Time & Frequency statistics with AllanTools

Anders Wallin, anders.wallin@vtt.fi
BIPM Summer School, September 9-11, 2025

AllanTools – an open source (LGPL) python library for Time & Frequency statistics

- Development https://github.com/aewallin/allantools
  - Started around 2014, latest release ’2024.06’
  - Contributors: AW, Frederic Meynadier (BIPM), Erik Benkler (PTB), Yan Xie (VSL), and many others..
- Documentation https://allantools.readthedocs.io/
- Installation Packages
  - PyPi https://pypi.org/project/AllanTools/
  - Conda https://anaconda.org/conda-forge/allantools
- Why AllanTools?
  - Open source
  - Python widely used for scientific computations, automated scripts/plots etc.
  - Well (?) documented
  - The outputs of AllanTools are thoroughly tested against: Stable32 and SigmaTheta

Features

Function	Description	Comment
`adev()`	Allan deviation	Classic - use only if required - relatively poor confidence.
`oadev()`	Overlapping Allan deviation	General purpose - most widely used - first choice
`mdev()`	Modified Allan deviation	Used to distinguish between White and Flicker Phase Modulation.
`tdev()`	Time deviation	Based on modified Allan variance.
`hdev()`	Hadamard deviation	Rejects frequency drift, and handles divergent noise.
`ohdev()`	Overlapping Hadamard deviation	Better confidence than normal Hadamard.
`pdev()`	Parabolic deviation	Estimate uncertainty of Omega-counter data
`totdev()`	Total deviation	Better confidence at long averages for Allan deviation.
`mtotdev()`	Modified total deviation	Modified Total deviation. Better confidence at long averages for modified Allan
`ttotdev()`	Time total deviation	
`htotdev()`	Hadamard total deviation	
`theo1()`	Theo1 deviation	Theo1 is a two-sample variance with improved confidence and extended averaging factor range.
`mtie()`	Maximum Time Interval Error	
`tierms()`	Time Interval Error RMS	
`gradev()`	Gap resistant overlapping Allan deviation	
`gcodev()`	Groslambert Covariance	Improved three-corner-hat analysis

(your contributions here! e.g. dynamic Allan variance, etc.)

- Noise generation
- Real-time ADEV/MDEV
- Utility functions
- Tests
  - Typical data from OCXO, Cs, H-maser(?)
    (todo: optical clock!)
- Masks (pass/fail testing)
  - e.g. ITU-T ePRTC

Alternatives

- Stable32 (https://www.wriley.com/#products)
  - Windows executable https://ieee-uffc.org/technical-committees/frequency-control/educational-resources/frequency-control-software-layout
  - No soure code (?) ☹, No development ☹
  - GUI application, no scripting(?)
- SigmaTau
  - https://gitlab.com/fm-ltfb/SigmaTheta
  - Fracois Vernotte et al., Université de Franche-Comté, Besançon
  - CeCILL license http://www.cecill.info
  - C/C++ code, run from terminal or scripts
- Matlab
  - https://se.mathworks.com/matlabcentral/fileexchange/26659-allan-v3-0
  - https://se.mathworks.com/matlabcentral/fileexchange/26637-allan_modified
  - And probably many others… probably best to validate before use!

References

- NIST SP1065
  https://www.nist.gov/publications/handbook-frequency-stability-analysis
- Stable32 User Manual
  http://www.stable32.com/Manual154.pdf
- IEEE 1139-2022
  https://ieeexplore.ieee.org/document/9973001
- Enrico Rubiola's chart
  https://rubiola.org/pdf-static/Enrico's-chart-EFTS.pdf
  - Companion paper https://arxiv.org/abs/2201.07109
- More references at https://allantools.readthedocs.io/

	Features[1]	Tool: AllanTools	SigmaTheta	Stable32	TimeLab
Fractional frequency	ADEV	Y[2]	Y	Y[2]	Y
	GCODEV	beta	Y	–	–
	GRADEV	Y	–	–	–
	HDEV	Y[2]	Y	Y[2]	Y
	MDEV	Y	Y	Y	Y
	PDEV	–	Y	–	–
	Thêo1	Y	–	Y	–
	ThêoH	–	–	Y	–
	TOTDEV	Y	–	Y	–
Time	MTIE	Y	–	Y	Y
	TDEV	Y	–	Y	Y
	TIErms	Y	–	Y	–
Other	Estimator	direct	Bayes	direct	direct
	Error bars	LA & GR[3,4,5]	GR[4]	GR[4,5]	LA[3]
	Curve fit	–	Y	–	–
General	Type	Python lib	CL[6], API[6], GUI[6]	GUI[6]	GUI[6]
	OS[7]	L/M/W	L/M[8]/W[8]	L[9]/W	W
	License	LGPLv3+ (open)	CeCILL (open)	©[10] (free)	© (free)

(1) Here, "DEV" implies that both DEV and VAR are available
(2) Also, non-overlapped algorithm
(3) LA = Lesage-Audoin algorithm, based on stdev(σ^2)
(4) GR = Greenhall-Riley algorithm, based on $\chi^2(\sigma^2)$
(5) See main text
(6) CL = Command line, API = Application Programming Interface, GUI = Graphical user interface
(7) L = Linux, M = macOS, W = Windows
(8) API and GUI on Mac and Windows require recompiling
(9) Wine environment, or Windows virtual machine
(10) The team is working at cleaning the code for public release

Hands-on demonstrations

- Basic usage
  - Input (fractional frequency, or time interval data)
  - Plotting
- Noise Generation
  - ADEV for different noise types/colors
- Confidence Intervals
  - Noise identification is not automated (like in Stable32)

- Estimating (statistical) uncertainty for a frequency measurement
  - ”On the relation between uncertainties of weighted frequency averages and the various types of Allan deviations” Benkler et al. https://arxiv.org/abs/1504.00466
- Three-cornered-hat

beyond

the obvious

Thank You!

www.mikes.fi
Vttresearch.com

Python environment + AllanTools, installation

- https://www.spyder-ide.org/
- Installing AllanTools, from the spyder terminal
  - ”%pip install allantools”
  - OR:
  - ”%conda install allantools”

These should (probably) be installed by default?
- numpy
- scipy
- matplotlib

Test if allantools is installed ok?

```
In [36]: import allantools

In [37]: print(allantools.__version__)
2024.04
```

Table B.1—Functional characteristics of the independent noise processes used in modeling frequency instability in time and frequency metrology

Description of noise process	Slope characteristics of log-log plot				
	Frequency domain		Time domain		
	$S_y(f)$	$S_\phi(f)$ or $S_x(f)$	$\sigma_y^2(\tau)$	$\sigma_y(\tau)$	$Mod\sigma_y(\tau)$
	α	β	μ	$\mu/2$	$\mu'/2$
Random walk FM	–2	–4	1	1/2	1/2
Flicker FM	–1	–3	0	0	0
White FM	0	–2	–1	–1/2	–1/2
Flicker PM	1	–1	–2	–1	–1
White PM	2	0	–2	–1	–3/2
Blue PM	3	1	–2	–1	–2

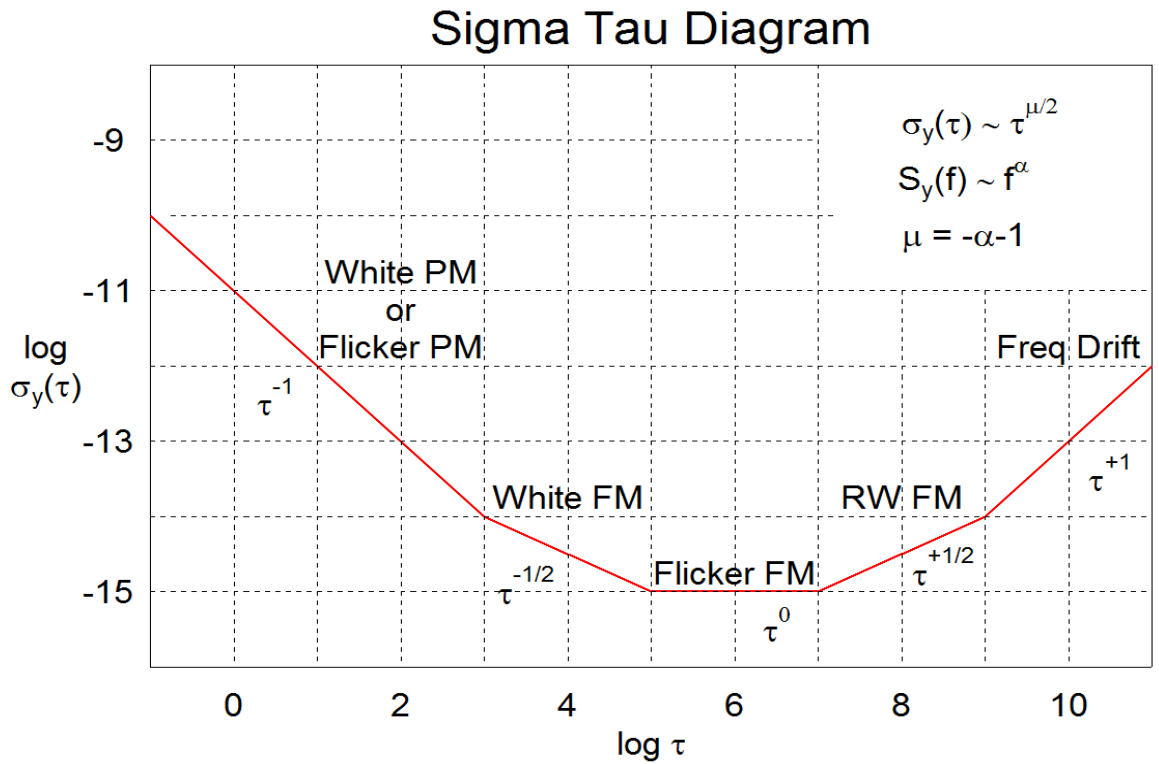
$$S_y(f) = h_\alpha f^\alpha \qquad S_x(f) = g_\beta f^\beta \qquad \sigma_y(\tau) \propto \tau^{\mu/2}$$

Colors of noise

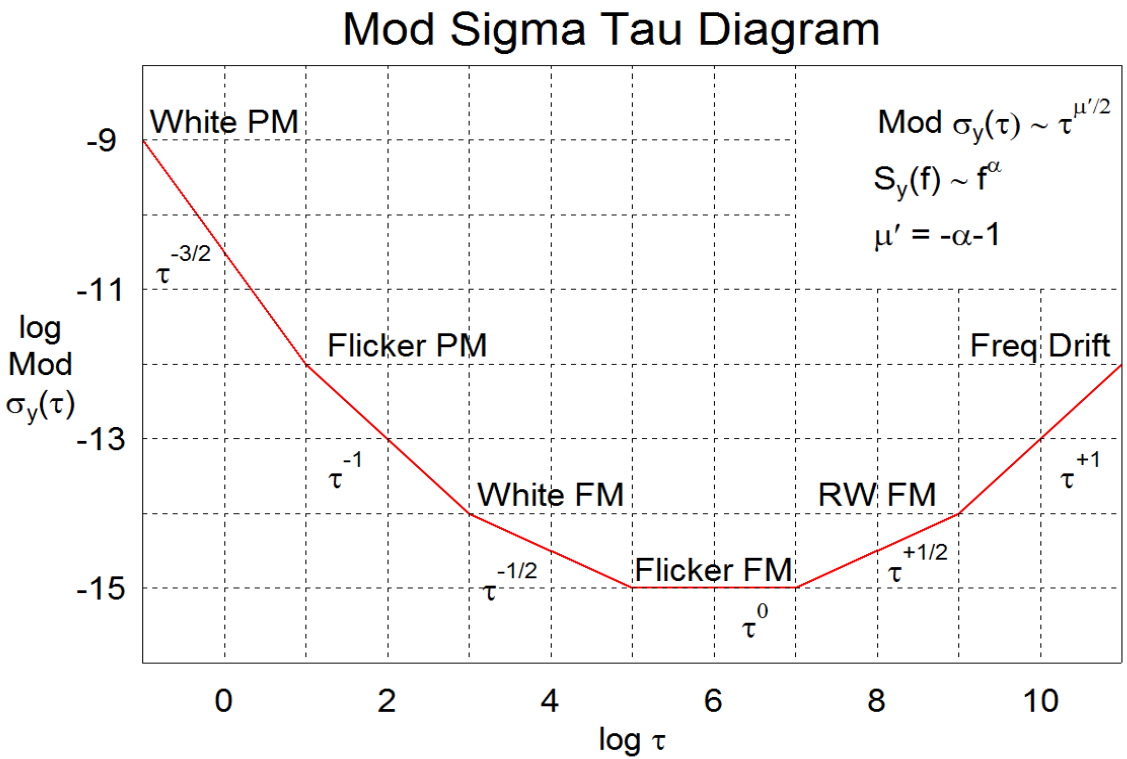
Noise Type	α	Stability Plot Noise Slope		
		ADEV	MDEV	TDEV
White PM	+2	-1	-3/2	-1/2
Flicker PM	+1	-1	-1	0
Random Walk PM or White FM	0	-1/2	-1/2	+1/2
Flicker FM	-1	0	0	+1
Random Walk FM	-2	+1/2	+1/2	+3/2

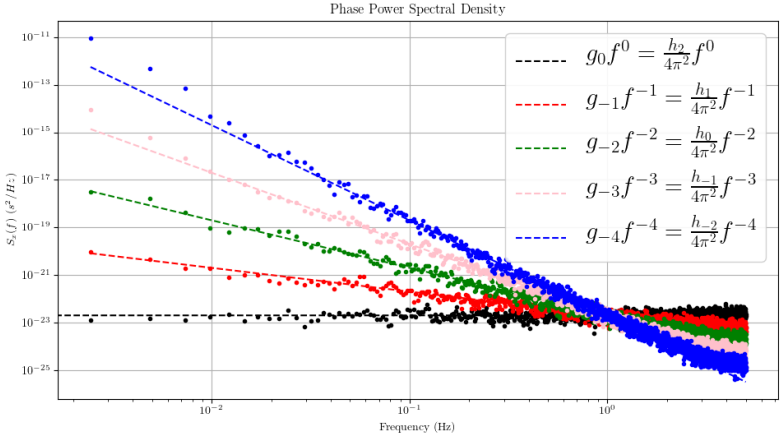
$$S_y(f) = h_\alpha f^\alpha$$

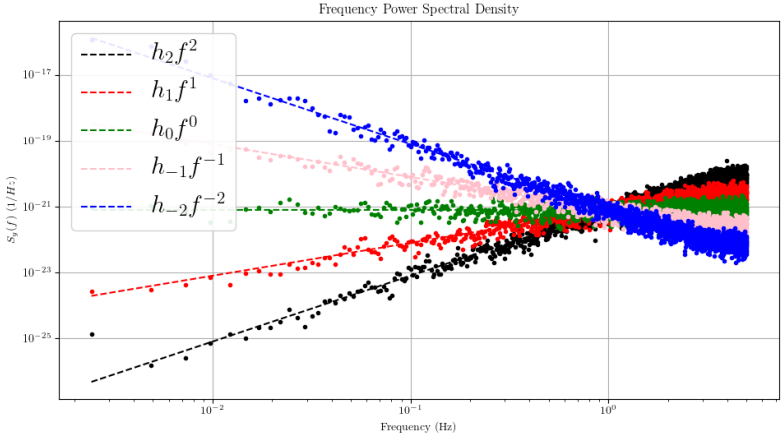
ADEV

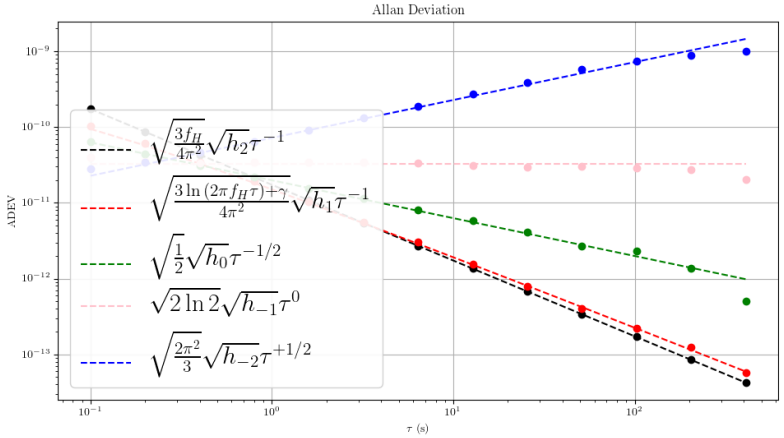

Sigma Tau Diagram
$\sigma_y(\tau) \sim \tau^{\mu/2}$
$S_y(f) \sim f^{\alpha}$
$\mu = -\alpha-1$
White PM
or
Flicker PM
Freq Drift
White FM
RW FM
Flicker FM
τ^{-1}
$\tau^{-1/2}$
τ^{0}
$\tau^{+1/2}$
τ^{+1}
log $\sigma_y(\tau)$
-9
-11
-13
-15
0
2
4
6
8
10
log τ

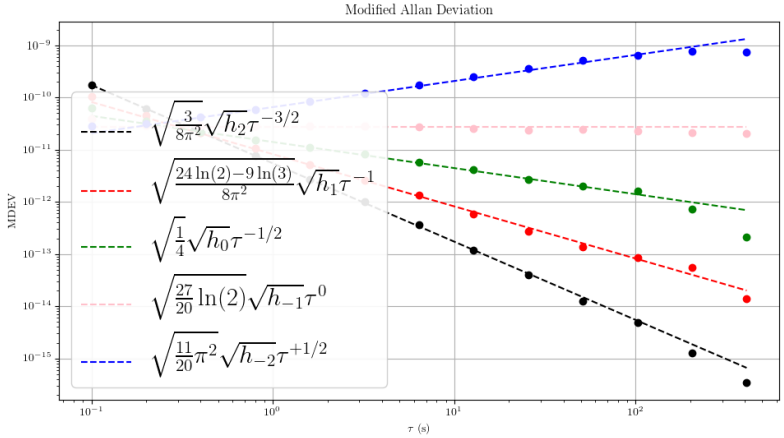
MDEV


Mod Sigma Tau Diagram
White PM
Flicker PM
White FM
Flicker FM
RW FM
Freq Drift
Mod $\sigma_y(\tau) \sim \tau^{\mu'/2}$
$S_y(f) \sim f^{\alpha}$
$\mu' = -\alpha-1$
$\tau^{-3/2}$
τ^{-1}
$\tau^{-1/2}$
τ^{0}
$\tau^{+1/2}$
τ^{+1}
log Mod $\sigma_y(\tau)$
-9
-11
-13
-15
0
2
4
6
8
10
log τ









https://github.com/aewallin/colorednoise

```
import allantools as at
import numpy as np
import matplotlib.pyplot as plt

N = pow(2,10)
dt = 10 # interval, in s, between data points, sample rate is 1/dt
f = 10e6 + 42*np.random.randn(N) # frequency in Hz
f0 = 10e6 # nominal frequency
y = (f-f0)/f0 # fractional frequency
x = np.cumsum(y)*dt # phase in seconds


#x = 1e-9*np.random.randn(N) # phase in s, sampled at 1 S/s
#y = np.diff(x)/10 # fractional frequency

(taus, devs, ers, ns) = at.oadev(x, rate=1/10, taus='all')
(taus2, devs2, ers, ns) = at.oadev(y, data_type='freq', rate=1/10, taus='all')

plt.figure()
plt.subplot(1,2,1)
plt.plot(x,'.')

plt.subplot(1,2,2)
plt.loglog(taus,devs,'o',label='OADEV, x')
plt.loglog(taus2,devs2,'.',label='OADEV, y')
plt.grid(which='both')
plt.legend()
```