# Bayesian Noise and Jitter removal algorithm – MCMCNJ

This document describes the files that constitute Release 1 of the National Physical Laboratory's (NPL) MCMCNJ Software.

The software, developed in Python 3.6, is provided in the form of py-files. It is intended to be used within the multiagent framework for the FoF project:

<https://github.com/bangxiangyong/agentMET4FOF/tree/28635a7eb7c1d98814d8409e1f1b96bdc36bb3d1>

The MCMCNJ software is provided with a software licence agreement “FoF\_NJ\_DS\_L\_20\_017.txt” and the use of the software is subject to the terms laid out in that agreement. By running the Python code, the user accepts the terms of the agreement.

Enquiries about this software should be directed to Kavya Jagan ([kavya.jagan@npl.co.uk](mailto:kavya.jagan@npl.co.uk)), Liam Wright ([liam.wright@npl.co.uk](mailto:liam.wright@npl.co.uk)) or Peter Harris ([peter.harris@npl.co.uk](mailto:peter.harris@npl.co.uk)).

## Introduction to the algorithm

Many measurements important in manufacturing relate to dynamical systems evolving over time, such as temperature and humidity, motion, position, etc. Sensors may have access to an accurate clock but for multiple, spatially distributed sensors, it is essential that the sensors are working to a common timescale, for example, through traceability to UTC, so that the data they record can be analysed correctly. Accurate time-stamping can be an important diagnostic aid to establishing which cause-effect relationships are feasible, and which are infeasible, or providing information about the path of a signal (such as vibrations) traversing a structure.

In addition to the issue of ensuring that the data recorded by different sensors is synchronised, such data can be subject to timing errors - jitter, as well as noise in the measured signal. Pre-processing the sensor signals can reduce the effects of noise and jitter and correct for such timing errors. It is important to understand the uncertainty of the resulting pre-processed signals, and to be able to propagate those uncertainties through any subsequent processing or aggregation of the sensor signals.

This software implements an algorithm to reduce timing and noise effects in the data recorded by sensors in industrial sensor networks. A Bayesian approach is used to estimate parameters describing the levels of jitter and noise in the measured signal and parameters of a model for the underlying ‘true’ signal, which are used to provide estimates of the values of the true signal. Since the Bayesian posterior distribution does not take a standard form, inferences about the parameters are made based on a sample derived from the posterior distribution using a Metropolis-Hastings (MH) Markov Chain Monte Carlo (MCMC) method. An important benefit of using a Bayesian approach is that the uncertainties associated with the estimates of the parameters, and hence the estimated signal, are also obtained without having to perform further computation.

For further details about the algorithm, please refer to the paper [1].

## Structure of the software

This is a simple implementation of the agent framework to showcase the noise and jitter removal capabilities of our algorithm on ‘real-time’ data generation. A simplified data workflow is shown in Figure 1, showcasing the agent framework for noise and jitter removal algorithm purposes. Data can be passed either directly into the ‘NJ alg’ module or passed into the MCMC module beforehand. The resulting output for both processes is plots of the ‘clean’ signal after noise and jitter removal.

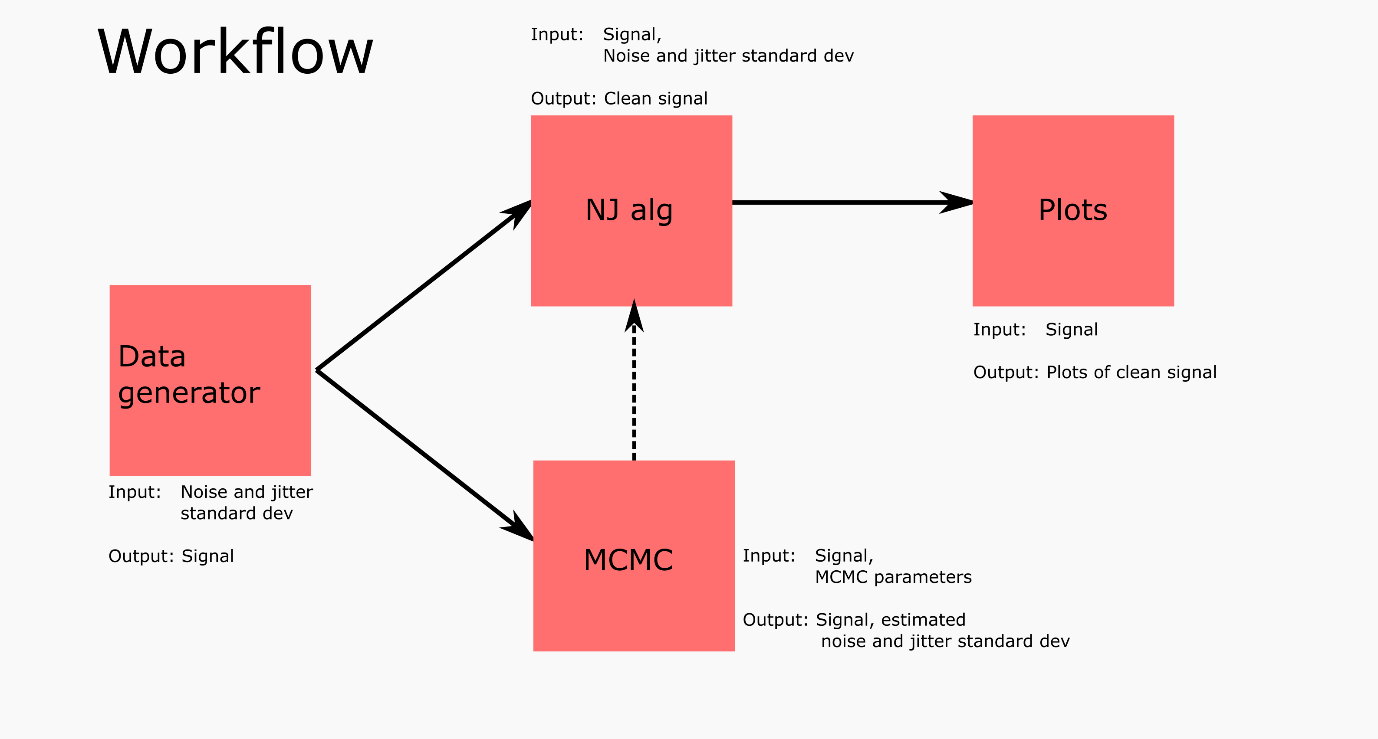


Figure 1: Simple workflow for noise and jitter modules within agent framework.

## Description of the functions

The toolbox consists of 3 python scripts described below. Inputs and outputs of functions are described in the header of each function in the code.

**NJRemoval\_class\_withmcmc.py -** Class containing a number of functions that perform noise and jitter removal. Save this file in a folder named “NJRemove”.

DecayExpFunction – Defines the decaying exponential function which has been used as a test case for the software

DecayExpFunction1der – 1st derivative of the decaying exponential function

DecayExpFunction2der – 2nd derivative of the decaying exponential function

MCMCMH\_NJ: Class containing the following functions:

* AnalyseSignalN - Analyse signal to remove noise and jitter providing signal estimates with associated uncertainty. Uses normalised independent variable
* NJAlgorithm – Iterative scheme that pre-processes data to reduce the effects of noise and jitter, resulting in an estimate of the true signal along with its associated uncertainty

random\_gaussian\_whrand – random draws from a Gaussian distribution based on transforming the output from whrand [2]. This was used to test the code and can be replaced by Python’s inbuilt random number generator if needed.

whrand - random draws from a uniform distribution based on using the Wichmann-Hill random number generator [2]. This was used to test the code and can be replaced by Python’s inbuilt random number generator if needed.

**mcmc\_decayexp.py –** Class that contains functions that generate mcmc samples from the posterior distribution of the noise and jitter variances. Save this file in a folder named “MCMCMH”.

mcmcci – Assesses convergence of multiple MCMC chains

mcsums – Summary statistics evaluated from MCMC samples

jumprwg – Gaussian random walk jumping distribution used to generate proposal samples for the Metropolis-Hastings algorithm

fgh\_cubic – Cubic function and its first and second derivative. Used for cubic approximation of the “true” signal

ln\_gauss\_pdf\_v – log of the Gaussian distribution

tar\_at – Target distribution from which we want to draw samples. In this case it is the log of the posterior distribution of the cubic parameters and the noise and jitter variances.

mcmcmh – Function that does the Metropolis-Hastings MCMC sampling and returns estimates of noise and jitter variance which is fed into the NJ removal routine

**Sinegen.py** - NPL addition to stream.py to add noise and jitter to sine generated data. Please copy and paste this function into stream.py.

**NJ\_with\_MCMC\_agent.py –** Main script that calls the above classes within the agent-based framework. It sets up agents that generate data and perform Bayesian noise and jitter removal.

## Acknowledgements

This project has received funding from the EMPIR programme co-financed by the Participating States and from the European Union’s Horizon 2020 research and innovation programme.

## References

[1] K. Jagan, L. Wright and P. Harris, "A Bayesian approach to account for timing effects in industrial sensor networks," *2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT*, Roma, Italy, 2020, pp. 89-94, doi: 10.1109/MetroInd4.0IoT48571.2020.9138266.

[2] Wichmann BA, Hill ID. Algorithm AS 183: An efficient and portable pseudo-random number generator. Journal of the Royal Statistical Society. Series C (Applied Statistics). 1982 Jan 1;31(2):188-90.

## Version of this document

Version 1 for release 1 of the code – Created 29-01-2021 KJ, LRW.