Three-way_timestamps_win10

October 26, 2019

1 Comparison of timestamps in three alternative NMEA data paths

1.0.1 Windows 10 - OpenCPN v5.0.0 - Signal K v1.17.0 - DashT v.0.5.2

We observe a five minute sampling period stored in InfluxDB database for each of the use case for single value of Apparent Wind Angle:

1. data via Signal K delta TCP channel with Signal K timestamps at its own reception

2. data via Signal K to NMEA-0183 via TCP channel timestamps at reception at the InfluxDB instruments

3. data directly from USB to OpenCPN

In all above cases the USB is set to 115200 baud at reception on Win10 (Surface 3 i7) running OpenCPN v5.0.0. Data is originated from Raymarine SeaTalk (4800 baud) and converted to USB in MiniPlex II multiplexer - about 40 values per second are transmitted through this channel but only Apparent Wind Angle timestamp behaviour is observed.

(zoom)

1.1 Numerical comparison

```
[1]: import numpy as np
     import pandas as pd
```

1.1.1 Data via Signal K delta TCP channel

 (zoom)

```
[2]: df = pd.read_csv("2019-10-05_224308_SignalK_delta_zoom.csv", sep=',', header=3)
```

```
[3]: df.head()
```

```
[3]:    Unnamed: 0  result  table                _start                 _stop  \
     0         NaN     NaN      0  2019-10-05T20:06:00Z  2019-10-05T20:11:00Z
     1         NaN     NaN      0  2019-10-05T20:06:00Z  2019-10-05T20:11:00Z
     2         NaN     NaN      0  2019-10-05T20:06:00Z  2019-10-05T20:11:00Z
     3         NaN     NaN      0  2019-10-05T20:06:00Z  2019-10-05T20:11:00Z
     4         NaN     NaN      0  2019-10-05T20:06:00Z  2019-10-05T20:11:00Z

                          _time  _value        _field _measurement prop1
     0  2019-10-05T20:06:00.969Z    16.0  angleApparent  environment  wind
     1  2019-10-05T20:06:02.904Z    17.0  angleApparent  environment  wind
     2  2019-10-05T20:06:04.841Z    23.0  angleApparent  environment  wind
     3  2019-10-05T20:06:06.775Z    30.5  angleApparent  environment  wind
     4   2019-10-05T20:06:08.71Z    31.0  angleApparent  environment  wind
```

```
[4]: df._value.describe()
```

```
[4]: count    158.000000
     mean      16.382911
     std        8.111321
     min        0.500000
     25%        9.500000
     50%       17.250000
     75%       23.000000
     max       34.000000
     Name: _value, dtype: float64
```

```
[5]: df1 = pd.to_datetime(df['_time'])
```

```
[6]: df1.describe()
```

```
[6]: count                                    158
     unique                                   158
     top         2019-10-05 20:09:49.403000+00:00
     freq                                       1
     first       2019-10-05 20:06:00.969000+00:00
     last        2019-10-05 20:10:58.127000+00:00
     Name: _time, dtype: object
```

```
[7]: df2 = df1.astype(np.int64).div(1e6)
```

```
[8]: df3 = df2.diff()
```

```
[9]: df3.describe()
```

```
[9]: count     157.000000
     mean     1892.726115
     std       188.331733
     min      1001.000000
     25%      1925.000000
     50%      1935.000244
     75%      1940.000000
     max      2058.000244
     Name: _time, dtype: float64
```

1.1.2 Data via Signal K to NMEA-0183 converter TCP channel

(zoom)

```
[10]: nf = pd.read_csv("2019-10-05_224509_SignalK_NMEA_TCP_zoom.csv", sep=',',
      →header=3)
```

```
[11]: nf.head()
```

```
[11]:    Unnamed: 0  result  table              _start               _stop  \
      0         NaN     NaN      0  2019-10-05T20:16:00Z  2019-10-05T20:21:00Z
      1         NaN     NaN      0  2019-10-05T20:16:00Z  2019-10-05T20:21:00Z
      2         NaN     NaN      0  2019-10-05T20:16:00Z  2019-10-05T20:21:00Z
      3         NaN     NaN      0  2019-10-05T20:16:00Z  2019-10-05T20:21:00Z
      4         NaN     NaN      0  2019-10-05T20:16:00Z  2019-10-05T20:21:00Z


                          _time  _value         _field _measurement prop1
      0  2019-10-05T20:16:01.101Z     1.5  angleApparent  environment  wind
      1  2019-10-05T20:16:03.041Z    13.5  angleApparent  environment  wind
      2  2019-10-05T20:16:04.981Z     0.5  angleApparent  environment  wind
      3  2019-10-05T20:16:06.015Z    15.0  angleApparent  environment  wind
      4  2019-10-05T20:16:07.881Z     7.0  angleApparent  environment  wind
```

```
[12]: nf._value.describe()
```

```
[12]: count    159.000000
      mean      10.710692
      std        8.320781
      min        0.000000
      25%        3.500000
      50%        9.500000
      75%       15.000000
      max       52.000000
      Name: _value, dtype: float64
```

```
[13]: nf1 = pd.to_datetime(nf['_time'])
```

```
[14]: nf1.describe()
```

```
[14]: count                           159
      unique                          159
      top       2019-10-05 20:16:01.101000+00:00
      freq                              1
      first     2019-10-05 20:16:01.101000+00:00
      last      2019-10-05 20:20:58.255000+00:00
      Name: _time, dtype: object
```

```
[15]: nf2 = nf1.astype(np.int64).div(1e6)
```

```
[16]: nf3 = nf2.diff()
```

```
[17]: nf3.describe()
```

```
[17]: count     158.000000
      mean     1880.721519
      std       212.353998
```

```
min         1004.999756
25%         1917.500000
50%         1935.000000
75%         1944.000000
max         2046.999756
Name: _time, dtype: float64
```

1.1.3   Data without Signal K directly from USB

(zoom)

```
[18]: of = pd.read_csv("2019-10-05_224657_USB_to_O_zoom.csv", sep=',', header=3)
```

```
[19]: of.head()
```

```
[19]:    Unnamed: 0  result  table                _start                 _stop  \
      0         NaN     NaN      0  2019-10-05T20:25:00Z  2019-10-05T20:30:00Z
      1         NaN     NaN      0  2019-10-05T20:25:00Z  2019-10-05T20:30:00Z
      2         NaN     NaN      0  2019-10-05T20:25:00Z  2019-10-05T20:30:00Z
      3         NaN     NaN      0  2019-10-05T20:25:00Z  2019-10-05T20:30:00Z
      4         NaN     NaN      0  2019-10-05T20:25:00Z  2019-10-05T20:30:00Z

                          _time  _value          _field _measurement prop1
      0  2019-10-05T20:25:00.249Z     3.5  angleApparent  environment  wind
      1  2019-10-05T20:25:02.189Z     8.5  angleApparent  environment  wind
      2  2019-10-05T20:25:04.106Z     9.0  angleApparent  environment  wind
      3  2019-10-05T20:25:06.051Z    17.0  angleApparent  environment  wind
      4   2019-10-05T20:25:07.99Z    24.0  angleApparent  environment  wind
```

```
[20]: of._value.describe()
```

```
[20]: count    165.000000
      mean      18.845455
      std        8.896015
      min        0.500000
```

5

```
25%         13.500000
50%         19.000000
75%         24.500000
max         49.000000
Name: _value, dtype: float64
```

[21]: `of1 = pd.to_datetime(of['_time'])`

[22]: `of1.describe()`

[22]:
```
count                                  165
unique                                 165
top        2019-10-05 20:25:11.876000+00:00
freq                                     1
first      2019-10-05 20:25:00.249000+00:00
last       2019-10-05 20:29:59.340000+00:00
Name: _time, dtype: object
```

[23]: `of2 = of1.astype(np.int64).div(1e6)`

[24]: `of3 = of2.diff()`

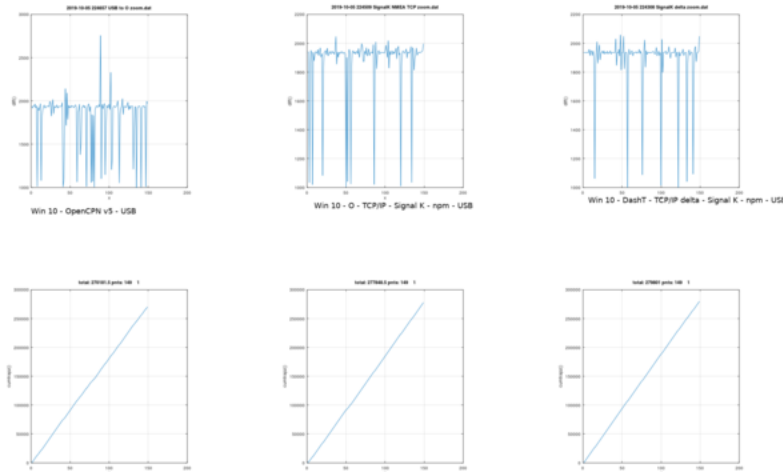[25]: `of3.describe()`

[25]:
```
count     164.000000
mean     1823.725610
std       299.438559
min      1000.000000
25%      1896.000000
50%      1931.500122
75%      1944.000000
max      2757.000244
Name: _time, dtype: float64
```

1.2 Graphical comparison

GNU Octave v5.1/MATLAB script `jitterplots.m` was developed to present the jitter in graphical plot format, including the cumulative difference. The datafiles were truncated to hold 150 measurement points each. A helper script `windowsjitter.m` was used to produce the following plots.

Win 10 - OpenCPN v5 - USB

Win 10 - O - TCP/IP - Signal K - npm - USB

Win 10 - DashT - TCP/IP delta - Signal K - npm - USB

(zoom)

1.3 Summary of results

data path	timestamp	standard deviation	maximum time difference
1 Signal K delta	at source	188 ms	2058 ms
2 Signal K NMEA TCP	at reception	212 ms	2047 ms
3 USB to OpenCPN	at reception	299 ms	2757 ms

1.4 Conclusion

Judged by a human eye there is no difference between the three methods - the needles and values are jumping back and forth as always!

The difference will come apparent when we want to eliminate that jumping by applying some statistical and continuous algorithms on the received time series data. The accuracy of the time stamps is, of course important for any time series analysis.

1. It is not surprising that the direct TCP connection to the Signal K emitted delta values is the most efficient what comes to the accuracy of the timestamps - they are set at the reception, *i.e.* at the closest possible position to the source. Although this method is penalized having to transmit also information in its payload to which we are not necessarily willing to be subscribed, the fact that the timestamp travels with the data compensates that inconvenience by reducing the jitter.

2. The fact that there is so little difference between the timestamp accuracy through the Signal K to NMEA conversion and its actual delta channel is a proof of the excellent quality and effiency of Signal K and npm implementation also on Windows platform. The TCP method of OpenCPN is also the preferred one compared to USB method since apparently the wxWidgets/OpenCPN TCP/IP interface works the best on Windows.

3. There is nothing to gain by eliminating Signal K to allow the OpenCPN to connect directly to the USB channel: this is the clearly the less desirable configuration for any algorithm which analyzes time series. It increases considerably the jitter albeit it performs well, but unfortunately in bursts.

On the Windows 10 platform the Signal K TCP feed to OpenCPN is the method to use, together with the Dashboard-Tactics plug-in's direct connection to Signal K's delta channel.

Finally, let's consider the following statement from *Numerical Recipes In C: The Art of Scientific Computing (ISBN 0-521-43108-5)* concerning irregularly sampled data, where the values f_i are not uniformly spaced in time:

> If the change in f across the full width of the $N = nL + nR + 1$ point window is less than $\sqrt{N/2}$ times the measurement noise on a single point, then the cheap method can be used.

The "*cheap method*" is to ignore the jitter. But as the above graphics illustrates, were are constantly switching, here for the wind data, between $1Hz$ and $0.5Hz$ for f_i!