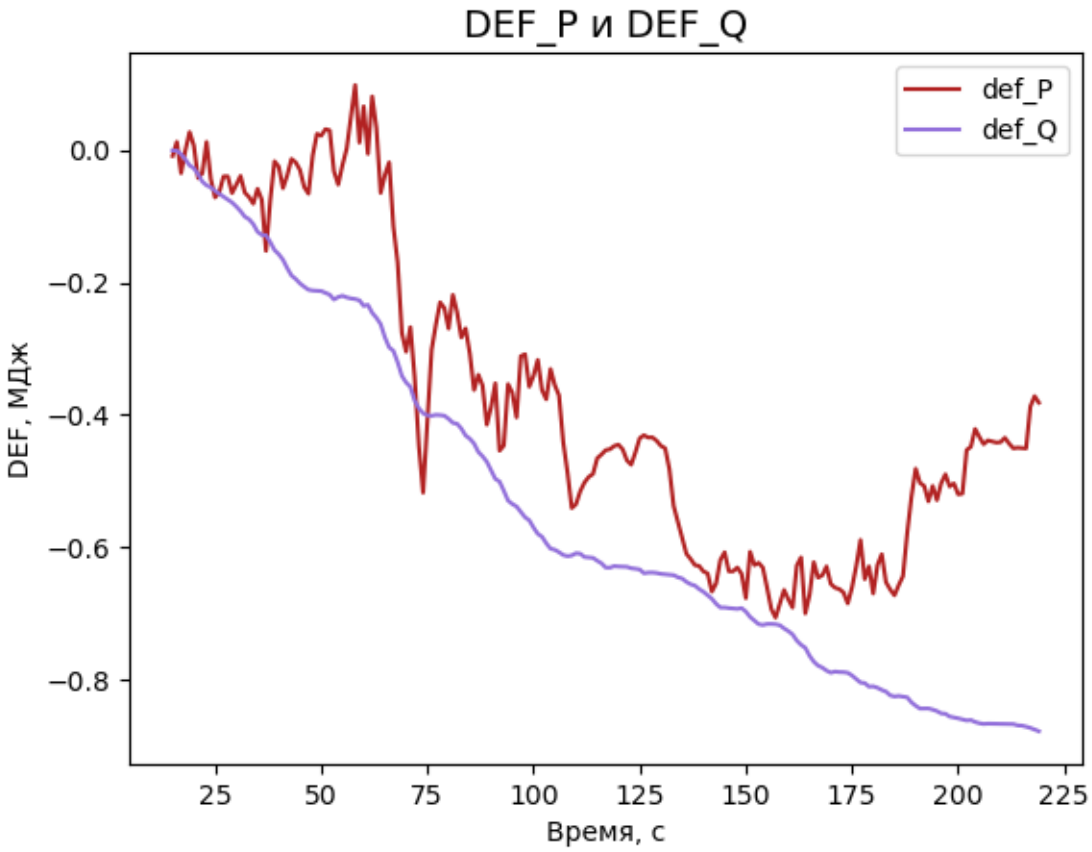
```
C:\Users\das\AppData\Local\Temp\ipykernel_23876\2146378088.py:4: FutureWarning:
Series.fillna with 'method' is deprecated and will raise in a future version.
Use obj.ffill() or obj.bfill() instead.
  y1_filled = pd.Series(y1).fillna(method='ffill')
```



[ ]:
```
#       -       (             )
x=df_sk["time"]
y1=df_sk["G1_def_Q_amp"]/1000000
y1_filled = pd.Series(y1).fillna(method='ffill')
plt.title('                :      ', fontsize=14)
plt.xlabel('   ,  ')
plt.ylabel('     ,    ')
plt.plot(x,y1_filled,'mediumpurple', label='     ')
plt.legend()
plt.show()
```

```
C:\Users\das\AppData\Local\Temp\ipykernel_23876\2729482021.py:4: FutureWarning:
Series.fillna with 'method' is deprecated and will raise in a future version.
Use obj.ffill() or obj.bfill() instead.
  y1_filled = pd.Series(y1).fillna(method='ffill')
```

[ ]:
```
# DEF_P   DEF_Q
x=df_sk["time"]
y1=df_sk["G1_def_P"]/1000000
y2=df_sk["G1_def_Q"]/1000000
plt.title('DEF_P   DEF_Q', fontsize=14)
plt.xlabel('   ,  ')
plt.ylabel('DEF,   ')
plt.plot(x,y1,'firebrick',label='def_P')
plt.plot(y2,'mediumpurple',label='def_Q')
plt.legend()
plt.show()
```

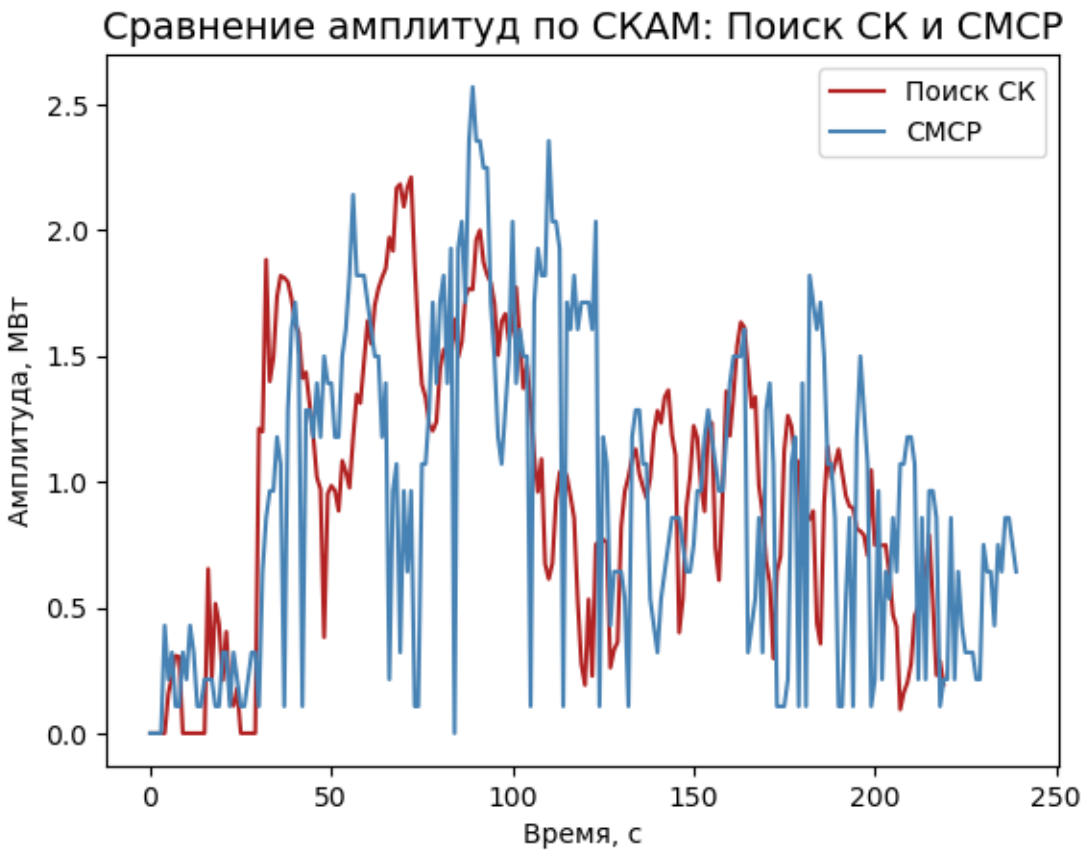


[ ]:
```
#           -     (           )
x=df_sk["time"]
y1=df_sk["G1_def_P_freq"]
y1_filled = pd.Series(y1).fillna(method='ffill')
y2=results_SMSR["osc5_f"]
plt.title('              :          ', fontsize=14)
plt.xlabel('   ,  ')
plt.ylabel('    ,  ')
plt.plot(x,y1_filled,'firebrick', label='     ')
plt.plot(y2,'steelblue', label='  ')
plt.legend()
plt.show()
```

```
C:\Users\das\AppData\Local\Temp\ipykernel_23876\2997130592.py:4: FutureWarning:
Series.fillna with 'method' is deprecated and will raise in a future version.
Use obj.ffill() or obj.bfill() instead.
  y1_filled = pd.Series(y1).fillna(method='ffill')
```

[ ]:
```
#           -      (           )
x=df_sk["time"]
y1=df_sk["G1_def_P_amp"]/1000000
y1_filled = pd.Series(y1).fillna(method='ffill')
y2=results_SMSR["osc5_amp"]*Pnom1
plt.title('                :          ', fontsize=14)
plt.xlabel('   ,  ')
plt.ylabel('     ,   ')
plt.plot(x,y1_filled,'firebrick', label='     ')
plt.plot(y2,'steelblue', label='   ')
plt.legend()
plt.show()
```

```
C:\Users\das\AppData\Local\Temp\ipykernel_23876\586679042.py:4: FutureWarning:
Series.fillna with 'method' is deprecated and will raise in a future version.
Use obj.ffill() or obj.bfill() instead.
  y1_filled = pd.Series(y1).fillna(method='ffill')
```


Сравнение амплитуд по СКАМ: Поиск СК и СМСР
Поиск СК
СМСР
Амплитуда, МВт
Время, с
0.0
0.5
1.0
1.5
2.0
2.5
0
50
100
150
200
250