# How to make irregular and missing sampling points uniform in LSL data

Makoto Miyakoshi

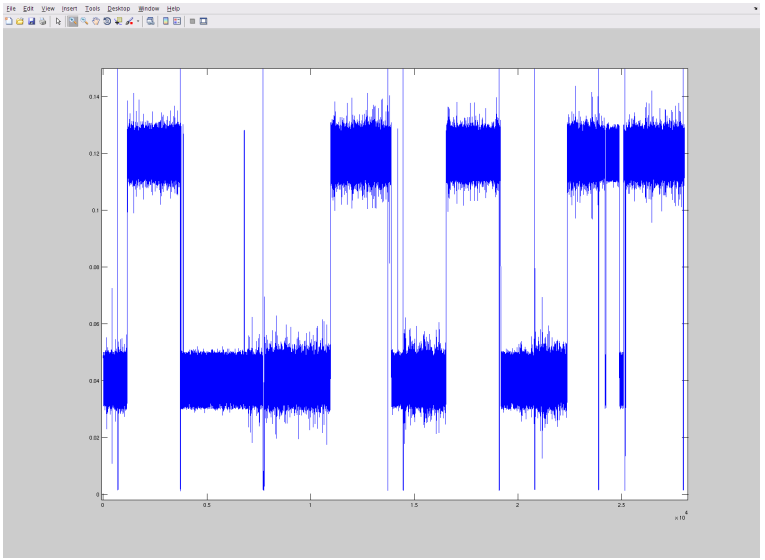
09/15/2016

Presentation at Loo lab at UCLA

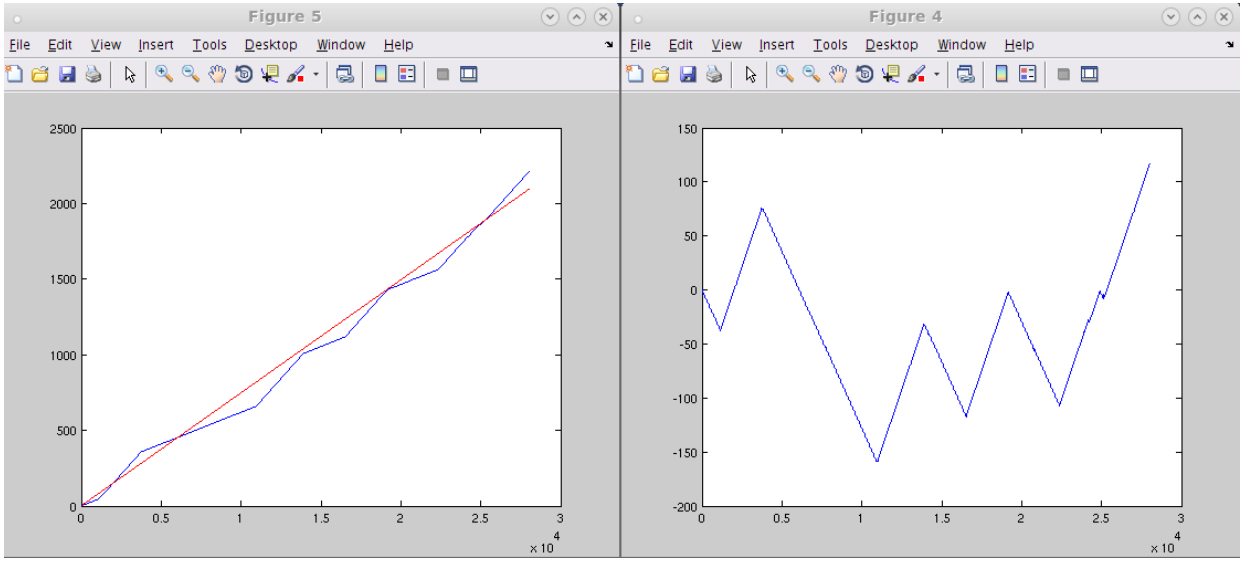
# Background and Problem

- LSL-time-stamped raw data do not have perfectly regular intervals between sampling points.
- load_xdf() has ‘HandleJitterRemoval’ option with default being ‘true’—that makes global linear interpolation on all sampling points.
- The assumption of this solution is that the raw data have more or less regular sampling intervals and their irregular deviation is infrequent and minor.
- However, real data showed that sampling intervals are much irregular and sometimes even fluctuates structurally. When the default linear interpolation is applied, the time stamps are severly distorted, up to +/- 150 sec in one of the past cases (see next slide).

# An example from tic data

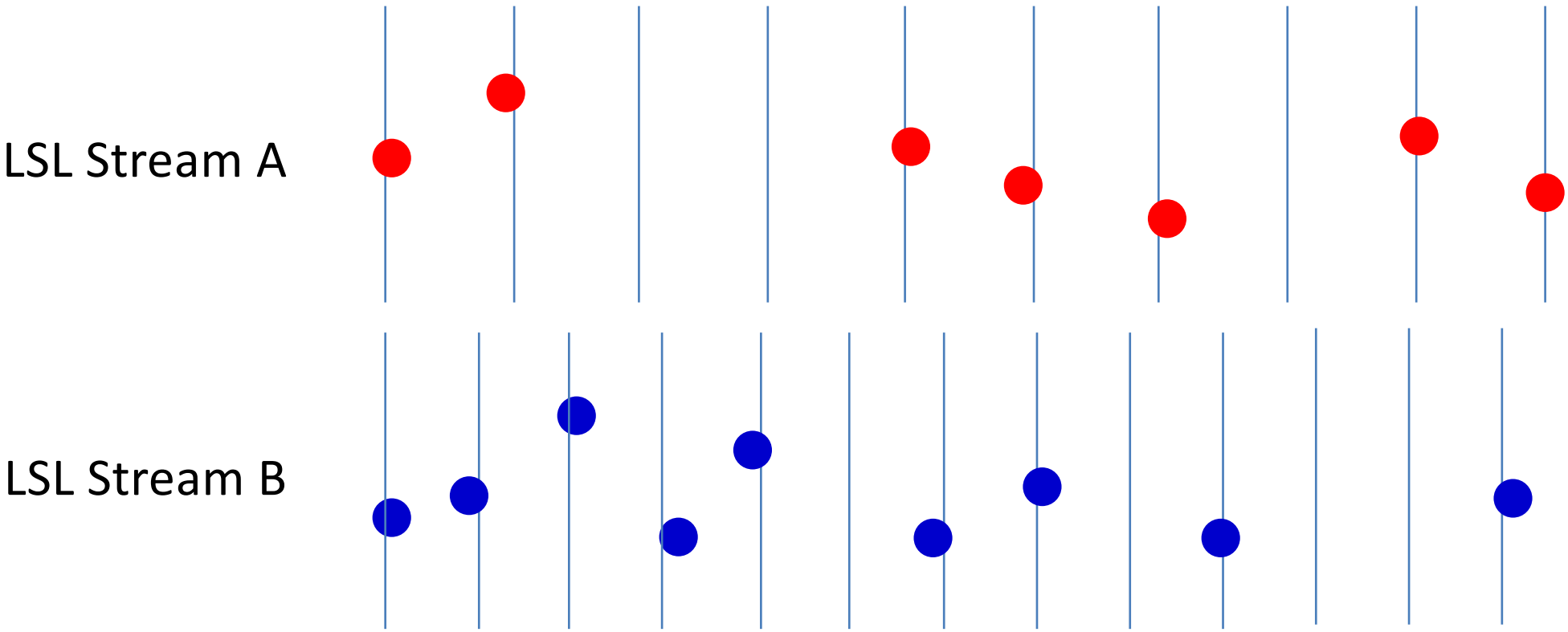


Interval of time stamps in one of LSL streams. It oscillates between two states, 40ms and 120 ms.




A result from a linear interpolation. Although global error is minimized successfully, the result showed maximum 150 second deviation locally.

# Technical detail of this problem

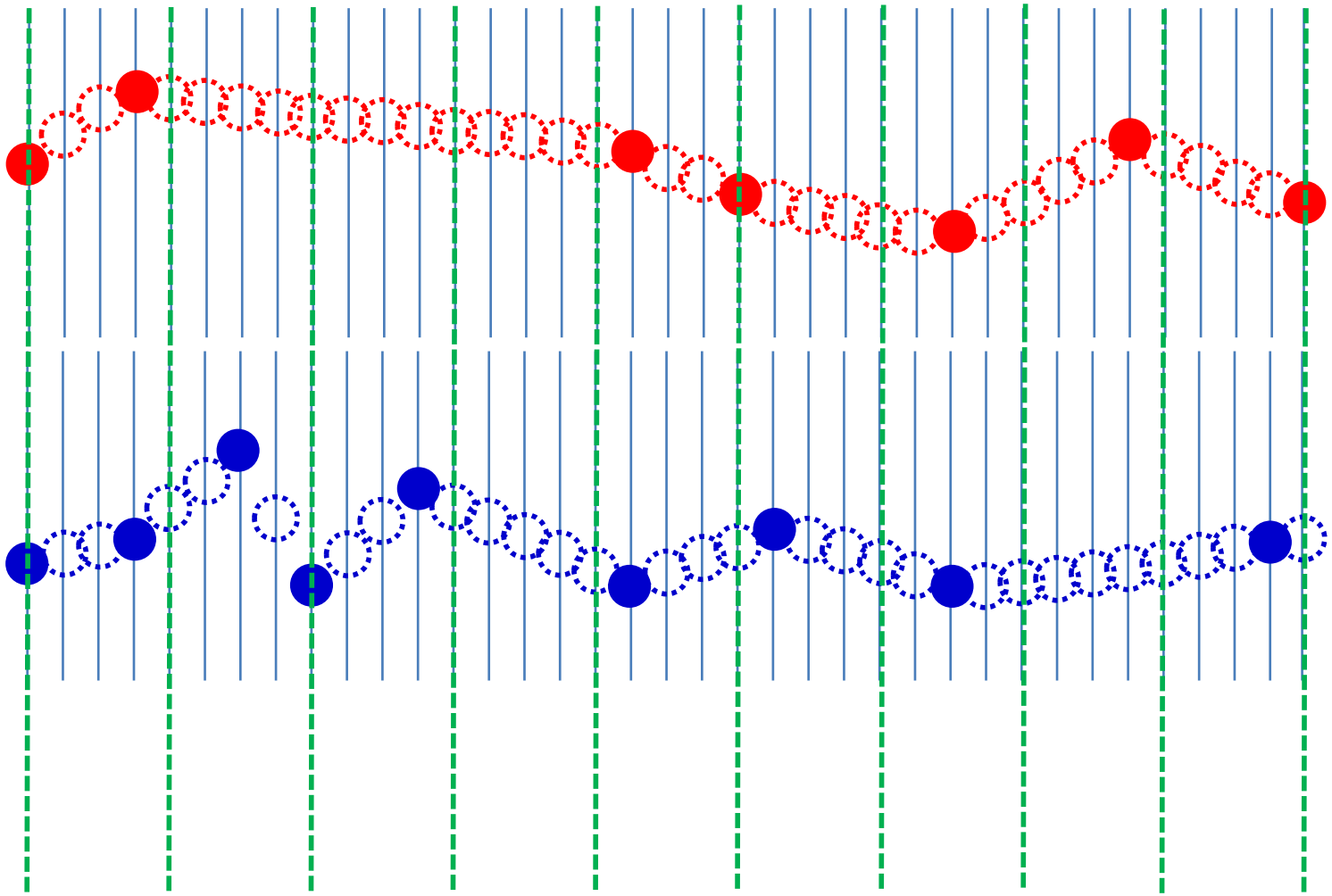



1. The sampling rates (vertical bars) are different across streams.
2. The data jitters and are not in pefect sync with sampling points.
3. There are missing data points (i.e. when jitters are too large).

# Suggested solution

LSL Stream A
*Upsampled and interpolated*

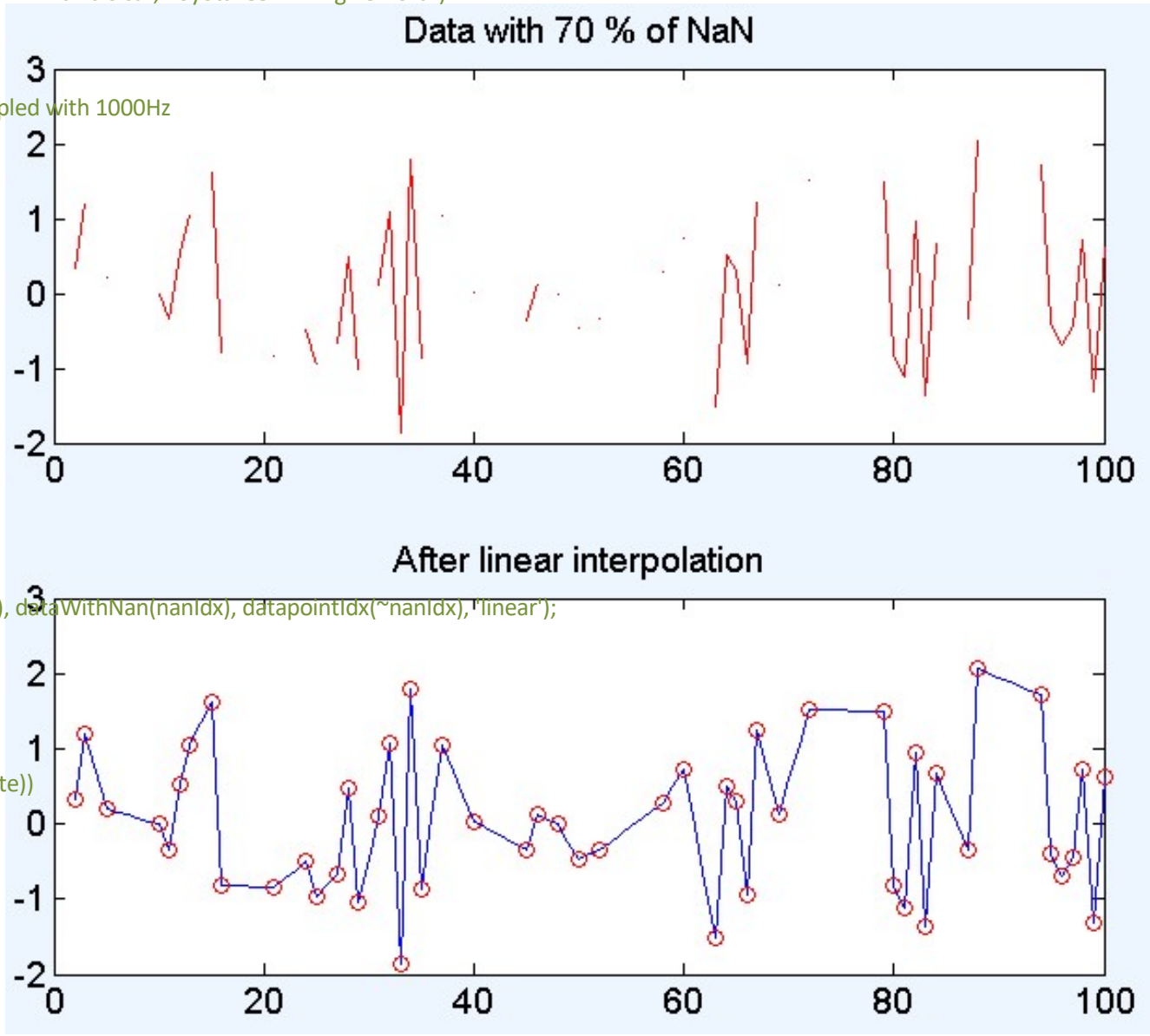
LSL Stream B
*Upsampled and interpolated*



New Sampling Points

# Matlab code to efficiently perform this interpolation (demo)

```
% 09/15/2016 Makoto. Created. UCLA visit today (I'm in David's car, Toyota '85 Driving I-5 north)

% In read data, perform the following first
% 1. Round up time stamps into millisecond
% 2. Prepare the same length (ms) of NaN data sampled with 1000Hz
% 3. Map the data onto the NaN data prepared
% 4. Perform the following interp1(..., 'linear')

% Prepare data with 20% of NaN
nanRate = 0.70;
data = randn(100, 1);
replaceNanIdx = randi(100, 100*nanRate, 1);
dataWithNan = data;
dataWithNan(replaceNanIdx) = NaN;

% Obtain data length
datapointIdx = 1:length(dataWithNan);

% Interpolate NaN using interp1
nanIdx = ~isnan(dataWithNan);
dataNoNan = dataWithNan;
dataNoNan(~nanIdx) = interp1(datapointIdx(nanIdx), dataWithNan(nanIdx), datapointIdx(~nanIdx), 'linear');

% Visualize results
figure; set(gcf, 'color', [0.93 0.96 1])
subplot(2,1,1)
plot(dataWithNan, 'r')
title(sprintf('Data with %.0f %% of NaN', 100*nanRate))
subplot(2,1,2)
plot(dataNoNan, 'b')
hold on
plot(dataWithNan, 'ro')
title('After linear interpolation')

set(findall(gcf, '-property', 'fontsize'), 'fontsize', 14)
```



# Conclusion

1. The sampling rates (vertical bars) are different across streams.

   -> Updample all streams to 1000Hz.

2. The data jitters and are not in pefect sync with sampling points.

   -> Map the upsampled data onto 1000Hz regular grid.
   Empty grid points will remain as NaN.

3. There are missing data points (i.e. when jitters are too large).

   -> Perform linear interpolation for all NaNs.

- After these processes, one can safely crop a same time window from different streams for analysis.
- This approach fundamentally address the LSL time-stamp issues. This method must be implemented as a default import option, and the current global linear interpolation should be removed.