Generalised Autocorrelation Function (GACF)

Updates 07/03/2018

Overview

- GACF basics
  - Lars’s paper
  - C++ implementation
  - Python wrapper
- Period extraction using FFT & peak detection
- Application to real NGTS data
- Noise threshold calculations

Lars's Paper

- https://www.overleaf.com/12550376jxwkqknjtqqp#/47821890/
- Introduction
- ACF
- GACF
  - Lag
  - Selection function
  - Weight function
  - Reduction to ACF for regular sampling
- Discussion (simple examples?)
- Conclusions

C++ implementation

- https://github.com/joshbriegal/GACF
- Branch: ‘pure-C++-code’
- Hardcoded file locations / generated data in main.cpp

Installation

Using CMAKE (https://cmake.org) is the easiest way to build the code. Code should be built into 'build' directory Example for UNIX based system:

1. Navigate to GACF/build
2. run "cmake .."
3. run "make"

Running "./GACF" will run the function main() from main.cpp

File Structure

```
.
├── CMakeLists.txt
├── build
├── include
│   ├── Correlator.h
│   └── DataStructure.h
└── src
    ├── Correlator.cpp
    ├── Correlator.h
    ├── DataStructure.cpp
    └── main.cpp
```

$$W(\Delta t)_{frac} = \frac{1}{1 + {}^{\Delta t}/_{\alpha}}$$

$$W(\Delta t)_{Gauss} = e^{-\frac{(\Delta t)^2}{2\alpha^2}}$$

- Selection functions – Fast (not good) or Natural
- Weight functions – linear or half-Gaussian with length scale. Definitions above differ slightly to current paper definition

C++ implementation details (non-examinable)

- DataStructure object
  - Read in time series, data points & errors
  - Calculate mean, median & normalised series

- Correlator object
  - Pointer to DataStructure object
  - Normalisation constant, max_lag, lag_resolution, alpha (length scale of weight function)
  - CorrelationData contains lag timeseries & correlation values

- CorrelationIterator object
  - Handles each lag time step of the correlation
  - Shifted time series, selection indices, time differences & weights
  - Returns one correlation value and one lag time step

Python Wrapper

- https://github.com/joshbriegal/GACF
- Branch: 'master'
- Uses pybind11 to expose pure C++ code
- Pip installable (not on PyPI yet)
- Returns a dictionary:
- {'lag_timeseries':[x], 'correlations': [x]}

Installation

Only requirement for installation is CMAKE (https://cmake.org). From above top level run

```
pip install ./GACF
```

in python:

```
from GACF import *

correlation_dictionary = find_correlation_from_file('filepath')
```

OR

```
correlation_dictionary = find_correlation_from_lists(values, timeseries, errors=None)
```

with options:

```
max_lag=None, lag_resolution=None, selection_function='natural', weight_function='gaussian', alpha=None
```

Examples

function_import_sine_wave_test.py creates a randomly sampled sine wave and finds the autocorrelation using the created functions when importing GACF

objects_test_from_file.py exposes the underlying c++ object structure to find the correlation of a timeseries from file

Code ’TODOs’

- More robust file reading (currently only accepts files in one format as tab delimited columns)
- More robust error handling in C++ (e.g. empty time series causes exit code 11 segmentation fault!)

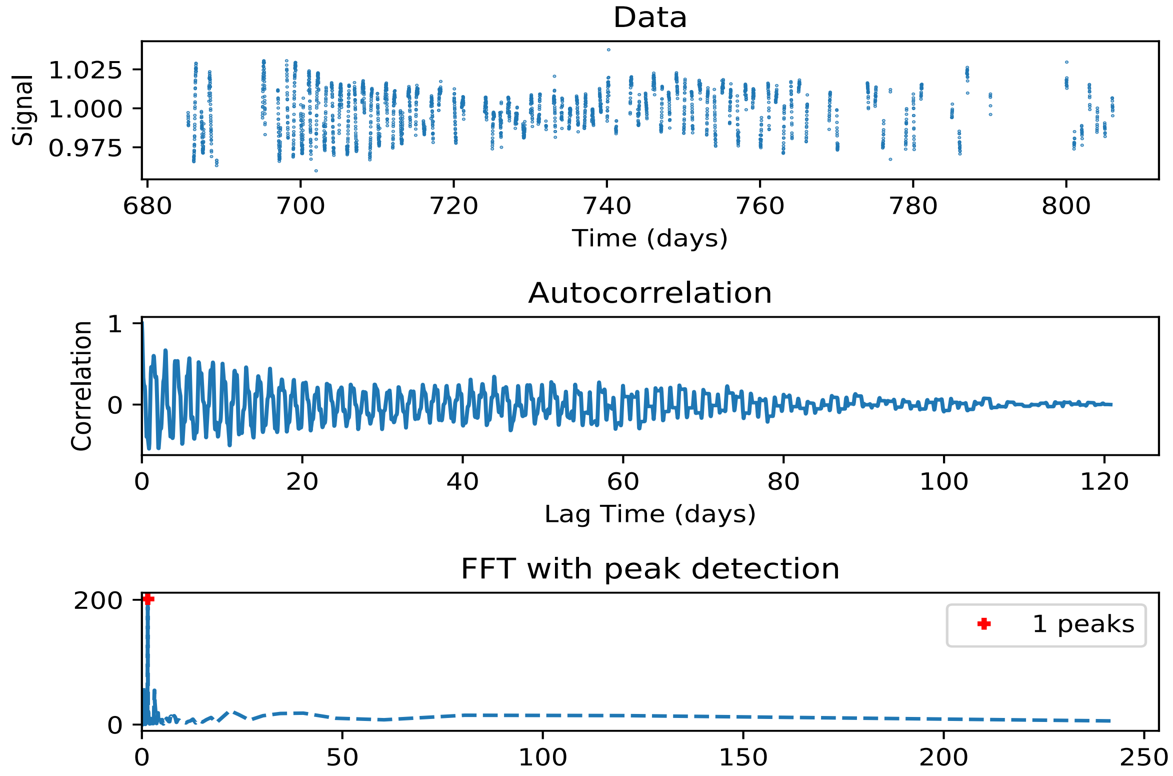
Performance:

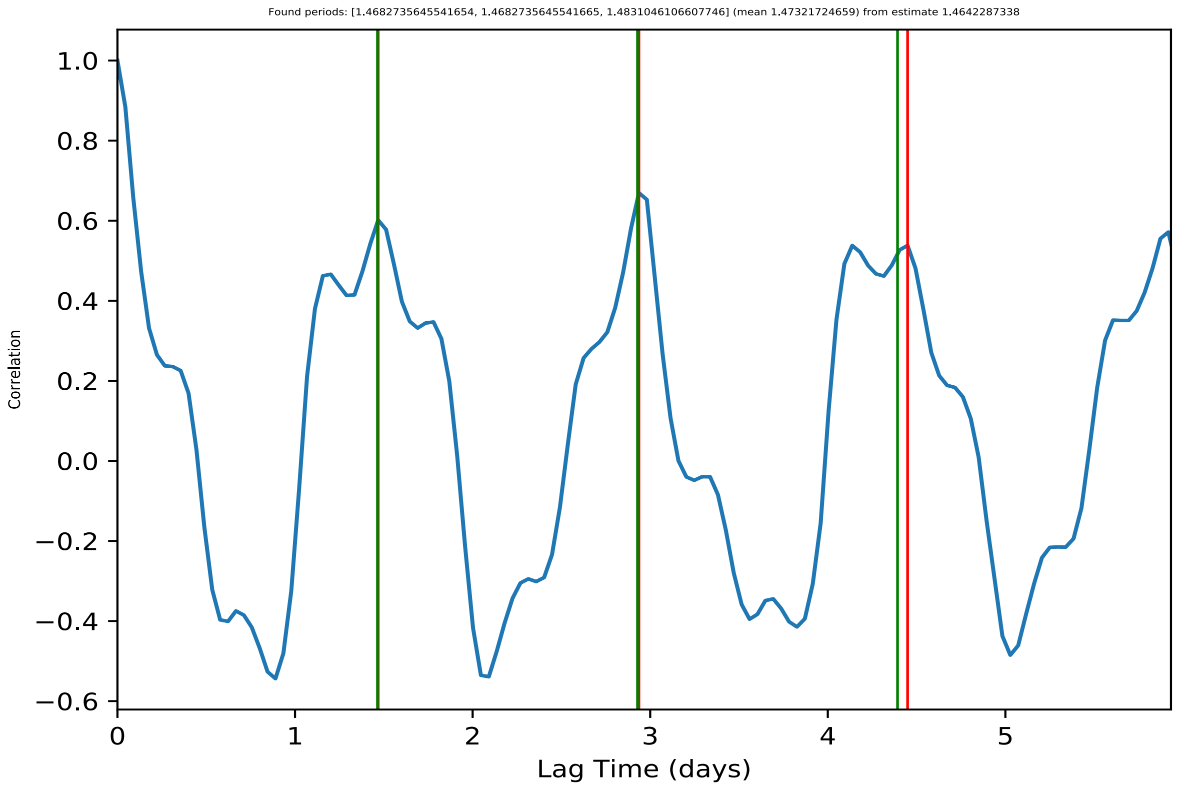
- 0.0631980895996 seconds for 984 data points / lag time steps
- 3.32579088211 seconds for 9985 data points / lag time steps
- 0.231755018234 seconds for 9982 data points / 700 lag time steps
- 66.3709959984 seconds for 199,981 data points / 8448 lag time steps (5 min resolution on 12 second cadence NGTS light curve)
- Time to do the rest of the stuff 1 - 4 seconds dependent on number of points

Period Extraction using FFT & peak detection in Python

- FFT
  - Fast & accurate periodicity detection for regular time series
- Peakutils
  - Extract peaks from data. Requires threshold (% of max peak) and width (no of data points) -> FFT point density not uniform in period
- Period interpolation
  - Take extracted period from FFT, fit to autocorrelation on first 3 periods and average

Period Extraction using FFT & peak detection in Python (e.g. 0409-1941_009529_LC_tbin=10min)





periods before pruning: [1.4642287337978166]
peak periods: [1.4642287337978166]

peak amplitudes: [201.3234954473777]
peak percentages: ['100.0%']
interpolated periods: [1.473217246589702]

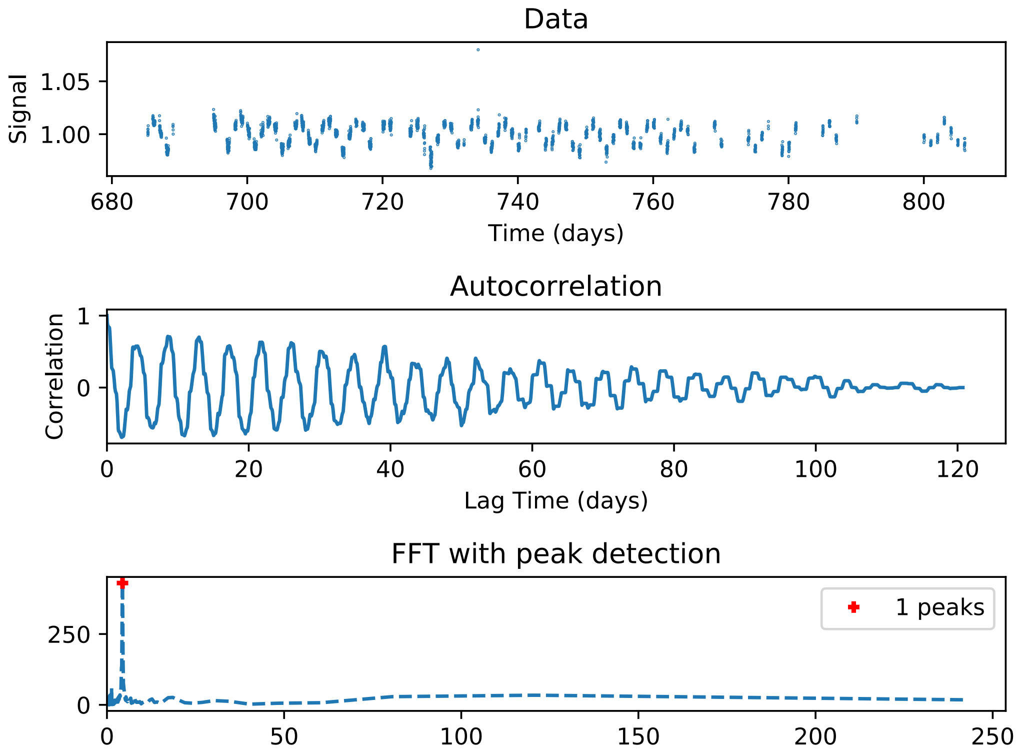
Application to NGTS data

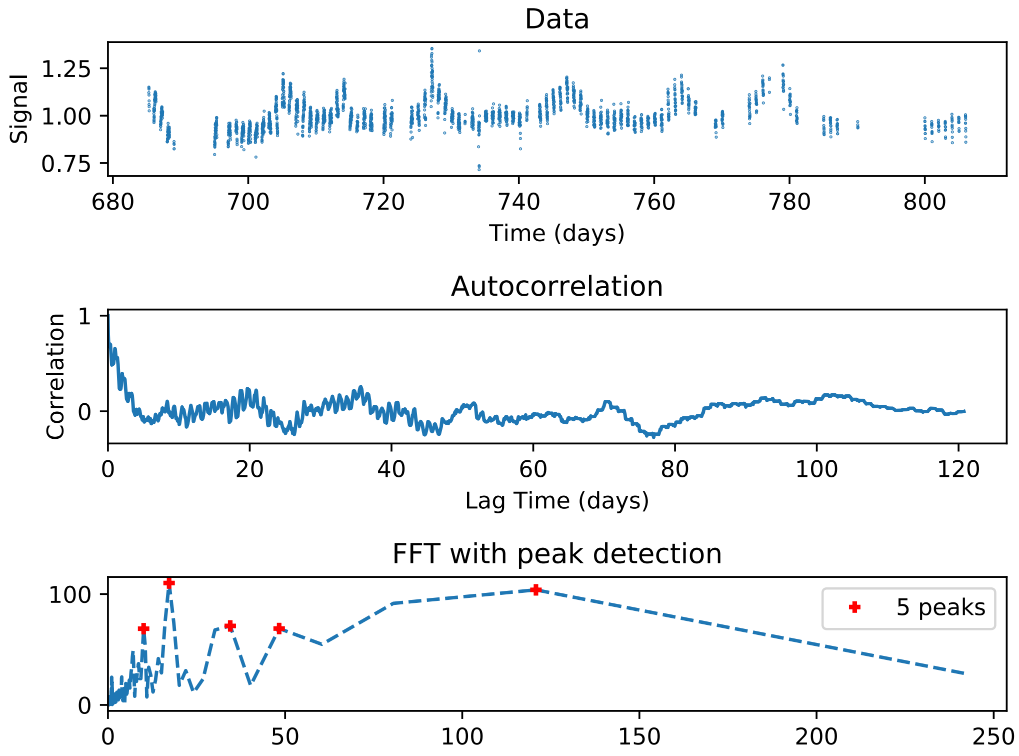
- Ran on ~ 100 light curves from TEST18, field 0409-1941
- Light curves binned to 10 minutes (median flux)
- When compared to Vedad's extracted LS periods we see:
  - For obvious periods (e.g. prev example) LS will give harmonics, GACF does not
  - For less obvious periods (e.g. below) both methods give nonsense

```
Object 9859
   Vedad periods: [[8.671760265098161, 2.1816488110477303, 4.335880132549081]] (double [1])
   GACF periods: [4.31582959204128]

Object 11035
   Vedad periods: [[0.4996555774966044, 1.0550831078330438, 1.1627757505199026]] (double [1])
   GACF periods: [17.25695161, 120.79866128, 34.51390322, 48.31946451, 10.06655511]
```

Application to NGTS data

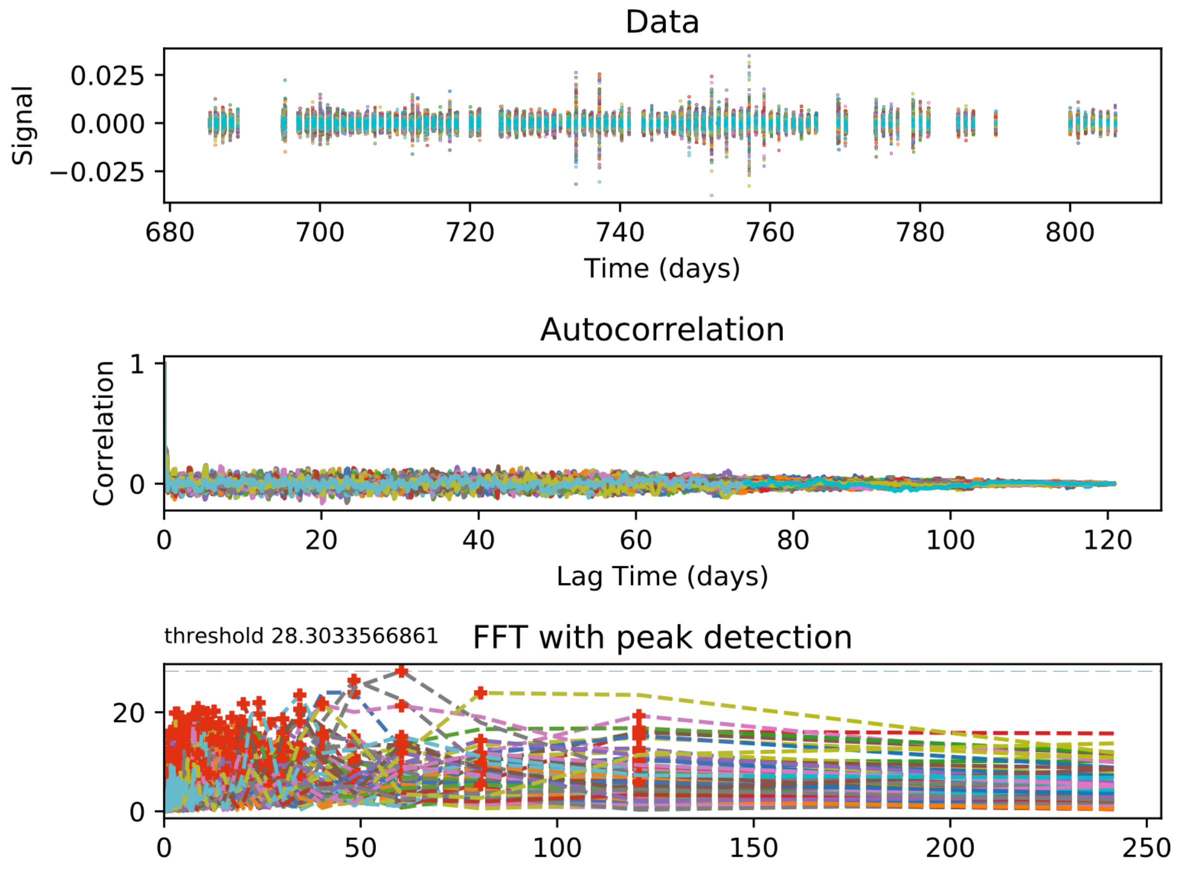
```
Object 9859
   Vedad periods: [[8.671760265098161, 2.1816488110477303, 4.335880132549081]] (double [1])
   GACF periods: [4.31582959204128]

Object 11035
   Vedad periods: [[0.4996555774966044, 1.0550831078330438, 1.1627757505199026]] (double [1])
   GACF periods: [17.25695161, 120.79866128, 34.51390322, 48.31946451, 10.06655511]
```





Noise threshold calculations

- Take error from binning (median stdev per bin, σ)
- Generate a number of 'noise signals'
  - Draw from Gaussian at each time point $X(t) \sim N(1, \sigma)$
- Calculate GACF of 'noise signal'
- Extract peak information
- Threshold = max peak from all samples



Noise threshold calculations

- Problems:
  - Need many samples – slow to calculate
  - Does the peak size match the same noise if a signal is present?
    - Signal to noise ratio
    - Shape of signal itself
  - Noise assumed uncorrelated
    - OK assumption given binned data?
    - OK assumption given speed? How would we consider red noise?

Noise threshold calculations

- Considered effect of noise signal on injected sine wave of different depths
- Conclusions unclear, not sure if worth pursuing