First look on two eventually cointegrated symbols for stat arb trading

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
import sys
sys.path.append('../../')
sys.path.append('../../../')
sys.path.append('../')
import helpers
import datetime
from pylab import rcParams
rcParams['figure.figsize'] = 13, 6
from statsmodels.tsa import stattools as stat
```

We would like to trade the spread: the difference $y = \beta x$ where x is e.g. USDCHF and y is GBPCHF. We estimate β to be a constant: $\beta = y_0/x_0$ from the very first sample of the lists.

Reading Data

In [2]:
```
n1=50320
tf="5m"
symbol1="usdchf"
symbol2="gbpchf"
x_candles=helpers.download_candles(host="172.19.0.5", symbol=symbo
l1, timeframe=tf, num=n1)
y_candles=helpers.download_candles(host="172.19.0.5", symbol=symbo
l2, timeframe=tf, num=n1) #1d
```

In [3]:
```
y_candles.head()
```

Out[3]:

time	symbol	open	close	min	max	vol	timestamp	interval	closed
2020-12-08 11:05:00	gbpchf	1.18778	1.18744	1.18729	1.18799	580	2020-12-08 11:09:59	5	True
2020-12-08 11:10:00	gbpchf	1.18744	1.18818	1.18720	1.18819	414	2020-12-08 11:14:59	5	True
2020-12-08 11:15:00	gbpchf	1.18819	1.18818	1.18780	1.18844	518	2020-12-08 11:19:59	5	True
2020-12-08 11:20:00	gbpchf	1.18818	1.18818	1.18804	1.18860	460	2020-12-08 11:24:59	5	True
2020-12-08 11:25:00	gbpchf	1.18818	1.18738	1.18727	1.18837	394	2020-12-08 11:29:59	5	True

In [4]:
```
x_candles.head()
```

Out[4]:

time	symbol	open	close	min	max	vol	timestamp	interval	closed
2020-12-08 09:50:00	usdchf	0.89078	0.89058	0.89050	0.89082	152	2020-12-08 09:54:59	5	True
2020-12-08 09:55:00	usdchf	0.89058	0.89063	0.89024	0.89071	257	2020-12-08 09:59:59	5	True
2020-12-08 10:00:00	usdchf	0.89062	0.89108	0.89047	0.89112	449	2020-12-08 10:04:59	5	True
2020-12-08 10:05:00	usdchf	0.89108	0.89104	0.89094	0.89120	328	2020-12-08 10:09:59	5	True
2020-12-08 10:10:00	usdchf	0.89104	0.89110	0.89102	0.89132	408	2020-12-08 10:14:59	5	True

Normalize to first sample

$y = \beta x, \beta = y_0/x_0$

In [5]:
```
b=y_candles['close'].iloc[0]/x_candles['close'].iloc[0]
print("beta: "+str(b))
yt=y_candles['close']
xt=x_candles['close']*b
```

```
beta: 1.3333333333333333
```

Check what the traditional cointegration estimation tells us

A p value < 0.05 is perfect.

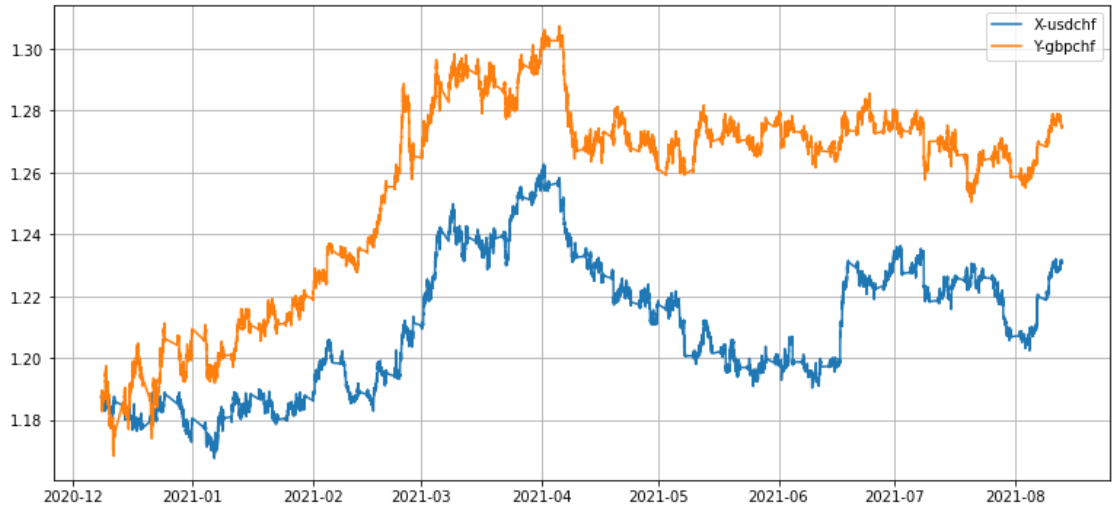
In [6]:
```
#see https://www.statsmodels.org/stable/generated/statsmodels.tsa.stattools.coint.html
ct, pval, crit_val=stat.coint(xt.values, yt.values, trend='c', method='aeg')
```

In [7]:
```
print("ct: "+str(ct))
print("pval: "+str(pval))
print("crit_val: "+str(crit_val))
```

```
ct: -2.687853840812084
pval: 0.20397752252285173
crit_val: [-3.89665766 -3.33625143 -3.04453429]
```

Plot the prices

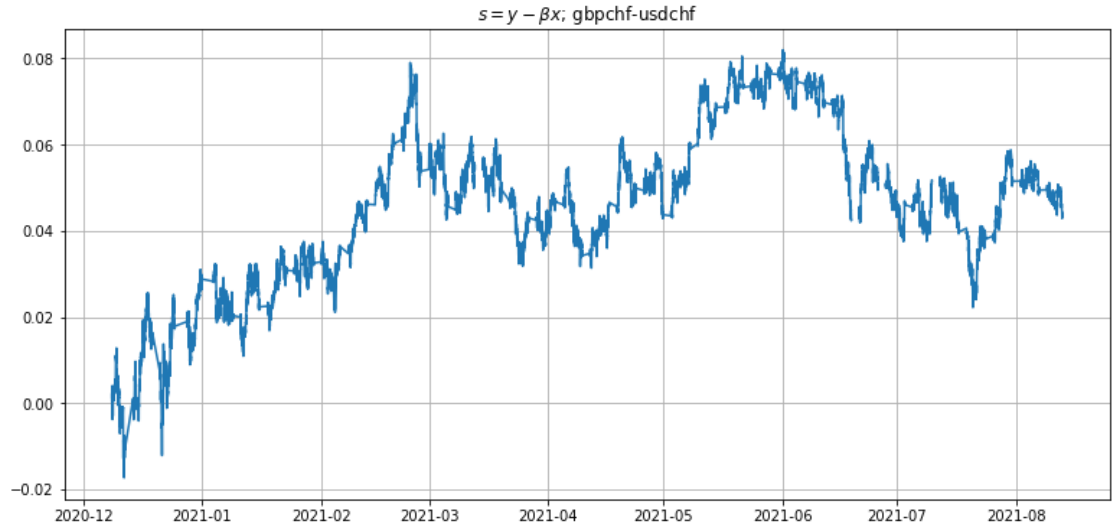
In [8]:
```
plt.plot(xt, label='X-'+symbol1)
plt.plot(yt, label='Y-'+symbol2)
plt.legend()
plt.grid()
plt.show()
```



Plot the spread $s(t) = y(t) - \beta_0 x(t)$

the spread over time:

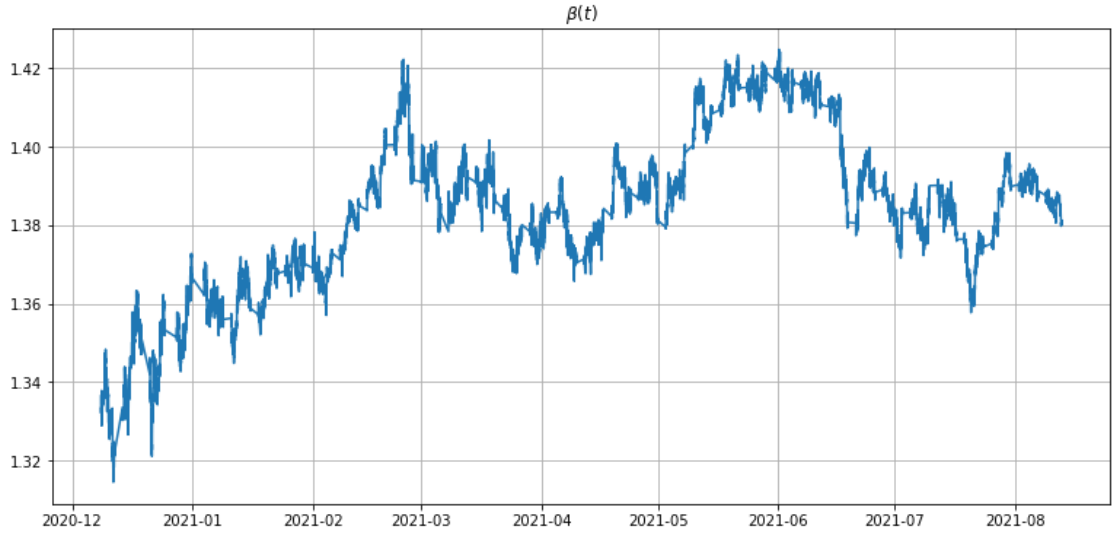
In [9]:
```
diff=yt-xt
#diff=diff[300:5000] #one week
plt.plot(diff)
plt.title(r"$s=y-\beta x$; "+symbol2+"-"+symbol1)
plt.grid()
```



Plot the scale β over time

$\beta(t) = y(t)/x(t)$

In [10]:
```
scale_t=y_candles['close'].values/x_candles['close'].values
plt.plot(y_candles['time'], scale_t)
plt.title(r"$\beta(t)$")
plt.grid()
```



Trading

In order to trade the spread, one needs to open two positions: lets say we would like to short the spread on 1st June. $y = 1$lot and $y = \beta x = 1.4 lots$.

And going short would mean we open -1lot of y (GBPUSD) at the price of 1.28 and -1.4 lots of x (USDCHF) at the price of 1.20.

When it felt down from 0.08 to 0.04 around 1st July 2021, the GBPCHF price was still 1.28 and USDCHF was approx 1.23. On the y-GBPUSD side we would have made 0 profit (minus swap and comissions). On the x-USDCHF side we would have made

In [11]:
```
leverage=500
pos_size_lots=1.4
lot_in_eur=250 #this is the margin required to open 1 lot of x (US
DCHF).
prc=(1.23/1.20-1)*100
print("gain of "+str(round(prc,2))+"%")
gain=prc*leverage/100*pos_size_lots*lot_in_eur
print("gain in €: "+str(round(gain,2))+"€")
```

```
gain of 2.5%
gain in €: 4375.0€
```

Custom Symbol in MT

The next test would be to create a custom symbol in MT and to see how CP (2.36) behaves.

In [12]:
```
df=pd.DataFrame(diff)#{'CLOSE':diff.values})

df['OPEN']=y_candles['open']-x_candles['open']*b
df['HIGH']=y_candles['max']-x_candles['max']*b
df['LOW']=y_candles['min']-x_candles['min']*b
df['TICKVOL']=1
df['VOL']=2
df['SPREAD']=0
df=df*1000

df['DATE']=y_candles['time'].dt.strftime('%Y.%m.%d')
df['TIME']=y_candles['time'].dt.strftime('%H:%M:%S')

df.dropna(inplace=True)
df.head()
```

Out[12]:

time	close	OPEN	HIGH	LOW	TICKVOL	VOL	SPREAD	DATE
2020-12-08 11:05:00	1.253333	1.726667	1.696667	1.370000	1000	2000	0	2020.12.08
2020-12-08 11:10:00	2.246667	1.253333	1.710000	1.306667	1000	2000	0	2020.12.08
2020-12-08 11:15:00	2.060000	2.256667	2.280000	2.013333	1000	2000	0	2020.12.08
2020-12-08 11:20:00	1.806667	2.060000	2.200000	2.000000	1000	2000	0	2020.12.08
2020-12-08 11:25:00	1.366667	1.806667	1.996667	1.443333	1000	2000	0	2020.12.08

In [13]:
```
df.to_csv("./mt_spread_"+symbol1+"_"+symbol2+".csv", index=False,
columns=["DATE", "TIME", "OPEN", "HIGH", "LOW", "close", "TICKVO
L", "VOL", "SPREAD"])
```

In [ ]:
```

```