Aristotle University of Thessaloniki
Faculty of Sciences – Department of Physics
Section of Astrophysics, Astronomy and Mechanics

Diploma thesis

Search for weak periodicities in astronomical data using the FFT and FFA algorithms

Χρήση αλγορίθμων Fast Folding στην ανίχνευση ασθενών περιοδικών αστρονομικών σημάτων

Author: Konstantinos K. Kovlakas
Supervisor: John H. Seiradakis

Thesis grading commitee:

Professor John Seiradakis
Professor Manolis Plionis
Associate Professor Nikolaos Stergioulas

Thessaloniki, May 2013

This page intentionally left blank

To my family,
for your continued support and encouragement

This page intentionally left blank

Abstract

Nowadays, earth–based and space observatories participate in surveys of thousands or even millions of celestial objects. Often, the goal of such quests is the identification/classification of multiple targets by detecting a periodic phenomenon of intrinsic (e.g. Cepheid variable stars) or extrinsic nature (e.g. extra–solar planet transits), in their light curve. Processing numerous measurements for each one of the target is a time–consuming task and the use of "clever" algorithms is crucial.

The Fast Fourier Transform algorithm (FFT) is a standard method for analyzing time–series and is repeatedly used in surveys. Though, in cases of non–sinusoidal wave–forms (e.g. Pulsars), FFT can be found weak. An alternative method, Fast Folding Algorithm (FFA), was introduced by David Staelin (Staelin, 1969) in order to solve this problem.

We created implementations of both algorithms which we checked for accurate output and speed. Then we used them to estimate the period in time–series of various parameters (period, noise, etc.) that simulate the light–curves of various types of astronomical objects and compared the results. Also, real astronomical data were processed.

We confirmed the good function of the algorithms and the proposed complexity of FFT (by previous studies) and FFA (by us). We found that both algorithms detect the period in most cases. Unfortunately, the exploitation of the output of the algorithms was proved simplistic and failed to display the advantage of FFA over FFT for pulses of small duty cycles.

Finally, we propose corrections to the code and a period searching method that utilizes both algorithms.

This page intentionally left blank

Acknowledgments

I would like to express my very great appreciation to the supervising professor and inspiring teacher, John Seiradakis[1] for his helpful directions, constructive notes and the sincere interest he demonstrated on this thesis.

I'm also thankful for the contribution of my personal friends, John Mademlis[2] for his useful input in matters of information theory and signal processing, and Ellie Tzortzoglou[3] for willingly offering her editing services.

Last, but not least, I owe my deepest gratitude to Christina Noutsou for her patience and understanding during the last few months. This dissertation has made use of the

- NASA Exoplanet Archive, which is operated by the California Institute of Technology, under contract with the National Aeronautics and Space Administration under the Exoplanet Exploration Program
- SIMBAD database, operated at CDS, Strasbourg, France
- NASA's Astrophysics Data System (SAO/NASA ADS)

1 Department of Physics, Aristotle University of Thessaloniki
2 Artificial Intelligence & Information Analysis Laboratory, Aristotle University of Thessaloniki
3 School of English Language and Literature, Aristotle University of Thessaloniki

This page intentionally left blank

Contents

This page intentionally left blank

List of Figures

List of Tables

This page intentionally left blank

Nomenclature

Term	Explanation
ASAS	All Sky Automated Survey
ASCII	The American Standard Code for Information Interchange encoding scheme
BJD	Barycentric Julian Date
BKJD	Barycentric Kepler Julian Date, which is equal to BJD - 2454833.0 (0 at midday of 2009-01-01)
Dec	Declination
DFT	Discrete Fourier Transform
ETS	Evenly spaced time series (set of magnitudes accompanied by two time measures: observation offset and interval)
FFA	Fast Folding Algorithm
FFT	Fast Fourier Transform — in this text refers to the Radix–2 Fast Fourier Transform
GTS	General time series
J2000.0	Epoch 2000 January 1, 12h TT (JD 2451545.0 TT) at the geocenter. In this text refers to the celestial reference system defined by the mean equator and equinox of J2000.0

continued on next page

continued from previous page

JD	The continuous count of days beginning with the Julian day number 0 assigned to the day starting at Greenwich mean noon on 1 January 4713 BC
KID	Kepler ID: an identifier of each target of the Kepler Mission
KOI	Kepler Object of Interest
LSM	Least Squares Method
NASA	National Aeronautics and Space Administration: the space agency of the USA
SAO/NASA ADS	The Smithsonian Astrophysical Observatory/-NASA Astrophysics Data System: a digital library
RA	Right ascension
SNR	Signal to noise ratio
SIMBAD	Set of Identifications, Measurements and Bibliography for Astronomical Data: a database managed by the Centre de Donées astronomiques de Strasbourg (CDS)
TT	Terrestrial Time
UST	Unevenly spaced time series: set of magnitudes and observation times

"2+2 ≃ 5"

Sheuer, Peter (German Astronomer)

1 Introduction

1.1 Properties of astronomical time series

Almost all of our information about celestial objects arrives in the form of photons[1]. The astronomical datasets are distributions of photon properties: position ($\vec{r}$), time (t), direction (e.g. geocentric equatorial coordinates: α, δ), energy (E) and polarization (Stokes parameters)(Horne, 2012). For example, an astronomical photograph is a distribution of energy (=frequency=color) and direction (position of each pixel in image) of the photons that were detected by the photographing device.

In many cases, the above parameters are studied as they evolve in time. E.g. a movie is the evolution of the distribution of the previous example and each frame corresponds to an observation time. In general, a dataset

[1] nowadays there is also neutrino astronomy and the research for direct observation of gravitational waves is promising

of observations collected at successive points in time is called **time series** (Cowpertwait & Metcalfe, Cowpertwait & Metcalfe).

The great importance of astronomical time series may not need highlighting: since antiquity, the measurement of time was based on the observation of the changes in position of celestial objects (Sun, Moon or even Venus.) Furthermore, some scholars point the study of patterns in astronomical time series as the source of the rationalism and modern science (Zakai, 2009).

Evenly vs. unevenly spaced data

Measurements taken in equally spaced intervals (e.g. every hour or every April 11th) present a clearer view of the observation subject than unevenly–spaced time series. Also, many analysis methods require such uniformity (like the implementations of FFA and FFT in this work.) Thus, time series are often considered evenly–spaced and are the goal of the great majority of observations and experiments in many fields of science.

Constant sampling period is a considerably difficult goal in observational astrophysics. In the case of ground–based surveys, the observation times are restricted by various parameters: climatic (e.g. cloudiness), orbital (e.g. daylight) or even social (e.g. off–schedule aeronautical activity.) Consequently, time series at equidistant intervals can be produced in only few cases (e.g. pulsar observations in radio frequencies are unaffected by daylight, most of climatic features and often minutes or hours of observation are more than enough.)

In contrast, space telescopes, due to the absence of climate (although one should take account of the solar and cosmic radiation) and the reconfigurability of their orbital parameters, can perform regular–in–time measurements.

Mathematical notation

We will use calligraphic capital letters ($\mathcal{A}, \mathcal{B}, \ldots$) to represent time series or functions performed on them.

- **Unevenly spaced time series (USTS):** a finite mathematical sequence of N vectors, each corresponding to a magnitude (x_i) of a certain quantity and an observation time (t_i):

$$\mathcal{X} = (x_i, t_i)_{i=0}^{N-1} \tag{1.1}$$

  where $t_{i+1} > t_i$.

- **Evenly spaced time series (ESTS):** a set of three objects: (i) a finite mathematical sequence of N magnitudes (x_i) of a certain quantity, (ii) the observation's time offset (τ_0) and (iii) the sampling period (p_s). Consequently each magnitude's was measured at $\tau_i = \tau_0 + i \cdot p_s$.

$$\mathcal{Y} = \{(x_i)_{i=0}^{N-1}, \tau_0, p_s\} \tag{1.2}$$

- **General time series (GTS):** For simplicity, when the offset and sampling period does not interest us, we will denote equally spaced data as

$$\mathcal{Z} = (x_i)_{i=0}^{N-1} \tag{1.3}$$

1.2 Noise

An intrinsic property of every measurement is uncertainty. A time series of observations can be considered as a digital signal with noise. We will use an example to better illustrate this fact.

In order to estimate the magnitude of a star using an optical telescope, one will measure the amount of energy of photons that arrived from a certain solid angle that contains the target, over a specific period. This energy is not yet representative of the magnitude of the star. In the solid angle there are many other targets: terrestrial atmosphere, interplanetary and interstellar

medium diffusing light, undetectable targets etc. Also, photons are discrete quantities and therefore, they induce quantum noise and their detection is of probabilistic nature. Furthermore, thermal photons are always detected because of the presence of electronic circuits.

To overcome these difficulties, a series of data reduction techniques are applied to transform the measurements into meaningful information. But even then, the numerical methods we use and the accuracy of our computers induce an amount of uncertainty.

Normality

The distribution of 'counting statistics' or similar, like counting the photons in astronomy, is the Poisson distribution. Though the involvement of numerous physical processes and the collection of large datasets usually allow as to assume Gaussian distribution — the occurrence of which stems from the 'central limit theorem' (Wall & Jenkins, 2003).

Notation

Summarizing, complementary to the notation we described in Equation (1.1) and Equation (1.2), the sequence of measurements (x_i) will be considered as a summation of a digital signal (s_i) and noise (n_i) of Gaussian distribution:

$$x_i = s_i + n_i \tag{1.4}$$

A measure of the quality of the time series, is 'signal to noise ratio' (SNR). It is the ratio of the power of the signal over the power of the noise. The power $\mathcal{P}$ of a time series is defined as the summation of the squares of it's contents (Sabin, 2008):

$$\mathcal{P}\left(\mathcal{Z}\right) = \sum_{i=0}^{N-1} x_i^2 \tag{1.5}$$

$$SNR(\mathcal{Z}) = \frac{\sum_{i=0}^{N-1} s_i^2}{\sum_{i=0}^{N-1} n_i^2} \tag{1.6}$$

The biggest obstacle in determining the SNR is in the fact that in observed time series (as opposed to manufactured) we cannot distinguish the signal from the noise part. There are methods (statistical, information theoretic etc.) that deal with this problem, but they depend on interpretations and estimations of the data.

1.3 Periodicity in astronomy

One of the most important findings in a time series is periodicity, in the sense that it allows predictions and promotes the discovery of the underlying physical processes. Some examples of periodic astronomical phenomena are:

- Light from pulsating variables, rotating non–spherical stars, eclipsing binaries[1] (Figure 1.1)
- Extra–solar planet transits
- The 22–year magnetic activity cycle of the Sun (Figure 1.2)
- Radial velocity of binary stars or stars hosting massive planets
- Radio emission of pulsars

Of course, various types of variability — periodic or not — can coexist in an observation subject. Example of this is the δ Scuti variable star WASP–33 (or HD 15082) in which extra–solar planet transits were detected (Herrero et al., 2011).

[1] as opposed to eruptive variables like novæwhose variability does not follow a repetitive pattern

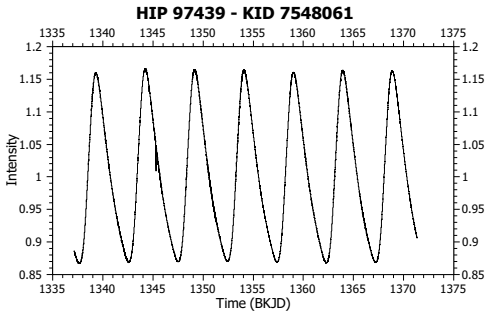


Figure 1.1: Light–curve of HIP 97439 (or KIC 7548061). It's a classical case of Cepheid with obvious periodicity (Thomson et al., 2013).

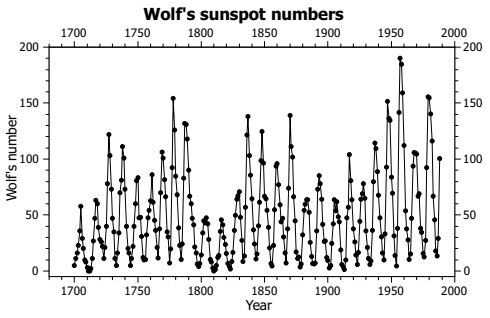


Figure 1.2: The Wolf's sunspot numbers indicate a periodicity in the solar magnetic activity (Hipel & McLeod, 1994)

1.4 Period searching

The periodicity in a light–curve like the one in Figure 1.1 may seem obvious through the naked eye but this is not always the case:

- High noise can hide periodicity (and other trends), especially when observing distant targets.
- The discrete nature of time series obscures the shape of the light–curves when sampling rate is comparable to the period.
- The presence of many periods can 'scramble' the light–curve in a seemingly random way.
- We need statistical measures of periodicity that can be automatically applied to the uncountable observation subjects of modern surveys (sometimes even millions.)

There are many approaches to the problem, but here we will discuss two autocorrelation[1] techniques: (i) Folding and (i) Discrete Fourier Transform.

[1] the process of comparing a signal/time series with itself shifted in time

1.4.1 Folding

If a time series contains a periodic signal then the periods will tend to be similar, but one should not expect them to be exactly the same for the reasons explained above (noise etc.). Correlating the data by superimposing the periods would result to a **folded time series** or **phase diagram** of length equal to the period. The folded profile would subsequently (as it will be illustrated later) contain an amplification of the signal and reduced noise in the phase bins.

SNR improvement

Let us have a look in a simple case of folding: two equally–length–ed sequences ($\mathcal{Z}'$ and $\mathcal{Z}''$) that represent two whole periods of the original data, added together to give a folded time series ($\mathcal{Z}$). Ideally, the signals are the same ($s'_i = s''_i$), but as the noise is considered random and independent of the signal (zero Pearson's correlation coefficient) it will differ and it's addition will not double it. In detail,

$$\begin{aligned} \mathcal{Z} &= \mathcal{Z}_1 + \mathcal{Z}_2 \\ x_i &= x'_i + x''_i \\ s_i + n_i &= s'_i + n'_i + s''_i + n''_i \\ s_i \approx 2s'_i &\text{ and } n_i \approx n'_i\sqrt{2} \end{aligned} \tag{1.7}$$

Actually, the noise n_i cannot be proportional to the noise part of any of the two sequences. The $\sqrt{2}$ part only represents the fact that when adding n random variables of Gaussian distribution with the same standard deviation, the resulting variable will originate from the same distribution but it's standard deviation will be amplified by $\sqrt{n}$, a consequence of the 'law of large numbers' (Spiegel, 1975).

If the period fits m times in the data, the overall improvement of the SNR

will be:

$$\frac{SNR_{fold}}{SNR_{data}} = \sqrt{m} \tag{1.8}$$

Period estimation

The period of the signal is in fact unknown and we can only apply the previously described process (§ 1.4.1) for trial periods. The method will give the best amplification in the case of close proximity of the trial period to the observed. Repetition of a wide range of trial periods can result to a good estimate of the period. This can be easily understood by a simple example:

Imagine a person repeating the first 8 natural numbers. The pauses between the numbers are 10 seconds. Thus, the period of the time series is 80 seconds.

If a second person enters and exits the room every 50 seconds, he will record an endless repetition of the sequence: "five, two, seven, four, one, six, three, eight"[1]. These numbers will seem randomly distributed to him and the sum of them will be 36. If he's late a few seconds (phase), he will end up with the same image and sum.

Now, a third person enters and exits the room every 80 seconds (equal to the period of the first person's signal.) Depending on whether he decides to start his 'invasions', he will record "one, one, one, ..." or "seven, seven, ..." etc. He will immediately perceive a repeating pattern and will calculate a sum of the numbers that will be either small or large (in respect to the 36 of the second person's) depending on his 'phase'.

The folding is the same process: we try entering the room in different intervals (folding in trial periods) and send a group of observers to account for all phases (phase bins). In each group, an observer records extreme values for the sum that represent the possibility of the group's period to be the equal to the signal period. A final report of all groups will lead to the best candidate for the period.

[1] The equality of the lengths of the two sequences and the maximum 'scrambling' is not a requirement but result of the coprimality of 5 and 8

Summary of the method

1. We use as an input the time series

$$\mathcal{Z} = (x_i)\,, \text{ where } i \in \{0, 1, \ldots, N-1\} \tag{1.9}$$

2. A starting trial period P is set

3. We fold the input into a new time series

$$\mathcal{F} = (f_j)\,, \text{ where } j \in \{0, 1, \ldots, P-1\} \tag{1.10}$$

4. Each phase bin f_j is calculated by the formula

$$f_j = \frac{\sum_{k=0}^{m_j-1} x_{j+kP}}{m_j} \tag{1.11}$$

where m_j is the number of folds performed. The normalization by this division is required because the trial period P cannot always perfectly devide N. Therefore, the value of m_j is not the same for all the phase bins (j) as we can fold, e.g. 8 data points into a 3 point series: the first and second bins would hold the average of three magnitudes (m_j=3) and the third bin, of 2 magnitudes (m_j=2). Consequently,

$$m_j = \begin{cases} \lfloor \frac{N}{P} \rfloor + 1 & \text{for } i < N\,(mod\,P) \\ \lfloor \frac{N}{P} \rfloor & \text{otherwise} \end{cases} \tag{1.12}$$

5. The power of the folded time series is calculated

6. The we select a new period (until a maximum period is reached) and goto step 2

7. After all periods have been tried, a power–period diagram can be constructed to give information about the signals in the data

8. If we are interested in the most powerful periodic signal (in accurately observed and processed curves may be a direct detection of variability), it's period is the one with the maximum power (peak period)

If straight–forward implemented, this algorithm has complexity (see Appendix A.1) of $O(n^2)$. The amount of operations needed for each of the trial periods is proportional to N: (i) addition of N numbers in groups, (ii) N reads of the data to compute the power of the fold. In blind searches, we check a range of periods that depends on the length of the data (e.g. from 2 samples to N/2 samples), the total of trial periods is proportional to N, hence the complexity of $O(n^2)$.

The folding technique is also practiced to depict the shape of a periodic signal, a 'profile'. Performing a fold using an estimate of the period (whatever the method for obtaining it) will enhance the signal, give an average (not always desired: there are semi–periodic physical processes) of many periods and shorten the length of data in case we work on a periodical phenomenon, speeding up the process.

1.4.2 Fast Folding Algorithm

The above brute–force period–folding can be significantly accelerated. A large number of additions performed for the calculation of phase bins (Equation (1.11)) is repeated. Inspired by Cooley and Tukey's Fast Fourier Transform (Cooley & Tukey, 1965), 4 years later, David Staelin created a similar algorithm, the Fast Folding Algorithm (Staelin, 1969), for computing the folds of time series avoiding the redundant additions.

Flow diagram

The function of FFA can be clarified by a short example. A time series of 12 elements is to be checked for a signal in the period range from 3 to 4 times the sampling rate (in short, "period of 3 to 4 elements".)

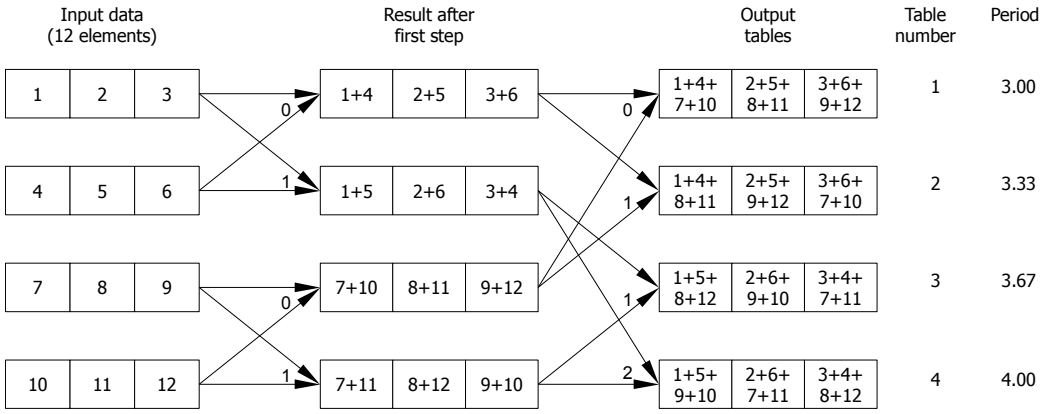


Figure 1.3: Flow diagram for FFA as applied to a time series of 12 elements. Notes: **(i)**. The numbers in the cells reveal the index of the element in the original time series, whereas the plus symbol (+) indicates additions of the contents of the respective cells, not their indices. **(ii)**. The arrows depict the flow of the information between each step. The integers, between each pair of arrows, represent the positions by which the elements of the second's arrow table, are circularly shifted to the right. **iii**. This is a reproduction of the first figure in Staelin (1969).

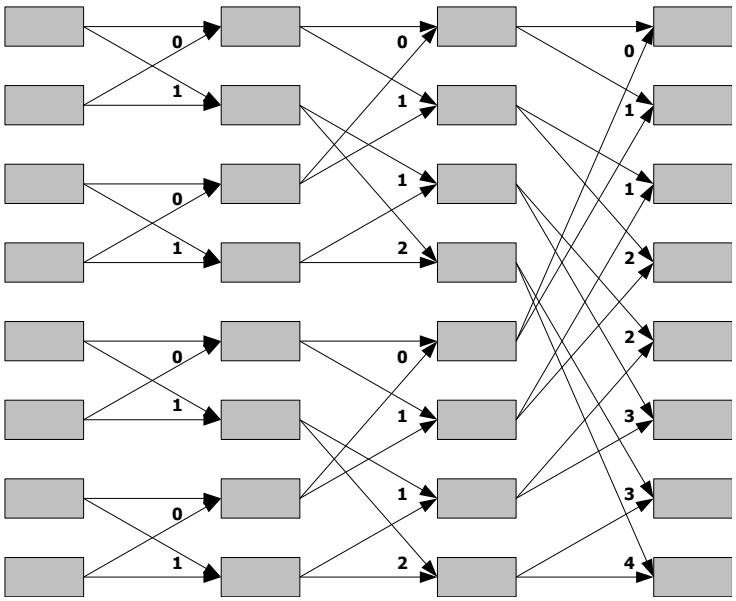


Figure 1.4: Flow diagram for FFA applied to N elements for basic period $P = N/8$

First, we need to fold the time series. The Figure 1.3 illustrates the transformation of the input data into four folds of it, each corresponding to a different period in the desired range.

In a series of steps, the algorithm adds the elements of pairs of tables from the original data (or the previous step's resulting data) and shifts them. Figure 1.4 shows the flow diagram in the case of 8 tables so that the reader will understand the selection of pairs and shift distances[1].

Application of FFA for n–length–ed data and basic period P_0

1. Grouping of the original data of n elements into M tables of P_0 elements each. As the tables are combined at pairs at constantly duplicating blocks of tables, M should be a power of 2, so we select the greater power we can:
$$M = 2^{\left\lfloor \log_2 \frac{n}{P_0} \right\rfloor} \tag{1.13}$$

2. The total elements that will be used, N, is
$$N = M \cdot P_0 \leq n \tag{1.14}$$

3. The total number of steps, S, is
$$S = \log_2 M = \left\lfloor \log_2 \frac{n}{P_0} \right\rfloor \tag{1.15}$$

4. M in count, tables (folds) will be formed : $\mathcal{F}_i$, where $i \in \{1, 2, \ldots, M\}$ each corresponding to a period $P(i)$:
$$P(i) = P_0 + \frac{i-1}{M-1} \tag{1.16}$$

[1] The shifts are shown between the arrows, not to be confused with the amplitude in common flow diagrams for digital signal processing

5. Now, the folds are ready to be normalized (e.g. divided by the number of summations/steps, S), analyzed or searched for maximum power as described in § 1.4.1

Complexity

In each step, there are N additions taking place, M shifts, etc. These operations are of $O(N)$ complexity (see Appendix § A.1). The number of steps is $\log_2 {}^{N}/_{P}$, so the total complexity is $O(NlogN)$.

Effectiveness for blind searches

To search for periodicity in a range of periods spanning more than one period, FFA must be applied as many times as the smallest difference between integers that enclose the desired range. For example, if we search for periods in the range $(5.67, 9.21)$ the algorithm should run 5 times — for trial period 5, 6, 7, 8, 9 and 10. The cumulative reports for all the folds can result to an estimate of the period (e.g. peak period.)

The complexity of a wide–range blind search using FFA is proven to be proportional to the square of the size of the input data ($O(n^2)$) — the same with the brute–force folding (see § 1.4.1). One may wonder why FFA is more effective.

The answer is hidden in the resolution of the methods. FFA folds the original time series not only in integer periods but also in numerous fractional periods. For example, for N=20, and restricting ourselves in the range of periods $[2, 6]$:

- **Brute–force folding**: 9 folds for trial periods 2, 3, 4, 5, 6, 7, 8, 9, 10 will be created
- **Fast folding algorithm**: the algorithm will run for 9 basic periods producing 30 folds (Table 1.1)

Some facts should be noted here. In Table 1.1 we can see that integer folding periods (except for the boundaries of 2.00 and 11.00) are approached

Table 1.1: Example of number of folds performed by FFA

Basic period	Periods of folds
2	2.00, 2.14, 2.29, 2.43, 2.57, 2.71, 2.86, 3.00
3	3.00, 3.33, 3.66, 4.00
4	4.00, 4.33, 4.66, 5.00
5	5.00, 5.33, 5.66, 6.00
6	6.00, 7.00
7	7.00, 8.00
8	8.00, 9.00
9	9.00, 10.00
10	10.00, 11.00

twice. Thus, only 22 different folding periods are resulted. Our experienced showed that the last fold of a run of FFA is an approximation (lower power) of the classical fold (see § 1.4.1) whereas the first fold of the next run (for the same folding period) is an accurate account of it. This dictates the inclusion of (i) all the duplicates in the statistics for period detection and (ii) the last run of FFA (10.00, 11.00) — although the range was reached in the previous run (9.00 and 10.00).

This means that an estimate of period with FFA can be more accurate and also, the improved resolution can be utilized by re–binning the data: the input size will be decreased and the n^2 will rapidly decrease in respect to the brute–force folding, whereas the resolution can still remain high.

Count of folds and period resolution

The total number of folds, $C_{FFA}(N)$ depends on the input size N and has a complex expression, because the requirement of each FFA run to apply on the largest power–of–2 section of the input, involves number theoretic functions.

Instead of a disciplined mathematical proof, we confirmed the following formulæ, by computational experiments:

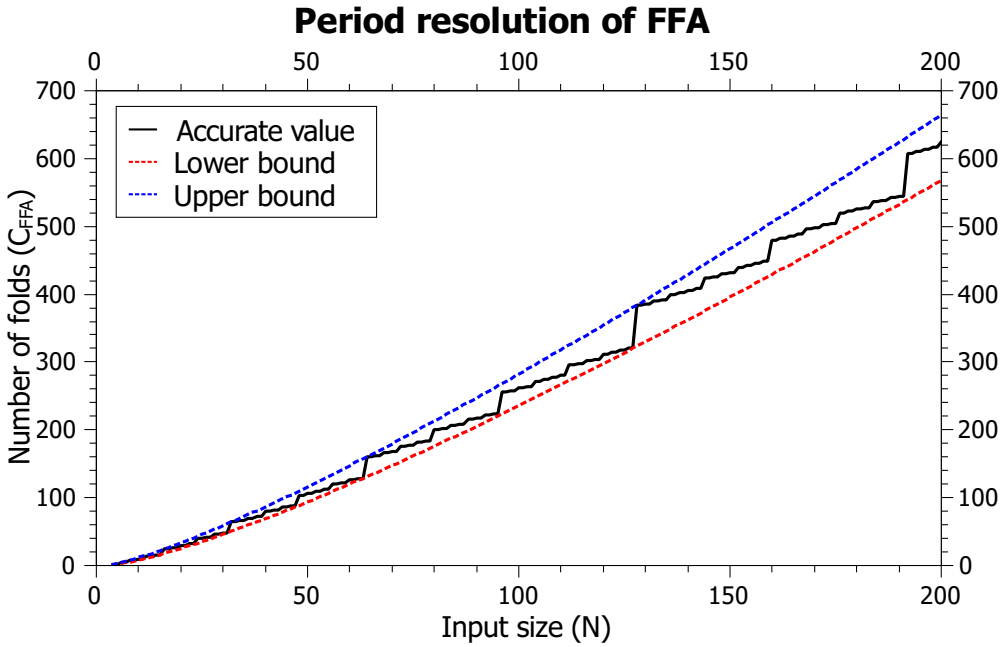


Figure 1.5: The value of C_{FFA}, (number of folds and consequently, number of folding periods), as a function of the input size. Note that the duplicates (see 'Effectiveness') are included.

$$C_{FFA} \simeq \frac{1}{2} N \log_2 N - \frac{3}{4} N \tag{1.17}$$

The exact value is bounded by two functions of N (Figure 1.5):

$$\frac{N+1}{2} \log_2 \frac{N+1}{4} \leq C_{FFA} \leq \frac{N}{2} \log_2 \frac{N}{2} \tag{1.18}$$

1.4.3 Fourier Analysis

The time series is a representation of data in time domain. It is also possible to represent the data in the frequency domain by performing the Fourier transform. Here some basic principles of Fourier Analysis will be highlighted in order to understand the function of FFT (Chu, 2008).

Fourier Series

A real periodic function $x(t)$ can be expressed as a sum of trigonometric series:

$$x(t) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty}\left[a_n \cos\left(\frac{\pi n}{L}t\right) + b_n \sin\left(\frac{\pi n}{L}t\right)\right] \tag{1.19}$$

where

$$a_n = \frac{1}{L}\int_{-L}^{L} x(t)\cos\left(\tfrac{\pi n}{L}t\right)dt \tag{1.20}$$

$$b_n = \frac{1}{L}\int_{-L}^{L} x(t)\sin\left(\tfrac{\pi n}{L}t\right)dt \tag{1.21}$$

Generalization for complex functions:

$$x(t) = \sum_{n=-\infty}^{\infty} c_n e^{i\frac{\pi n}{L}t} \tag{1.22}$$

where

$$c_n = \frac{1}{2L}\int_{-L}^{L} x(t)\, e^{-i\frac{\pi n}{L}t}dt \tag{1.23}$$

Continuous Fourier Transform

Continuous Fourier Transform (CFT) is a generalization of Fourier Series for infinite domains, enabling the representation of continuous non–periodic signals:

$$x(t) = \int_{-\infty}^{\infty} F(f)\, e^{-2\pi i f t}df \tag{1.24}$$

where

$$F(f) = \int_{-\infty}^{\infty} x(t)\, e^{2\pi i f t}dt \tag{1.25}$$

Discrete Fourier Transform

For an equispaced discrete signal (or evenly spaced time series) with size N, x_n where $n \in \{0, 1, \ldots, N-1\}$, the Discrete Fourier Transform (DFT), X_p where $p \in \{0, 1, \ldots, N-1\}$, is calculated by the following formula:

$$X_p = \sum_{n=0}^{N-1} x_n e^{-i\frac{2\pi}{N}np} \tag{1.26}$$

The Inverse DFT can also be computed:

$$x_n = \frac{1}{N} \sum_{p=0}^{N-1} X_p e^{i\frac{2\pi}{N}np} \tag{1.27}$$

Thus, an algorithm performing DFT would add and multiply N values, N times, resulting to a computational complexity of $O(N^2)$.

DFT: Magnitude and phase spectra

The result of the DFT is a series of complex numbers. The first element is the constant a_0, the DC offset. The next elements point to the information we have got on the harmonics. Their magnitude ($|X_i|$) is the amplitude of the corresponding trigonometric function and their argument $\left(\tan^{-1}\frac{Im(X_i)}{Re(X_i)}\right)$ is the phase (Sabin, 2008).

It is possible then, to reconstruct the original data from this information. As one can see in Figure 1.6, even using a small number[1] of the most powerful harmonics we approximate the square wave (which is a 'difficult' function to approximate.)

[1] In this way it is possible to use smaller data, when a desired precision is reached, making Fourier transform a powerful tool for lossy data compression

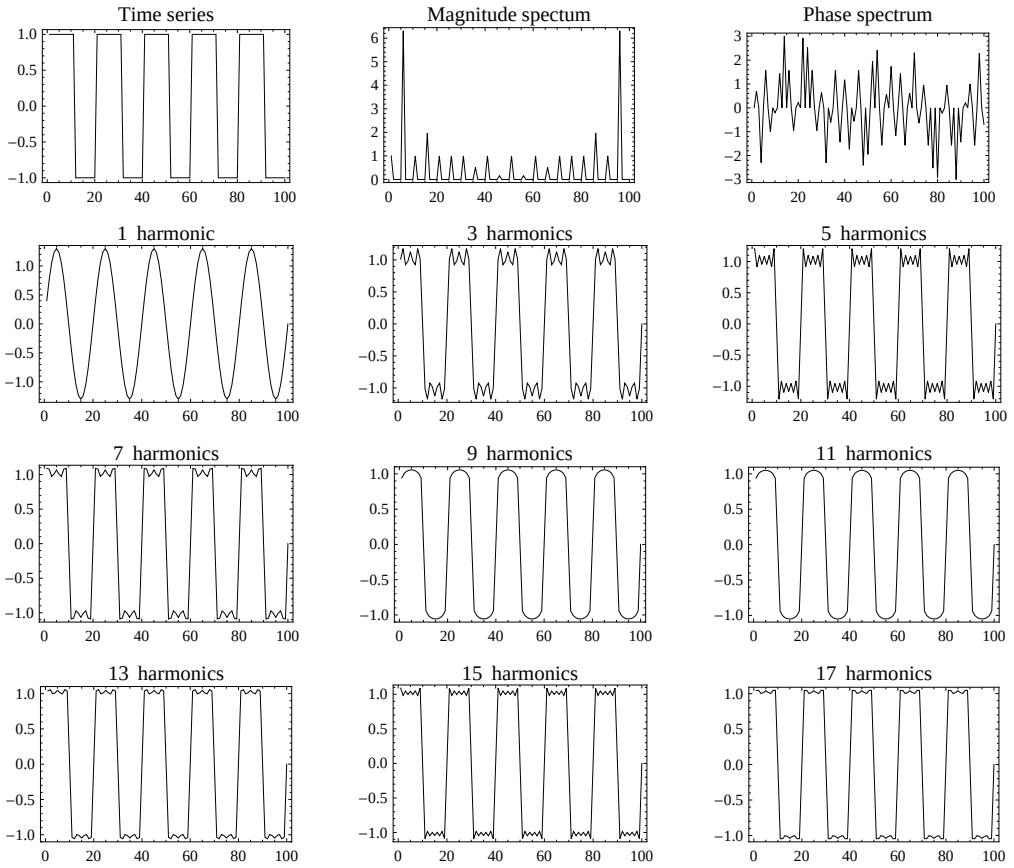


Figure 1.6: An example of the decomposition of a square wave signal (first row, first column) into it's spectrum. Both magnitude and phase spectra are shown (first row). As we add the harmonics the DFT gave, the summation approximates the original signal

Nyquist–Shannon sampling theorem

Harry Theodor Nyqvist and Claude Elwood Shannon independently discovered the 'sampling theorem' according to which,

> "The sampling rate required to exactly reconstruct a signal from its samples, the **Nyquist rate**, is more than twice the highest frequency at which the Fourier transform of the signal is non–zero, called **Nyquist frequency** or folding frequency."

When the Nyquist criterion is not met, the phenomenon of aliasing (shift on the spectrum) appears. Example of this is a common effect of wheel

spokes looking as rotating in the opposite direction of the car's movement when a certain speed is reached. The sampling rate of the video camera or of our eyes is not high enough to detect the actual period (reappearance of any of the spokes in the position we focus to.)

Because of this fact,

- the search for periodicities in a sequence of N elements is restricted to periods from 2 to N/2 time the sampling rate
- half of the Fourier coefficients of a time series can be discarded ($N/2$ to $N-1$)

Spectral leakage

Each element of the DFT, or 'frequency bin', of a signal corresponds to a multiple of a basic frequency. If the processed signal consists of only a sinusoidal waveform of equal frequency to one of the frequency bins, the DFT will give only one non–zero value: no other frequency will be detected. What happens in the case the frequency is not a multiple of the basic one? All of the DFT's elements get a non–zero value and a bell–shaped distribution is formed around the real frequency. Consequently, the maximum is located near the real frequency and the power is distributed to many frequency bins, lowering the total profile of the output (Figure 1.7). The phenomenon is called **spectral leakage**.

Matrix representation of DFT

The Equation (1.26) can be written in the matrix form

$$\begin{pmatrix} X_0 \\ X_1 \\ \vdots \\ X_{N-1} \end{pmatrix} = \begin{pmatrix} w^0 & w^0 & w^0 & \cdots & w^0 \\ w^0 & w^1 & w^2 & \cdots & w^{N-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w^0 & w^{N-1} & w^{2(N-1)} & \cdots & w^{(N-1)^2} \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{pmatrix} \quad (1.28)$$

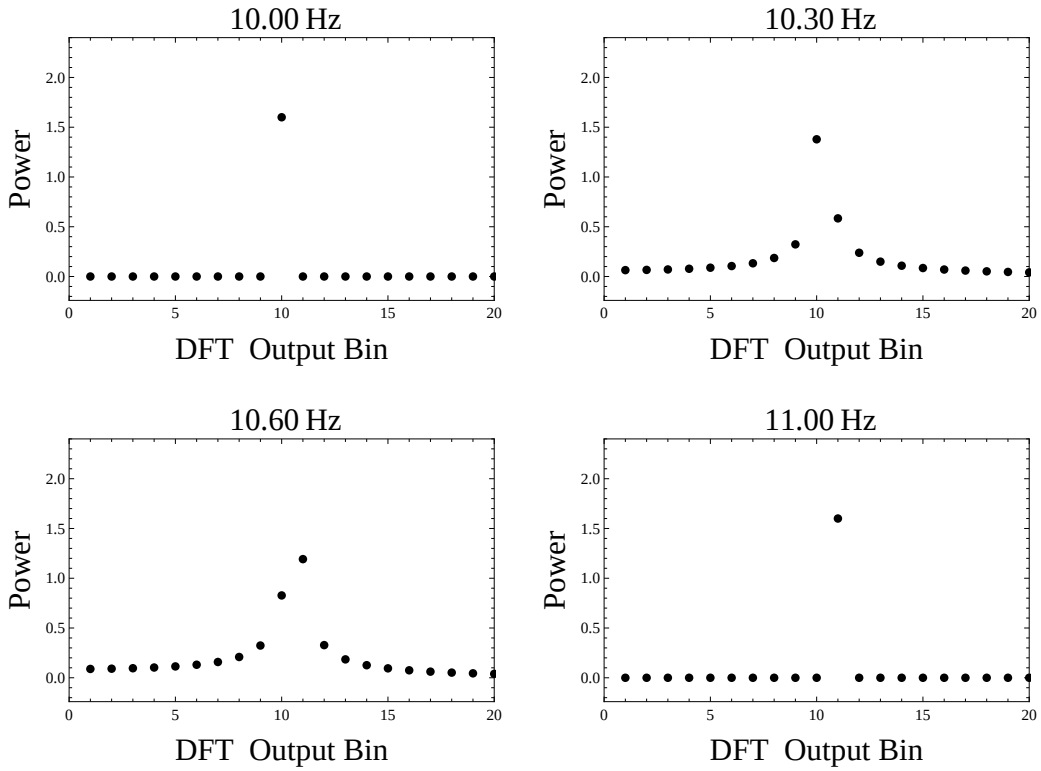


Figure 1.7: Example of the spectral leakage in a discrete Fourier transform. The four figures represent the DFT of a signal with a sinusoidal waveform of the frequency noted above their frames. The integer frequencies present no spectral leakage and coincide with the maximum–power frequency bin

where

$$w = e^{-i\frac{2\pi}{N}} \tag{1.29}$$

The symmetries of the middle array can taken advantage of, leading to optimized algorithms computing DFT, like the 'Radix–2 Fast Fourier Transform' that will be described here.

1.4.4 Radix–2 Fast Fourier Transform

Divide and conquer

The division of a problem into smaller sub–problems is often highly effective. Consider a simple search for a name in a phone book. Reading the indices one–by–one, the answer may come hours later. Instead, we all instinctively make a clever selection of the pages we look into. We open the book in a page, we check if the name is in the previous pages or the next and we continue the search in the appropriate 'half'. The search will be over in seconds.

The same technique can be applied to the computation of the DFT. When N is even, the Equation (1.26) can be split into two parts (even and odd n's):

$$X_p = \sum_{n=0}^{\frac{N}{2}-1} x_{2n} e^{-i\frac{2\pi}{N}(2n)p} + \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} e^{-i\frac{2\pi}{N}(2n+1)p} = \tag{1.30}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x_{2n} e^{-i\frac{2\pi}{N/2}np} + e^{-i\frac{2\pi}{N}p} \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} e^{-i\frac{2\pi}{N/2}np} = \tag{1.31}$$

$$= A_p + w^p B_p \tag{1.32}$$

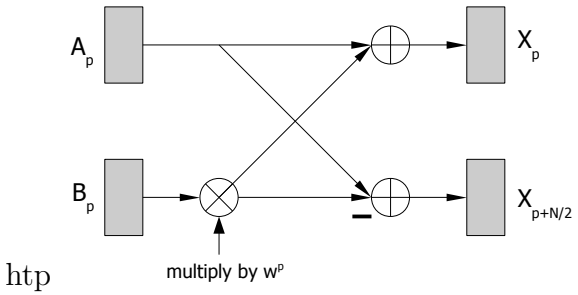
where the expression

$$A_p = \sum_{n=0}^{\frac{N}{2}-1} x_{2n} e^{-i\frac{2\pi}{N/2}np} \tag{1.33}$$

$$B_p = \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} e^{-i\frac{2\pi}{N/2}np} \tag{1.34}$$

are obviously the DFTs of the $\frac{N}{2}$–sized sequences

- $(x_0, x_2, \ldots, x_{N-2})$
- $(x_1, x_3, \ldots, x_{N-1})$



htp

Figure 1.8: The 'FFT butterfly diagram'. The equally sized data tables A_p and B_p are combined to give the even and odd–indexed elements of the DFT, X_p

The division of the problem into two smaller parts optimizes the algorithm because of the previously noted computational complexity ($O\left(N^2\right)$) for DFT. By computing two half–sized DFTs we reduced the operations by a factor of two: $\left(\frac{N}{2}\right)^2 + \left(\frac{N}{2}\right)^2 = \frac{N^2}{2}$.

Symmetry

Exploiting the fact that the DFT is periodic with period $\frac{N}{2}$, we are led to further simplification

$$X_p = A_p + w^p B_p \tag{1.35}$$

$$X_{p+\frac{N}{2}} = Ap - w^p B_p \tag{1.36}$$

The flow diagram of this process (Figure 1.8) is called 'FFT butterfly diagram' and reflects the combination of the results of two DFTs on divisions of the data, into one DFT of the whole data.

Radix–2: strategy and flow diagram

If $N = 2^M$, the divide and conquer strategy can be repeated M times, reducing the computation of the DFT into 2–point DFTs. As the data is separated to even/odd indexed elements and not sequentially, the input data must be shuffled appropriately. This is done by bit–reversal[1] the indices

[1] In FFT implementations, bit–reversal is directly or indirectly applied

(Table 1.2).

Table 1.2: Example of bit reversing of indices in data of size 8

Index (decimal)	Index (binary)	Bit–reversal	New index
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

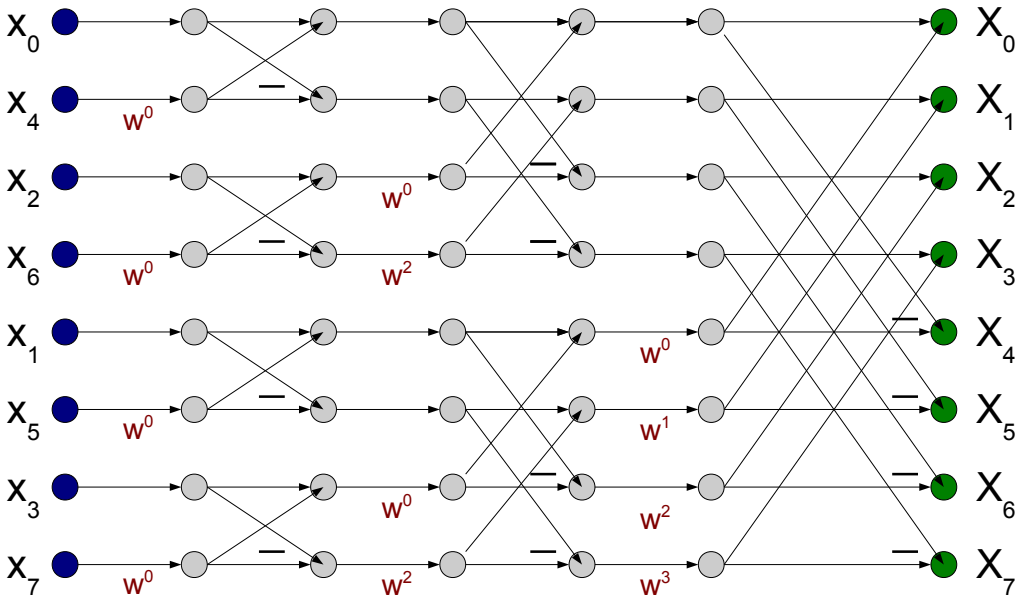


Figure 1.9: Flow diagram for FFT. The input data (blue) is scrambled (bit–reversing) and a series of 'FFT butterflies' lead to the DFT (green). Note that this diagram differs from the one in Figure 1.4: **i.** elements and not groups of elements are processed, **ii.** there are no shifts but multiplications by powers of w (red) or -1 (black minus signs)

The overall flow diagram of the Radix–2 FFT in the case of 8 elements is shown in Figure 1.9.

Historical Note

About two decades after the publication of the first Fast Fourier Transform (Cooley & Tukey, 1965) it was revealed that the famous German mathematician, Carl F. Gauss, discovered the FFT in 1805, even before the French mathematician, Joseph B. J. Fourier investigated Fourier Series (Heideman et al., 1985).

1.5 Properties of specific types of light–curves

In this section we will overview some basic properties of three types of periodically variable stars and their light–curves. The selection of these groups was not random as they seem to fit in two extreme cases and an intermediate one. One category is the **Cepheid variable stars** which are characterized by an approximately sinusoidal light–curve. Another, is the **hosts of extra–solar planets**, where short rectangular pulses are the observation target. As a 'bridge' connecting the characteristics of these groups, we consider appropriate the **eclipsing binary stars** who present sinusoidal or rectangular, short or long pulses — depending on their kind and properties.

1.5.1 Cepheid variables

General information

The group is named after its prototype - δ *Cephei* - and is of great historical importance in astronomy. The pulsation cycle of a Cepheid betrays its absolute magnitude (Cepheid period–luminosity relation) and combined with a measurement of the apparent magnitude we can estimate the distance of the star or the galaxy including it (Shu, 1982).

Cepheids are 'intrinsic variables' as their variability is caused by internal physical processes. Sir Arthur Eddington indicated a mechanism ('Eddington's valve') of pulsation in stars due to mechanical instability which was later used to explain the behavior of Cepheid stars. In layman's terms, these variables are like balloons, contracting and expanding (radial pulsation mode), changing their luminosity as their radiation interacts with their gas (Percy, 2007).

Light–curve properties

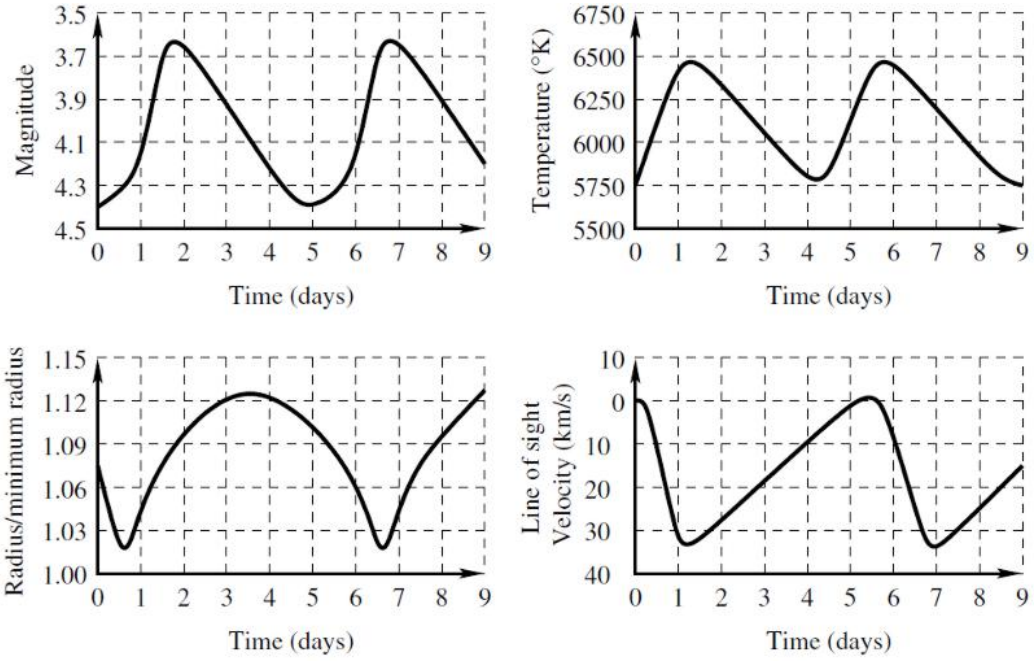


Figure 1.10: The magnitude (top left), temperature (top right), radius (bottom left) and radial velocity (bottom right) of δ Cephei (Percy, 2007).

The profile of a Cepheid resembles slanted sinusoidal peaks (Figure 1.10). The maxima and minima do not happen when the radius is at extreme values as temperature is also changed (due to energy production variations). Remember the standard formula

$$L = 4\pi R^2 \sigma T^4 \tag{1.37}$$

where L is the luminosity, R is the radius and T is the effective temperature of the star.

1.5.2 Eclipsing binary stars

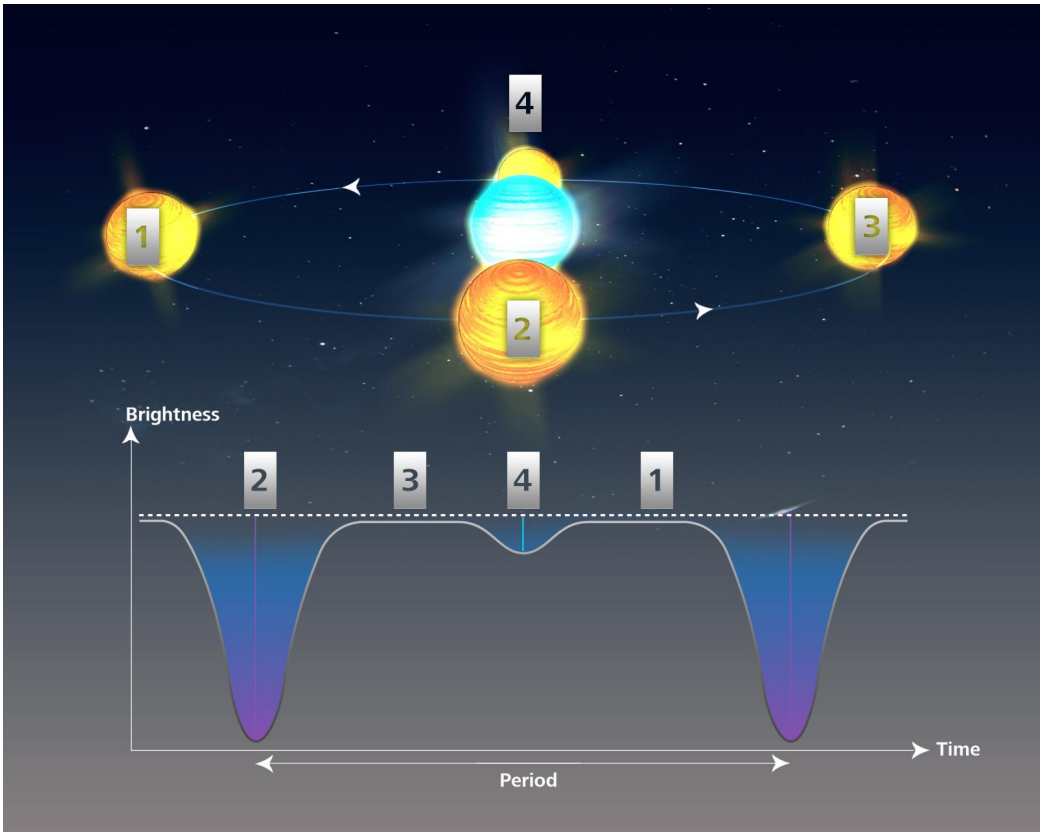


Figure 1.11: A representation of how eclipsing binary star light–curves are created. Credit: German Aerospace Center (*Deutsches Zentrum für Luft- und Raumfahrt e. V.*)

General information

Two stars orbiting around a common center of mass form a **binary star** — or **binary system**. An **eclipsing binary star** has the orbit plane of

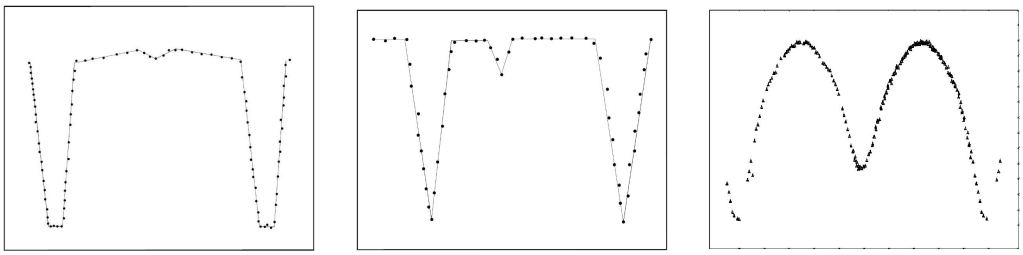
its components approximately in the observer's line of sight. Thus, mutual eclipses (as seen by the observer) occur, resulting to periodic drops of the magnitude of the star (§ 1.5.2) (Warner, 2006).

They are also called 'extrinsic variables' as the variability of the magnitude of the system is not caused by internal changes in the stars. Of course, one or both the stars may be 'intrinsic variable'.

Types of eclipsing binary stars

The main types are Algol, Beta Lyræand W UrsæMajoris (W UMa) and correspond to a different configuration. In Algol-type binaries the stars are well-separated, whereas Beta Lyrae and W Uma signify systems where the stars are close to each other. The light–curves of close binaries present a high distortion ("egg-like") due to the tidal effects between the members (Figure 1.12(c)). In the case of W UMa group, one or both stars expanded beyond their 'Roche limits' and the turbulent nature of these objects may change the profile of the light–curve dramatically in even few cycles.

Light–curve properties



(a) Reflection effect on Algol-type eclipsing binary

(b) Eccentricity effect on eclipsing binaries

(c) Close binary light–curve

Figure 1.12: Various effects on the light–curves of eclipsing binaries. Figures taken from Warner (2006).

The form of the light–curve depends on many factors, leading to different results:

- **Relative magnitude**: If the stars are of approximately equal magnitude, then the drop of magnitude for each eclipse will always be the same. In most cases, there is a difference in the intensity of the light produced by each star — leading to a sequence of big and small drops of intensity when the brightest or the faintest star is obscured, respectively. We call them **primary** and **secondary** minima.

- **Intrinsic variability**: One or both of the components may present another kind of variability which will be shown completely or partially (during an eclipse).

- **Eccentricity**: In a circular orbit, the eclipses will be equally spaced in time. On the other hand, eccentricity may result to a shift of the one minima in respect to the other. Of course, the effect also depends on the orientation of the orbit. For example, if the line of sight lies on the semi-major axis of the orbit, then no shift will be presented (Figure 1.12(b)).

- **Limb darkening**: the disk of a star is not equally luminous in all its parts. When looking at the center of the disk we look deeper and hotter portions of the solar atmosphere in comparison to looking at the limb. Thus, the optical depth varies depending on the distance from the center of the disk leading to a softer drop of intensity during eclipses and to non–flat minimums. There are many models of representation of the phenomenon and usually two limb darkening coefficients are taken under consideration.

- **Inclination**: for exactly 90°orbit inclination in respect to the observer's line of sight, the smaller member of the system will be totally eclipsed resulting to a flat component of the minima. When this is not the case, no star can be totally obscured, leading to no flat regions and smaller drops of intensity (sometimes the smaller minima is undetected).

- **Reflection effect**: in the intervals between minima, the magnitude slightly increases and decreases, peaking around the second minimum.

The cause is the extra heating of the one side of the secondary by the radiation of the primary star, making it a little brighter (Figure 1.12(a)).

- **Gravity darkening**: for contact binaries, the tidal effects deform the stars to ellipsoidal shapes that their ends are cooler, thus fainter. The change of orientation of those ends present as a variability.

1.5.3 Extra–solar planet transits

General information

Since late 80s and the first observational indications for the existence of planets around other stars[1], and the first unambiguous detection of an extra–solar planet revolving the star 51 Peg (Mayor & Queloz, 1995), 885 of confirmed and thousands of unconfirmed planets have been detected[2].

There are numerous successful and proposed techniques for detecting exoplanets. The **transit method** is the most successful after the **radial velocity method** and utilizes the shape of the host star's light–curve. The main idea is that we can detect a slight reduction of the magnitude of a star in case a planet of it's supposed planetary system passes in front of it as we observe it from Earth (Perryman, 2011).

Light–curve properties

The light–curve of the host of a transiting exoplanet can be considered a variation of that of an eclipsing binary. Although, the companion is not a star but a planet, the differences are quantitative and not qualitative (Figure 1.13). In comparison to Figure 1.11,

[1] Philosophers and scholars in antiquity and Renaissance suggested that other stars are distant Suns and may be coupled with Earths

[2] As of May 7, 2013. Source: The Extrasolar Planets Encyclopaedia (www.exoplanet.eu)

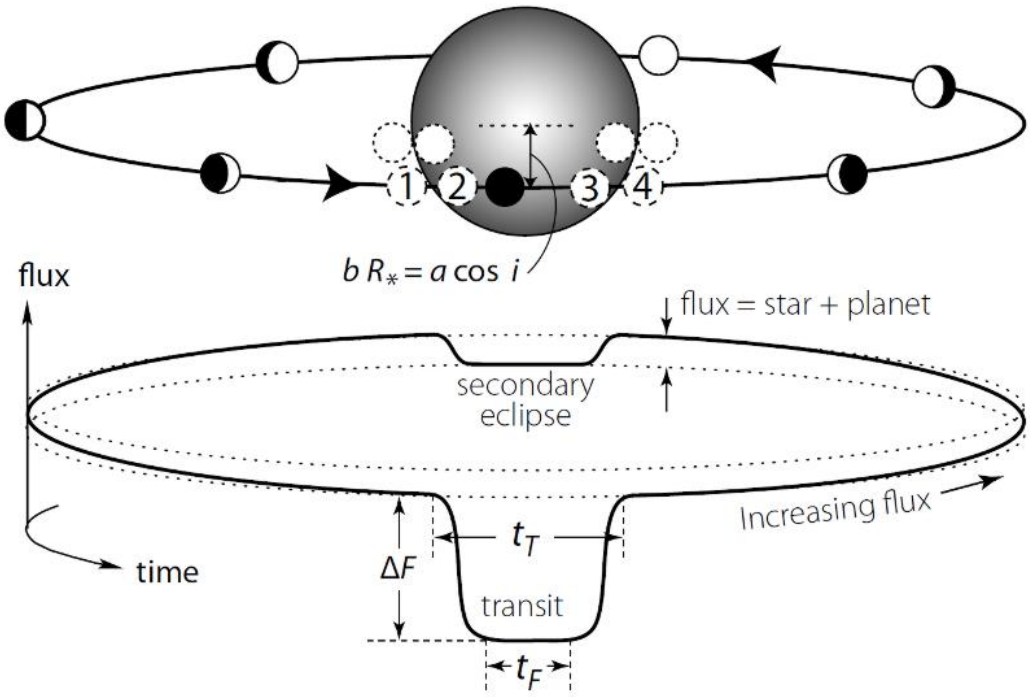


Figure 1.13: 3D representation of an exoplanet transit and the effect of the various phases to the light–curve of the host star (Perryman, 2011).

- The small size of planets creates a smaller drop of magnitude during the primary eclipse, and also tends to create more flat minima in contrast to eclipsing binaries
- Planetary radiation is millions or billions less powerful than stellar in optical wave–lengths making the secondary minimum undetectable. Thus, eccentricity can be detected only by careful observation of the shape of transit.
- The limb darkening effect is also present here (Figure 1.14), making the baselines of minima curved. The orbital inclination plays a significant role as a small object — a planet — may only pass through darkened regions of the stellar disk.
- The reflection effect is very small but it can be seen[1]

[1] performed in few cases of targets of the CoRoT and Kepler missions, with the latter even discovering two candidates through this effect

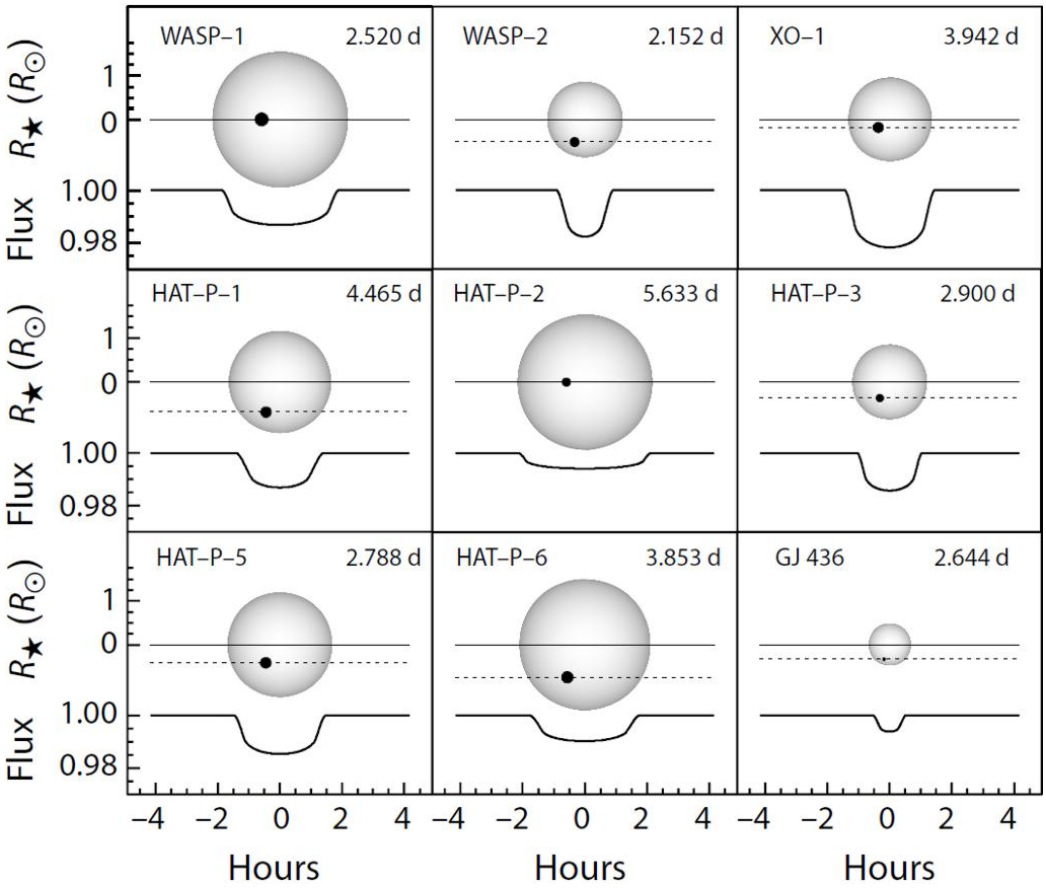


Figure 1.14: Examples of exoplanet transit profiles. Planet–star sizes are shown in scale. We can see the effect of the planet size, the orbit inclination and limb darkening to the form of the light–curve (Perryman, 2011).

1.6 Goals of the thesis

In order to check and compare the use of the FFT and FFA algorithms in astronomical data we have to perform a number of tasks and answer certain questions:

1. Implementation of the algorithms and check of their good function
2. Are they as time consuming as we expect them to be?
3. What are their success rate when analyzing time–series of various parameters: shape, period, noise level?

4. Can they be used to detect the period of three representative classes of periodic astronomical phenomena with small errors (Cepheid variables, eclipsing binaries, extra–solar planet transits)?

"—Watson, you are coming along wonderfully. You have really done very well indeed. It is true that you have missed everything of importance, but you have hit upon the method..."

Sherlock Holmes in Sir Arthur Conan Doyle's 'A Case of Identity'

2 Methodology

2.1 Programming

2.1.1 Programming conventions

Among the various programming languages that are commonly used in astrophysics (Fortran, C/C++, IDL, Perl, etc.), we selected ANSI C (or C89/C90) for reasons of compatibility, readability and familiarity. As opposed to the object–oriented C++, the algorithms developed here, can easily be translated to other structured–programming languages.

The current version of the software manipulates time series as arrays of floating–point variables (length of one computer word, which in modern computers usually means 32–bit information.) On the other hand, times are stored into double precision floating point variables (two computer words, 64–bit) as we decided to use BJD days as unit: many time–series are sampled

at milliseconds (e.g. pulsar radiation).

An advantage of of floating–point data types is that in most cases, we do not need to check if the value of a summation approaches the minimum or maximum value, risking a change of sign[1]. All input and output streams were of ASCII format.

2.1.2 Implementations of FFT, FFA and Folding

Here there is a description of the main functions the software uses. The input of FFT and FFA algorithms is an array of measurements (bins). The folding algorithms are also fed with the observation times of each measurement.

FFT

The Fast Fourier Transform is performed by two cooperating functions. The `FFT` function:

- Allocates memory for the buffer arrays and initializes them
- Calls the `RECURSIVE_FFT` which
  - calls itself two times — each for one half of the input, until size–2 halving occurs
  - combines the resulting tables in a 'FFT–butterfly' manner, and returns them

  This process ultimately results to the DFT of the input
- The output is saved in a report file,
- and searched for the maximum which is rendered as the most probable periodicity of the signal.

[1] For example a `char` variable in C programming language will jump from 127 to -128 if we add 1

- as the spectral leakage is not analyzed, an uncertainty is returned. It's calculation steps are:
  - i_{max} is the index of maximum in the DFT output
  - the minimum and maximum periods that would result to this maximum are
$$\frac{\text{input's duration}}{i_{max} \pm 0.5}$$
- their mean is rendered as the detected period
- the half of their absolute difference is stored as the quantization error of the process

FFA

The Fast Folding Algorithm was implemented by the function `FFA` which utilizes the `FFA_RESULT` structure to store the results. The latter include the detected `period`, the resolution or distance between two periods in the output data — `dp`, the `significance` measure and a flag, the `outcome`.

The algorithm,

- computes the number of trial periods that can fit in the whole extend of the input and reduces it to the maximum power of 2. That is the number of tables which will result to the output folds for different periods
- allocates the memory for computation buffers and initializes them
- performs the FFA using the buffers to work in–place
- calculates the standard deviation[1] for each of the resulting fold tables
- stores the the maximum standard deviation and the period of the table in which it was found — using the Equation (1.16) — as the `significance` and `period` elements of the structure the function returns

[1] Which — in digital signal processing language — is the square of the AC power

- the `significance` is divided by the square root of the number of folds to simulate the reduction of the signal–to–noise ratio (see Equation (1.8))
- the resolution of the period is returned as it is a measure of uncertainty due to the discrete nature of the procedure. As it can been seen in Equation (1.16), it is equal to $\frac{1}{\text{number of folds - } 1}$
- memory is freed and the report file is closed

Also, for blind searches the above function must be called for many trial periods. We restricted the range to 8 to $N/8$ elements. Overall the function `FFAsearch` performs this task, taking the following steps:

- memory allocation for buffers
- preparation for writing in a report file
- the `FFA` is called for all the trial periods and the results (pairs of `period` and `significance` values) are stored to the report file
- the maximum `significance` indicates the most probable period.
- a new significance measure for the blind–search is defined. All the `significance` values that were resulted by the `FFA` calls, are stored. The new significance measure is the distance of the detected period's `significance` from the mean, divided by the standard deviation of the distribution.
- the detected period and the quantization error that corresponds to it (period resolution) are expressed in physical units
- memory is freed and the report file is closed

The code of FFA is listed in Appendix C.4

Folding

The function `FOLD`, folds the input into a phase diagram for a desired period (call argument). The algorithm is as follows:

- preparation of a report file
- the first observation time is abstracted by the rest so that we result to a $(0, \text{duration})$ range
- the fractional part of the division of the times by the period is taken $\frac{t}{p} - \left\lfloor \frac{t}{p} \right\rfloor$ (range=0 to 1)
- abstracting a 0.5 from the previous value, we end up with the phase of that measurement in the folding period (range=-0.5 to 0.5)
- the phase and the normalized to 1 observation measure of each element, are saved in the report file

The result is a same–sized data table with no additions performed between the measurements. In this way, we can easily distinguish the case where the folding period corresponds to a resonance of two or more periods, a harmonic, or just approximates the real period of the signal.

On the other hand, if the fold is done by adding the elements of the time–series, in the case of a successful detection of the actual period, high noise we be reduced. The function `FOLD_REBINNED` produces the above fold, only this time the phase is stored in phase bins, the number of which is given as a call argument. Each phase bin relates to a certain range of phases and the output will contain the mean value of the measurements that fell into that bin. The steps taken are:

- two tables are allocated, one for the bins and one for the count of observations stored in each bin
- preparation of a report file

- the first observation time is abstracted by the rest so that we result to a $(0, \text{duration})$ range
- the fractional part of the division of the times by the period is taken $\frac{t}{p} - \left\lfloor \frac{t}{p} \right\rfloor$ (range=0 to 1)
- the previous value is multiplied by the number of bins and the integer part is taken (0 to BIN-1) giving the index of the bin to store the measurement
- the measurement is added to the appropriate bin and the count of observations for this bin increases by one
- after all elements are processed, all bins are divided by the number of observations they contain to give the mean, and are stored to the report file
- for bins that the count was 0 the output was suppressed as they contain only the initialization value 0 which has nothing to do with the actual data

2.2 Complexity test

By performing this test, we tried to confirm the computational complexity of FFT and FFA. Here, the conducted experiment steps are enumerated:

1. The report files `reportffa.txt` and `reportfft.txt` were created.
2. A function creating random light–curves was included. The parameters are of no importance as the algorithms' speed does not depend on the contents of the input.
3. The size of inputs was 32, 64, 128, ..., 16777216. In the case of FFA the $O(n^2)$ complexity led us to use a limit of 262144.

4. Each of the `FFT` and `FFAsearch` were called and timed using CPU clocks.

5. Small inputs were processed in a matter of micro– or milliseconds, a duration that cannot be measured by the built-in functions of C (some milliseconds accuracy). To overcome this problem, every step (different size of input) was repeated until at least 10 seconds had passed. By division with the number of repetitions we resulted to a good estimate of the computation duration.

6. For better statistics, each measure was repeated at least 5 times — independently of the previous step

7. The times were translated to nanoseconds and were saved to the report files. Then, divided by $N \log_2 N$ (FFT) and N^2 (FFA), we resulted to the coefficient of the first term of the computational complexity for each input size, which was also stored to the report file. Fairly constant coefficients would confirm the expected behavior of the algorithms.

2.3 Test on constructed lightcurves

The performance of FFA and FFT under the presence of high noise and various signals, was tested by manufacturing our own light–curves with known parameters.

As the absolute value of signal or noise power is not important, we selected to hold still one of them — the signal power — and vary the noise in respect of that. Also, the size of the time-series or the periods is irrelevant as for sufficiently large values, they present a continuous behavior. Utilizing this fact, we held the size of the time-series low to avoid unnecessary time consumption and varied the period.

The signal's form is a very important factor in order to be detected. As there are many forms met in the field, we chose to include two basic ones that approximate a wide range of astronomical phenomena: (i) sinusoidal

and (ii) rectangular, or 'box-like'. The first approximates the light–curve of many types of variable stars. The second can highlight the behavior of the algorithms in the cases of pulsars, extra-solar planet transits etc — where narrow 'spikes' or 'bumps' are of interest.

Though, a rectangular waveform requires one more parameter than the sinusoidal: the duty cycle (D). The latter — in the context of periodical signals — is the duration of the 'event' over the total period of the signal. For example, in the case of extra–solar planets detected by the transit method, the duration of the planet's passing in front of the star might last some hours, whereas the period might last days, months or even years[1].

The phase of the periodic signals isn't considered to alter the results for both the algorithms. Therefore, it was selected randomly for each constructed light–curve.

Description of the process

1. A light–curve manufacturing algorithm was created. The parameters were
   - DC offset
   - Noise level
   - Sinusoidal component's (i) amplitude, (ii) period and (iii) phase
   - Rectangular component's (i) amplitude, (ii) period, (iii) phase and (iv) duty cycle
2. The DC offset was set 0 as the period searching algorithms don't depend on it — they compute the standard deviation instead of the signal power
3. The noise level lied in the range 0.0 to 3.0 in respect to the signal's amplitude that was constant (1.0) and took 20 equidistant values

[1] The current confirmed extra–solar planets have small orbital periods due to detection bias: we're searching for only two decades with surveys lasting a few years in the best case, and the confirmation usually requires a second event

4. The period took 50 different values in the range of 32 to 128 elements in time-series of 1024 elements.

5. The phase was randomly picked for each light–curve

6. The test run for 4 shapes:

   - Sinusoidal
   - Rectangular of duty cycle 20%
   - Rectangular of duty cycle 10%
   - Rectangular of duty cycle 5%

7. Having a known period, we recorded the error of the estimates that resulted from the FFT and FFA algorithms. We selected a tolerance of 0.1 to render a detection failed and the error was reset to 1 (100% failure). As it will be seen, this will better illustrate the performance of the algorithm.

8. A series of three-dimensional plots will be created to show the dependence of detection from the period or the level of noise for all the 5000 thousand light–curves that were created.

2.4 Test on Kepler mission data

As the FFA and FFT algorithm implementations in hand, have as input evenly–spaced time series, one should be careful when using actual astronomical data. For the reasons explained in §1.1 we selected data from the space telescope Kepler.

Another reason for using this mission's data is the availability of the raw data, but also of many publications with results, with which we can compare our own.

2.4.1 Selection of targets

The All Sky Automated Survey (ASAS) has published a list of 947 variable stars in the Kepler field (Pigulski et al., 2009). The type of the variable and the detected period are also provided. Using this as a guide, we selected 4 objects of each category:

- Cepheid variables
- Eclipsing binaries
- Hosts of transiting extra–solar planets

2.4.2 Light–curve data manipulation

The Kepler ID of each ASAS target was identified by their coordinates in the equatorial system (right ascension and declination). Then, the light–curves of the stars were taken from the *NASA Exoplanet Archive*[1]. The data files contain the exact time of every measurement in BJD.

2.4.3 Regularization of near–evenly–spaced time series

Although the sampling frequency of the Kepler space telescope is fairly constant (approximately to 30 minutes for long cadence and 1 minute for short cadence), we performed a test on this fact.

In most of the Kepler Mission's observation quarters, there are gaps lasting a few days, which correspond to the Safe Modes in which the space telescope enters for technical reasons or as a precautionary measure. For example, in quarter 8 there were three Safe Modes. The first one occurred between the transition from Q7 to Q8 causing a delay in the resumption of

[1] http://exoplanetarchive.ipac.caltech.edu/applications/ETSS/Kepler_index.html

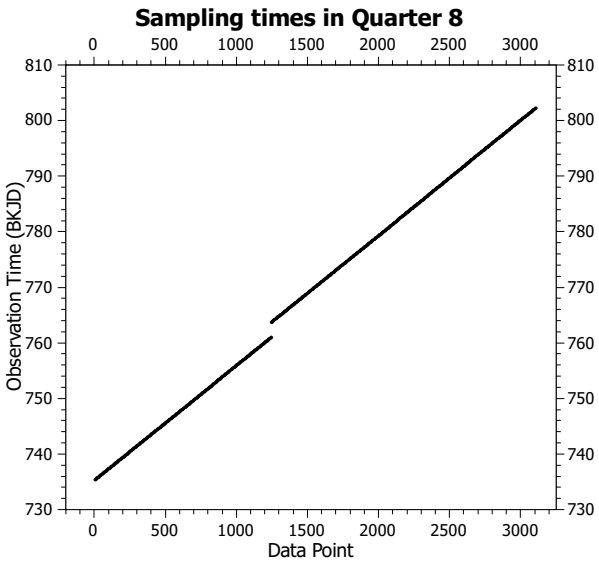


Figure 2.1: Sampling times in observational quarter 8 of Kepler Mission. Note the three–day gap in the second month.

the observations. The second one occurred in the second month of the quarter. The third one, having happened in the end of the quarter, halted the observations early (Figure 2.1) (Christiansen et al., 2011).

Now, we excluded the data before the gap in order to focus on the larger part of the quarter with continuous observations. By subtracting the observation times in pairs (next minus previous) we derive an estimation of the sampling period[1] for each data point. For evenly–spaced time–series, the plot of sampling period would be a straight line. Instead, there is a small number of cases were the sampling period is two or four times larger (Figure 2.2). These deviations correspond to measurements affected by Momentum Desaturations: approximately every 3 days the telescope's thrusters correct unwanted angular momentum caused by solar radiation torque (Christiansen et al., 2011).

Excluding the 'bad' data points, the sampling period has (i) mean value of 0.0204341 BJD (29 minutes and 25.5 seconds) and (ii) standard deviation $1.489 \cdot 10^{-7}$ BJD (13 milliseconds). That half–hour sampling rate corresponds

[1] the time interval between two measurements

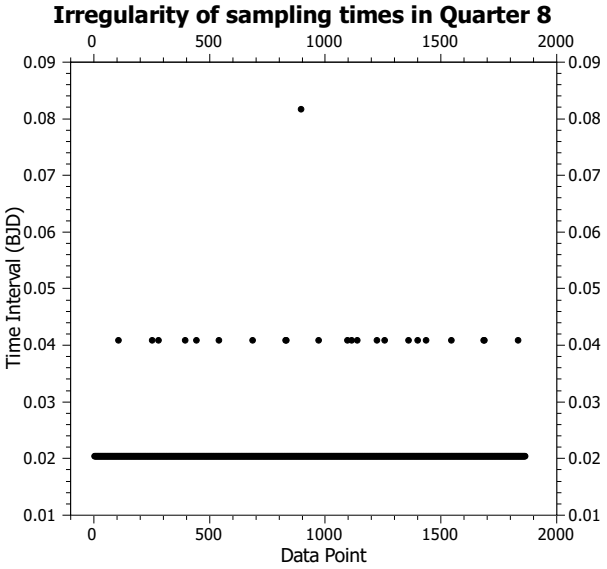


Figure 2.2: Irregularities in sampling period in Kepler's Quarter 8. Note that we do not refer to Telescope Safe Modes which result in larger gaps in time.

to the 'long cadence' configuration of the Kepler mission. Though, the intervals between momentum desaturation events last a few days and by cropping the data (as we did in the case of Safe Modes) is not desirable.

The mean and standard deviation of the intervals without excluding the deviated points is 0.0207194 BJD (29 minutes and 50.2 seconds) and 0.00266 BJD (3 minutes and 50.2 seconds) respectively. As we can see, those few data points can result to small uncertainties to period estimation in respect to sampling period. Besides that, the code should also be able to handle measurements of other surveys with less uniform timing and even 'random–like' declinations in observation times.

Proposed solution

One way to overcome these difficulties[1] is to fit the times in a linear model using the least squares method (LSM) (although it assumes Gaussian

[1] there are methods discovered after FFT and FFA for analyzing unevenly spaced data but they exceed the current study. For more information, see Lomb–Scargle Periodogram: Lomb (1976); Scargle (1982).

distribution for the residuals, Wall & Jenkins (2003)):

- We enlist the N observation times t_i, where $i \in \{0, 1, ..., N-1\}$.
- We consider a linear model that depends on τ_0 indicating the beginning offset and a constant sampling period p_s which corresponds to N evenly–spaced observations times τ_i where $\tau_i = \tau_0 + i \cdot p_s$.
- As a 'goodness–of–fit' measure we choose the summation of squares of residuals $g = \sum_{i=0}^{N-1} (t_i - \tau_i)^2$ which obviously is minimum for the best fit.
- We derive the values of τ_0 and p_s for optimum fit by solving the system of equations $\begin{cases} \frac{\partial g}{\partial \tau_0} = 0 \\ \frac{\partial g}{\partial p_s} = 0 \end{cases}$.
- It is proven (see § A.3) that:

$$\tau_0 = \frac{2}{N^2 + N} \left[(2N-1) \sum_{i=0}^{N-1} t_i - 3 \sum_{i=0}^{N-1} (i \cdot t_i) \right] \tag{2.1}$$

$$p_s = \frac{6}{N(N^2-1)} \left[2 \sum_{i=0}^{N-1} (i \cdot t_i) - (N-1) \sum_{i=0}^{N-1} t_i \right] \tag{2.2}$$

- From the resulting minimum goodness–of–fit value, g_{min} we calculate the standard error of regression in time that we will call 'regularization time error' and denote as t_{err}[1]. By the definition of g and the standard deviation, and our selection to use of 2.58 standard deviations (99% of values in the range, assuming normal distribution of the residuals), we define:

$$t_{err} = 2.58 \sqrt{\frac{g_{min}}{N-2}} \tag{2.3}$$

  Note that the two–parameter model dictates division by $N - 2$.

After the above process, any proceeding analysis will make use of the new sampling period and take account of the regularization time error. For the code listing see Appendix C.2.

[1] it is a form of scalar quantization error

2.4.4 Data analysis

The analysis of each light–curve of the selected targets was conducted by the following steps:

- Reading the file and recording the observation times and measures (photon counts in this case)
- Regularization of the time–series
- Application of FFT and FFA algorithms
- Production of the fold and re–binned fold of the data, for the two detected periods

"Results? Why, man, I have gotten lots of results! If I find 10,000 ways something won't work, I haven't failed. I am not discouraged, because every wrong attempt discarded is often a step forward..."

Thomas Alva Edison (American Inventor)

3 Results

3.1 Complexity test

In this section there are the results of the complexity tests, where the FFA and FFT algorithms were times in relation to the input size. In the following pages we can see the Duration - Input size diagrams but also the coefficients of the highest order terms of the expected complexity ($nlogn$ for FFT and n^2 for FFA blind–search).

Table 3.1: Complexity test results

	FFT		**FFA**	
Input size	Timing (s)	Coefficient of $N \log_2 N$ (ns)	Timing (s)	Coefficient of N^2 (ns)
32	0.000013	79.431076	0.000001	0.046539
64	0.000027	70.500275	0.000007	1.802245
128	0.000059	65.380890	0.000100	6.087590
256	0.000132	64.414253	0.000513	7.825421
512	0.000294	63.883896	0.002188	8.346994
1024	0.000699	68.224289	0.009205	8.778537
2048	0.001462	64.917915	0.038372	9.148514
4096	0.003310	67.334999	0.159222	9.490384
8192	0.006983	65.572792	0.596471	8.888104
16384	0.014793	64.493263	2.418000	9.007751
32768	0.034235	69.652298	10.043400	9.353645
65536	0.119595	114.054901	41.555199	9.675323
131072	0.271108	121.670052	160.449402	9.339384
262144	0.565944	119.939262	706.730835	10.284287
524288	1.229000	123.375336		
1048576	2.427400	115.747459		
2097152	5.070000	115.122116		
4194304	10.667400	115.604836		
8388608	23.297001	120.748650		
16777216	59.145802	146.890182		

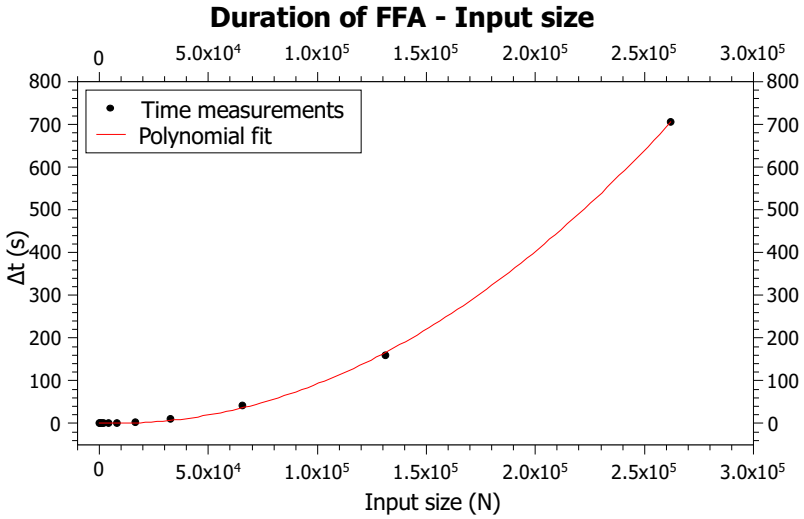


Figure 3.1: Dependence of FFA algorithm's time consumption to the input size

The fit's equation is $0.96 - 0.17 \cdot 10^{-4} N + 1.09 \cdot 10^{-8} N^2$ with $R^2 = 0.99985$.

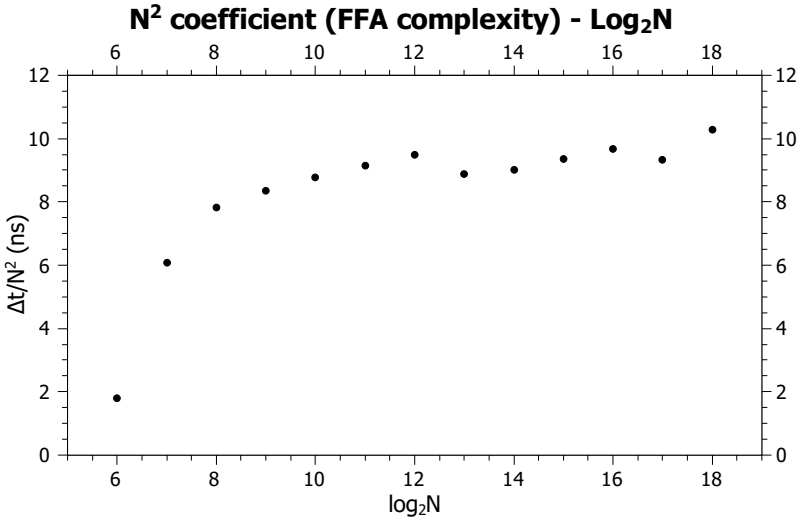


Figure 3.2: Result of complexity test for the FFA

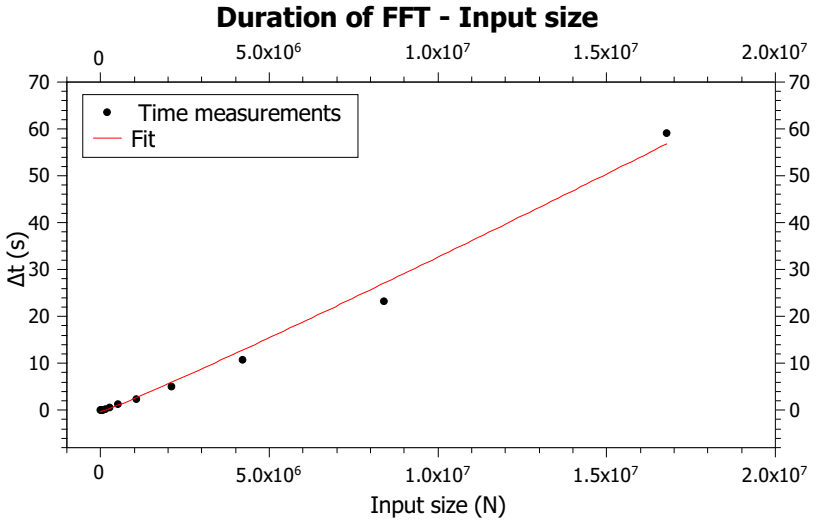


Figure 3.3: Dependence of FFT algorithm's time consumption to the input size

The fit's equation is $-0.33 + 1.42 \cdot 10^{-7} N \log_2 N$ with $R^2 = 0.99287$.

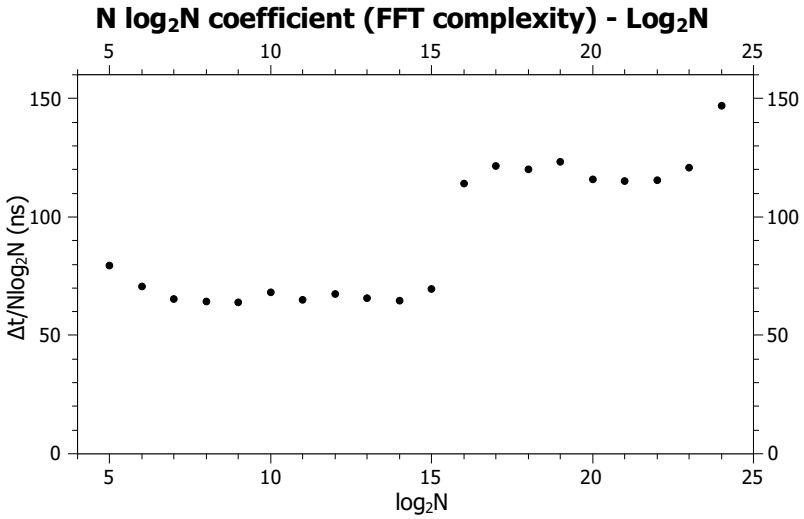
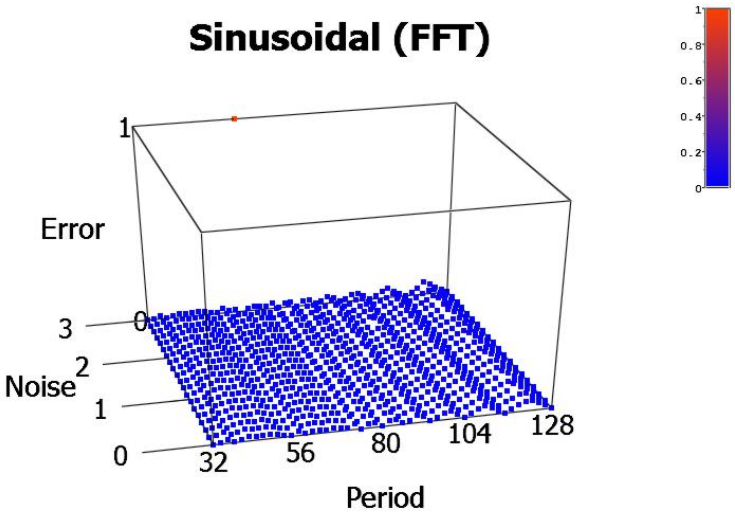
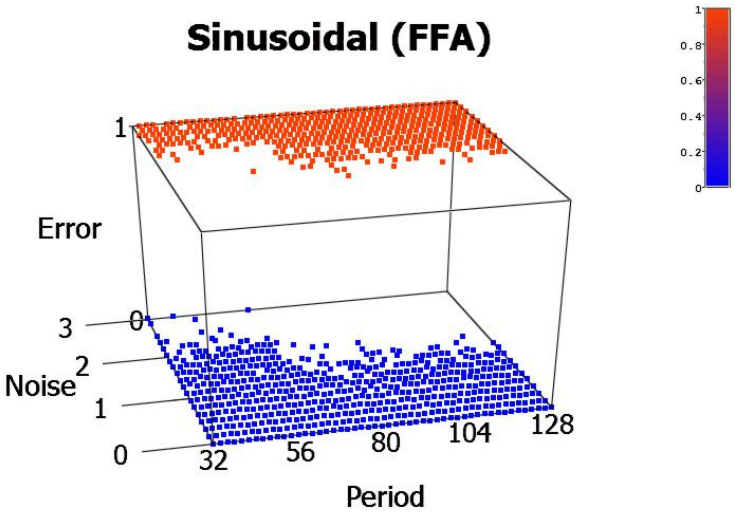


Figure 3.4: Result of complexity test for the FFT

3.2 Test on constructed light–curves

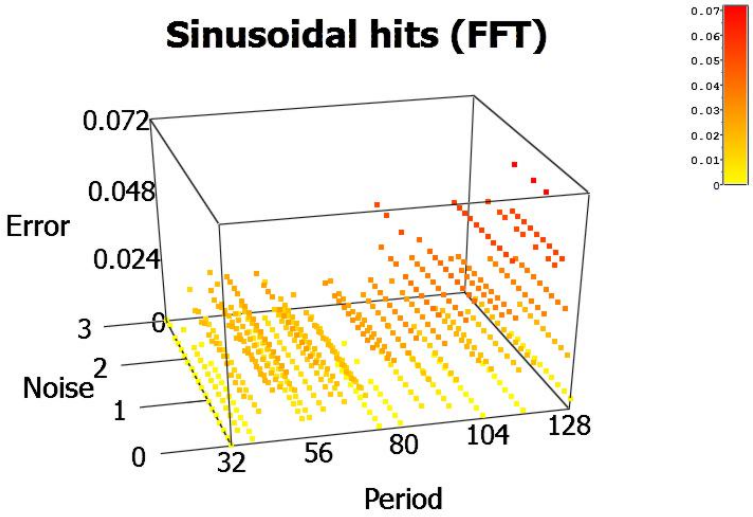


(a) Hits and misses in period estimation by FFT

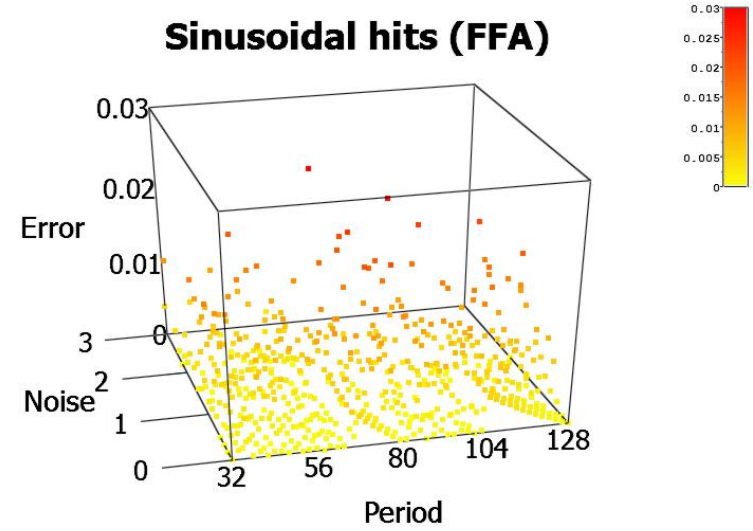


(b) Hits and misses in period estimation by FFA

Figure 3.5: Sinusoidal signal detection for various periods and noise levels

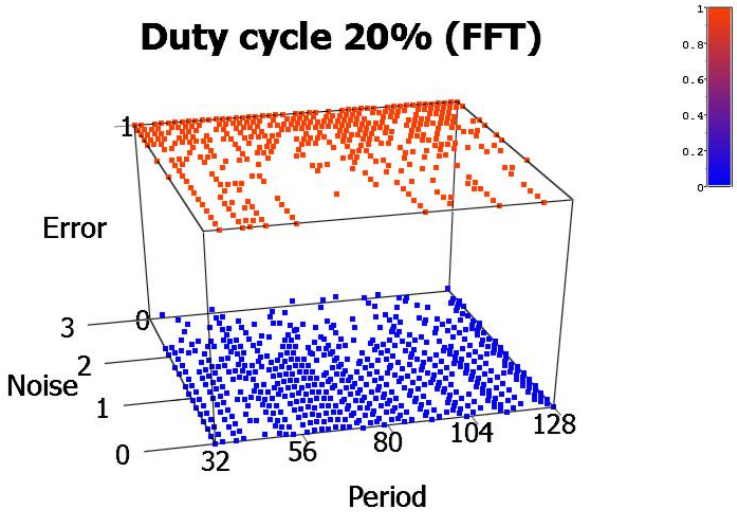


(a) Hits in period estimation by FFT in the case of sinusoidal signal

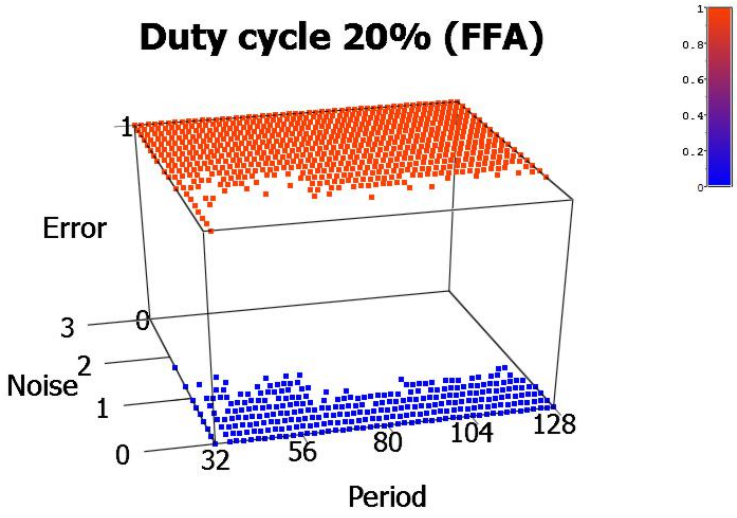


(b) Hits in period estimation by FFA in the case of sinusoidal signal

Figure 3.6: Sinusoidal signal detection errors: focus on hits by FFT and FFA

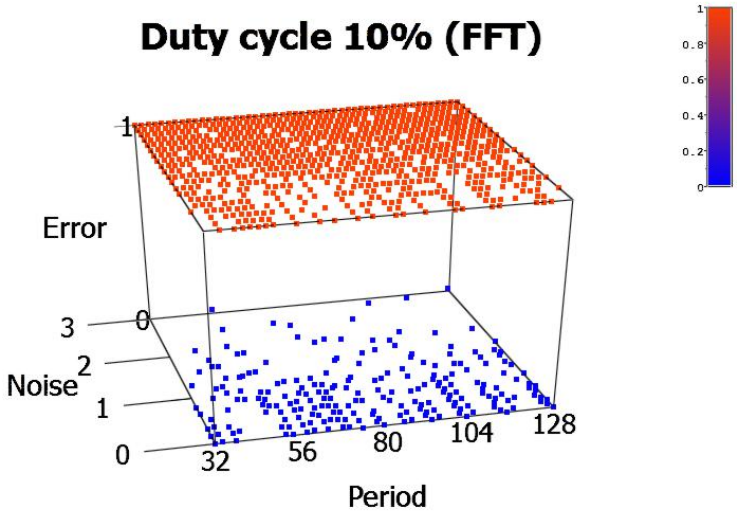


(a) Hits and misses in period estimation by FFT

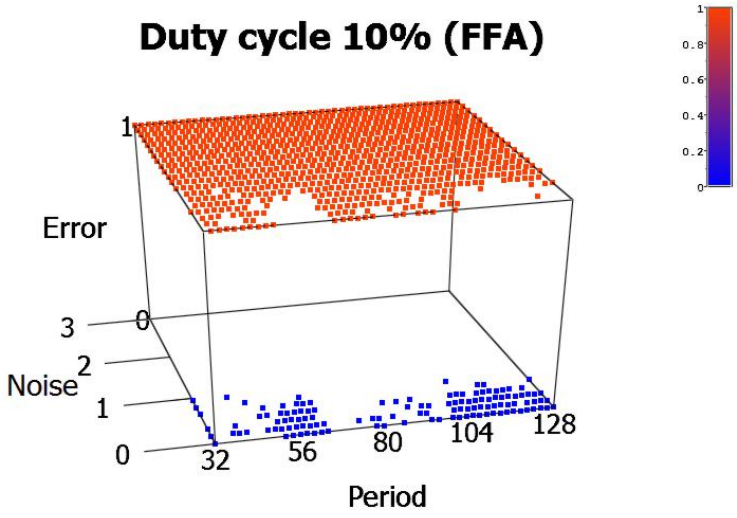


(b) Hits and misses in period estimation by FFA

Figure 3.7: Rectangular signal of duty cycle 20% detection for various periods and noise levels

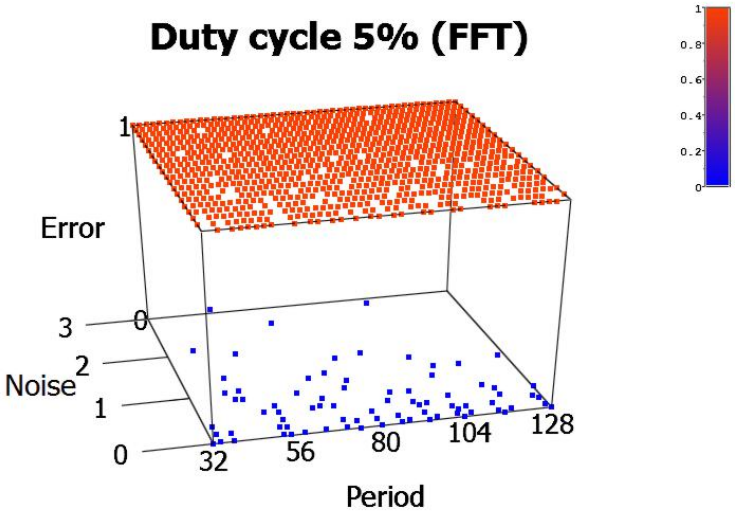


(a) Hits and misses in period estimation by FFT

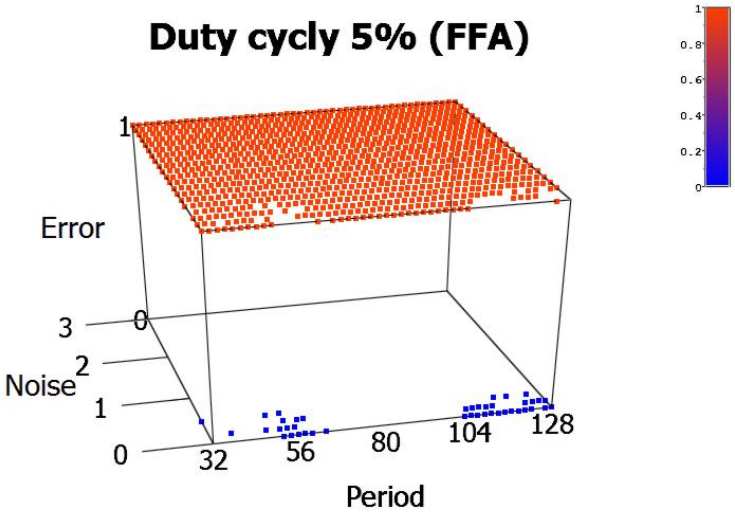


(b) Hits and misses in period estimation by FFA

Figure 3.8: Rectangular signal of duty cycle 10% detection for various periods and noise levels



(a) Hits and misses in period estimation by FFT



(b) Hits and misses in period estimation by FFA

Figure 3.9: Rectangular signal of duty cycle 5% detection for various periods and noise levels

3.3 Test on Kepler mission data

3.3.1 Output example

```
INPUT REPORT
--------------------
Kepler ID               7548061
Time offset             2454833.000000 BJD
ASAS period             4.926000 days
Notes                   Q14s
Bins                    49957

REGULARIZE REPORT
--------------------
New time offset         2456170.169175 BJD
Sampling rate           0.984465 min
Regularization error    9.043122 min

FFT REPORT
--------------------
> Spectrum file created.

Frequency:              5 bins
Period:                 4.525666 days   +/- 0.452567

> Folding at p=4.525666 saved to: foldfft.txt

> Re-binned (256) folding at p=4.525666 saved to: foldfftr.txt

FFA REPORT
--------------------
> Periodogram file created.

Period:                 4.931896 days   +/- 0.000114

> Folding at p=4.931896 saved to: foldffa.txt

> Re-binned (256) folding at p=4.931896 saved to: foldffar.txt

Press any key to continue . . .
```

Listing 3.1: The output of the program after analyzing the light–curve of a short cadence target of Kepler telescope

3.3.2 Results

Here is presented a list of all analyzed Kepler targets accompanied by the periods detected by the FFA and FFT algorithms. The deviation from the published result (Pigulski et al., 2009; O'Donovan et al., 2006; Welsh et al., 2010; Shporer et al., 2011; Boruckiet al. , 2011) (under the column *Period* in the *Target Description* section) is also given, as percentage error. Note that the detected periods are sometimes multiplied by a factor of 2 or 3 in case of false harmonic detection (see Discussion).

Table 3.2: Summary of Kepler data results

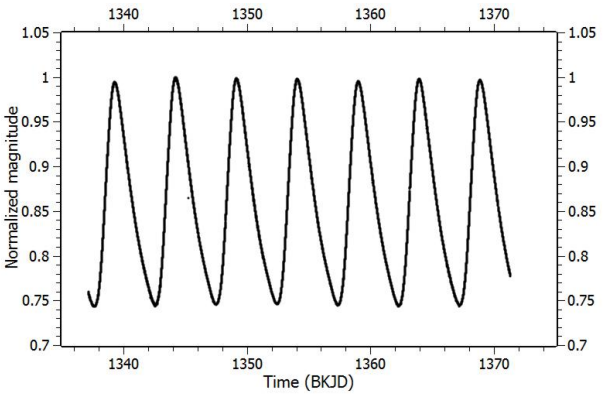
Target Description				**FFT**		**FFA**	
Kepler ID	RA (°) (J2000.0)	Dec (°)	Period (d)	Period (d)	Error (%)	Period (d)	Error (%)
Cepheid Variables							
7548061	297.064	43.127	4.9260	4.5267	8.11	4.9319	0.12
7899428	294.650	43.692	11.573	11.029	4.70	11.471	0.88
11347875	290.467	49.199	3.4521	3.3454	3.09	3.4322	0.58
12406908	290.937	51.270	13.367	14.883	11.3	3.4322	0.37
Eclipsing Binaries							
4660997	293.514	39.711	0.56256	0.56669	0.73	0.56177	0.14
4544587	285.886	39.683	2.1893	2.1745	0.68	2.1913	0.09
8868650	284.398	45.129	4.4472	4.3453	2.29	4.4460	0.03
10618251	298.125	47.812	0.43742	0.43815	0.17	0.43693	0.11
Hosts of Transiting Planets							
11446443	286.808	49.316	2.4706	2.5104	1.61	0.0034	99.9
10666592	292.247	47.970	2.2047	2.2444	1.80	0.0035	99.8
9941662	286.971	46.868	1.7637	1.7689	0.29	0.0027	99.8
11414511	297.015	49.225	2.8165	2.8110	0.20	0.0030	99.9

In the following pages we can see one table and seven figures for each target. The former accumulates properties of the star and some of the output of our software. The figures are plots of the original time series, the spectrum and periodogram created by FFT and FFA respectively and for foldings of

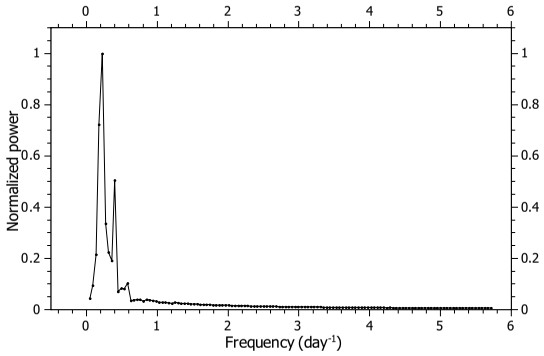
the data: a normal and a re–binned fold for the period that each algorithm detected. This extensive plotting aims to a better illustration of the process and to an inspection of the quality of the results.

Kepler ID	7548061
Time offset	2454833 BJD
Quarter	14 (short)
Type (ASAS)	Cepheid variable
Period (ASAS)	4.9260
Bins	49957
FFT period	4.526 660(452567)
FFA period	4.931 896(114)

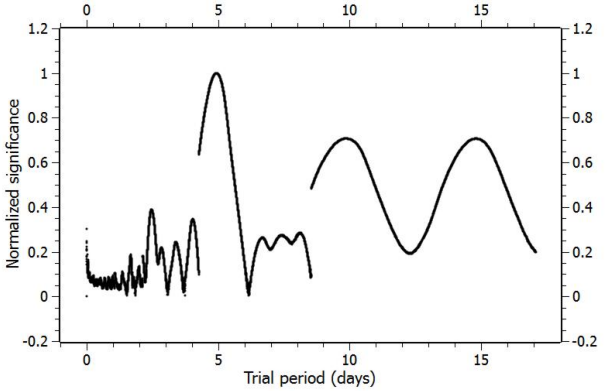
(a) Description & detected periods



(b) Lightcurve

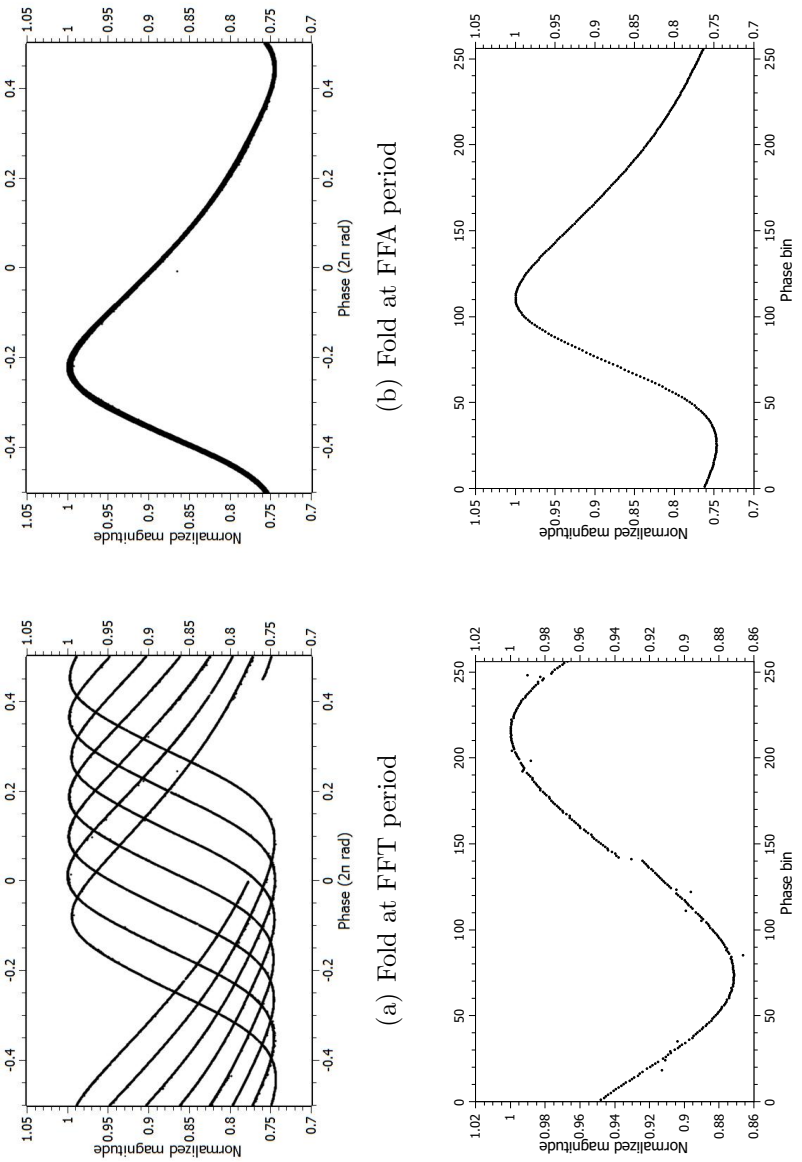


(c) FFT spectrum



(d) FFA periodogram

Figure 3.10: Results (a) for KID 7548061 — HIP 97439



(a) Fold at FFT period

(b) Fold at FFA period

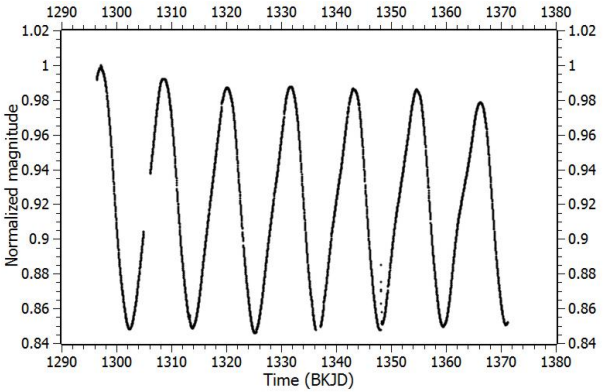
(c) Rebinned fold at FFT period

(d) Rebinned fold at FFA period

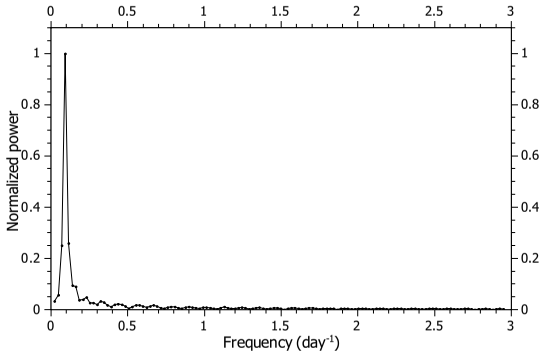
Figure 3.11: Results (b) for KID 7548061 — HIP 97439

Kepler ID	7899428
Time offset	2454833 BJD
Quarter	14 (long)
Type (ASAS)	Cepheid variable
Period (ASAS)	11.573
Bins	3535
FFT period	11.028 791(1378599)
FFA period	11.471 383(3534)

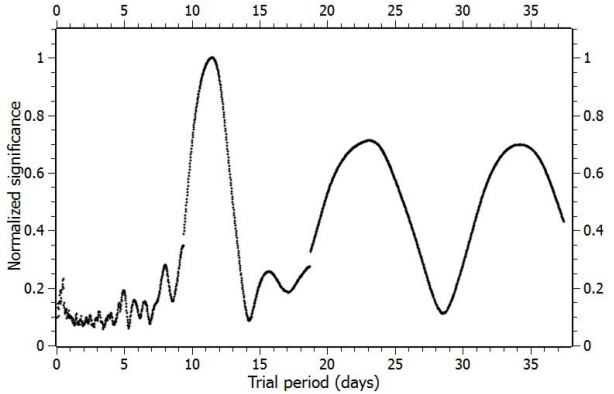
(a) Description & detected periods



(b) Lightcurve

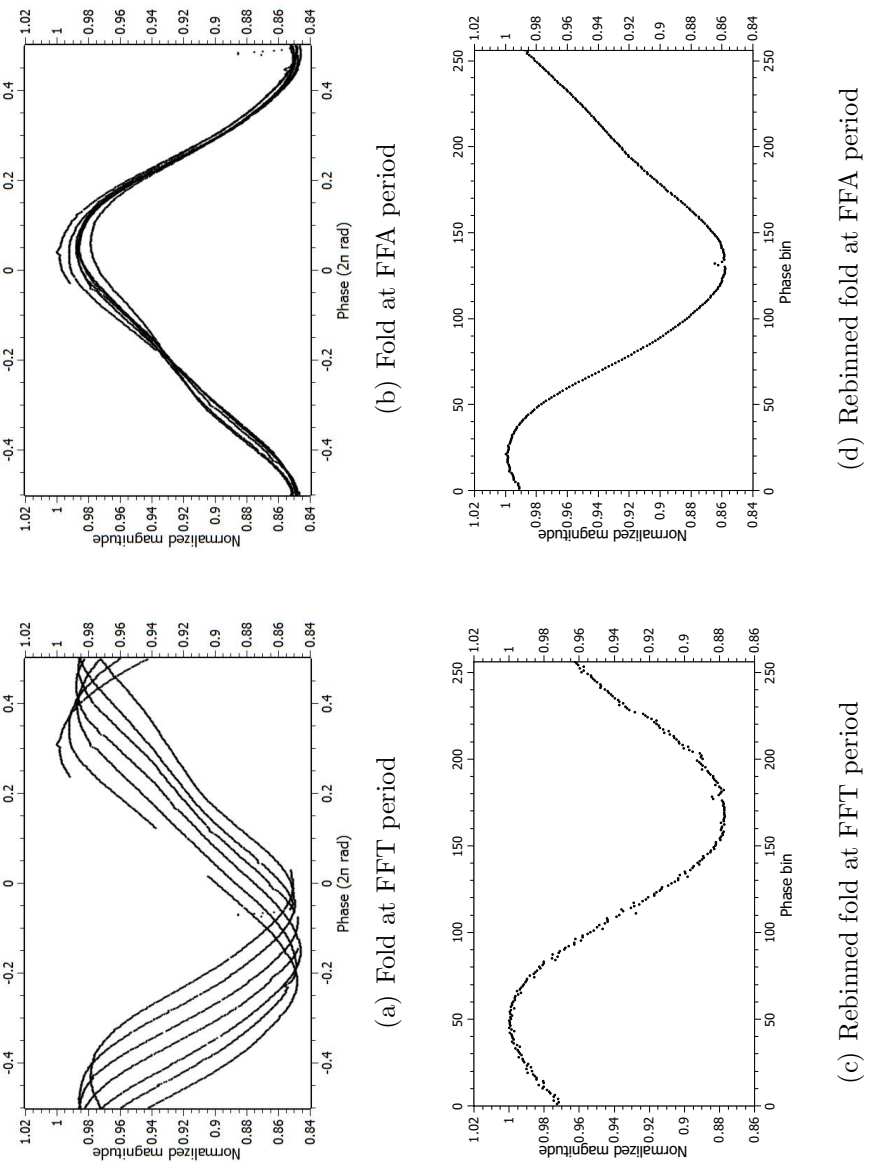


(c) FFT spectrum



(d) FFA periodogram

Figure 3.12: Results (a) for KID 7899428 — TYC 3147–400–1



(a) Fold at FFT period

(b) Fold at FFA period

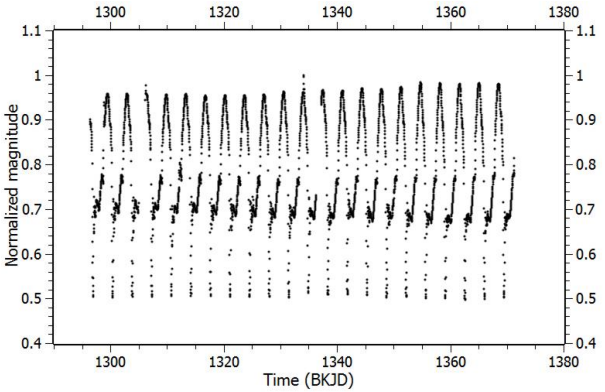
(c) Rebinned fold at FFT period

(d) Rebinned fold at FFA period

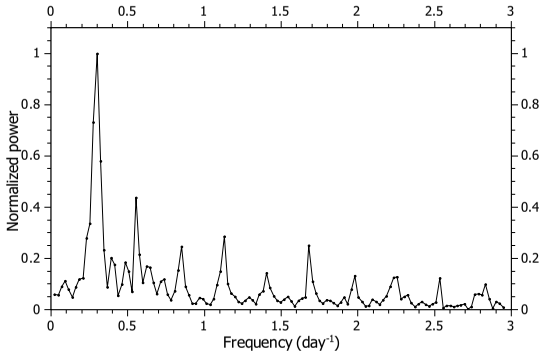
Figure 3.13: Results (b) for KID 7899428 — TYC 3147–400–1

Kepler ID	11347875
Time offset	2454833 BJD
Quarter	14 (long)
Type (ASAS)	Cepheid variable
Period (ASAS)	3.4521
Bins	3535
FFT period	3.345 397(128669)
FFA period	3.432 224(707)

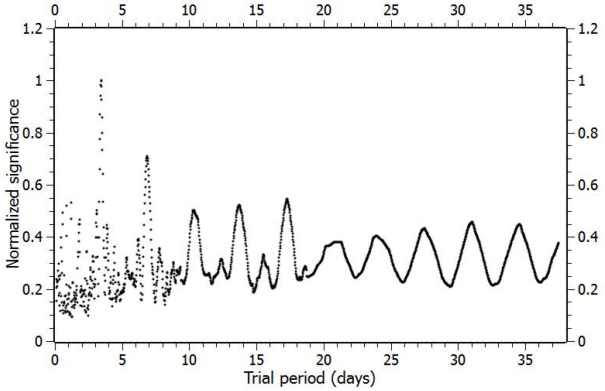
(a) Description & detected periods



(b) Lightcurve

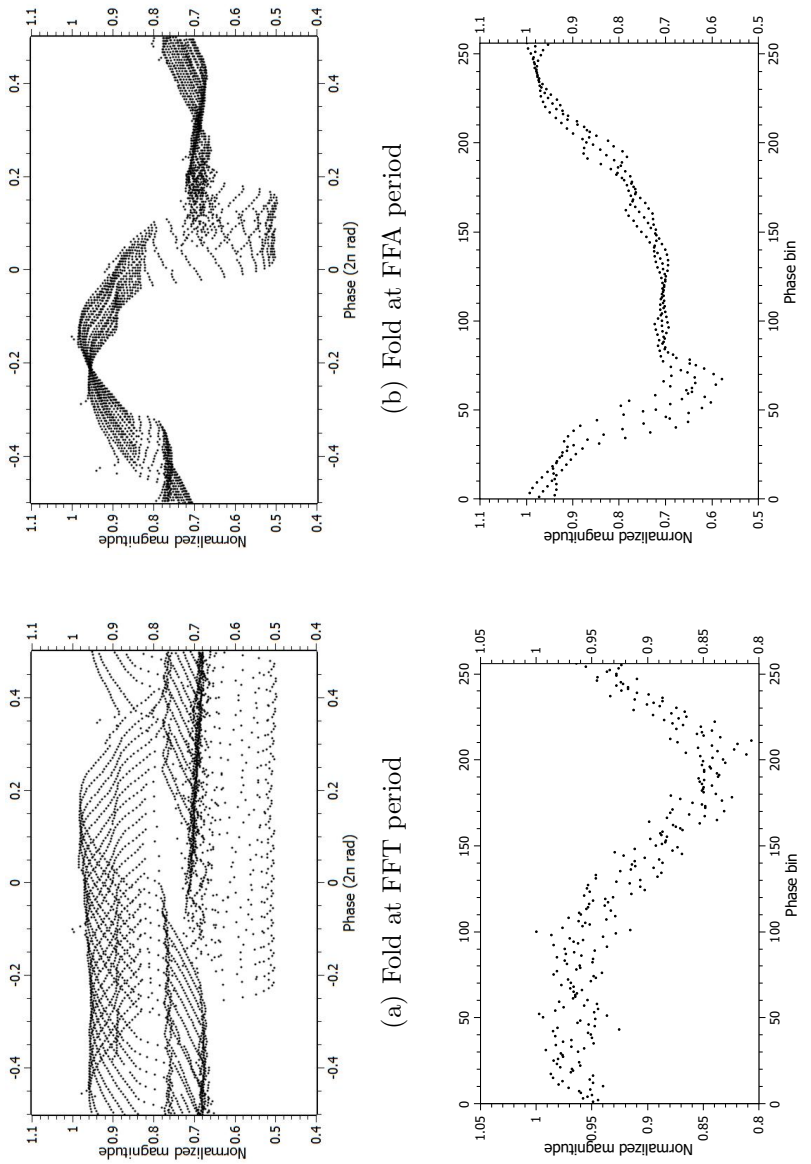


(c) FFT spectrum



(d) FFA periodogram

Figure 3.14: Results (a) for KID 11347875 —2MASS J19215214+4911551



(a) Fold at FFT period

(b) Fold at FFA period

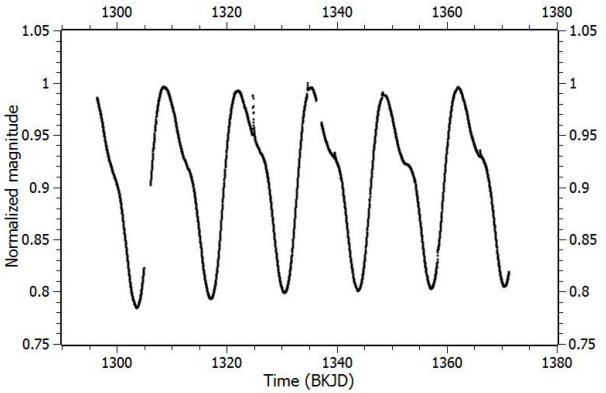
(c) Rebinned fold at FFT period

(d) Rebinned fold at FFA period

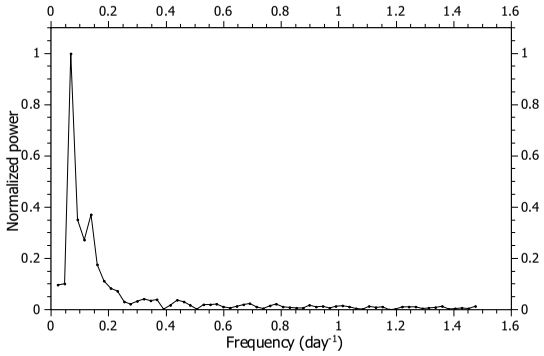
Figure 3.15: Results (b) for KID 11347875 — 2MASS J19215214+4911551

Kepler ID	12406908
Time offset	2454833 BJD
Quarter	14 (long)
Type (ASAS)	Cepheid variable
Period (ASAS)	13.367
Bins	3536
FFT period	14.883 124(2480521)
FFA period	13.318 062(3533)

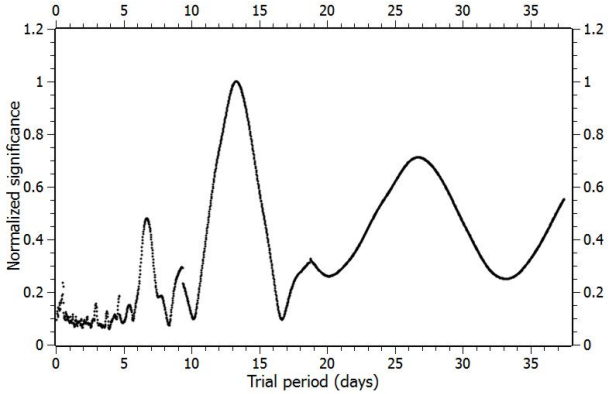
(a) Description & detected periods



(b) Lightcurve

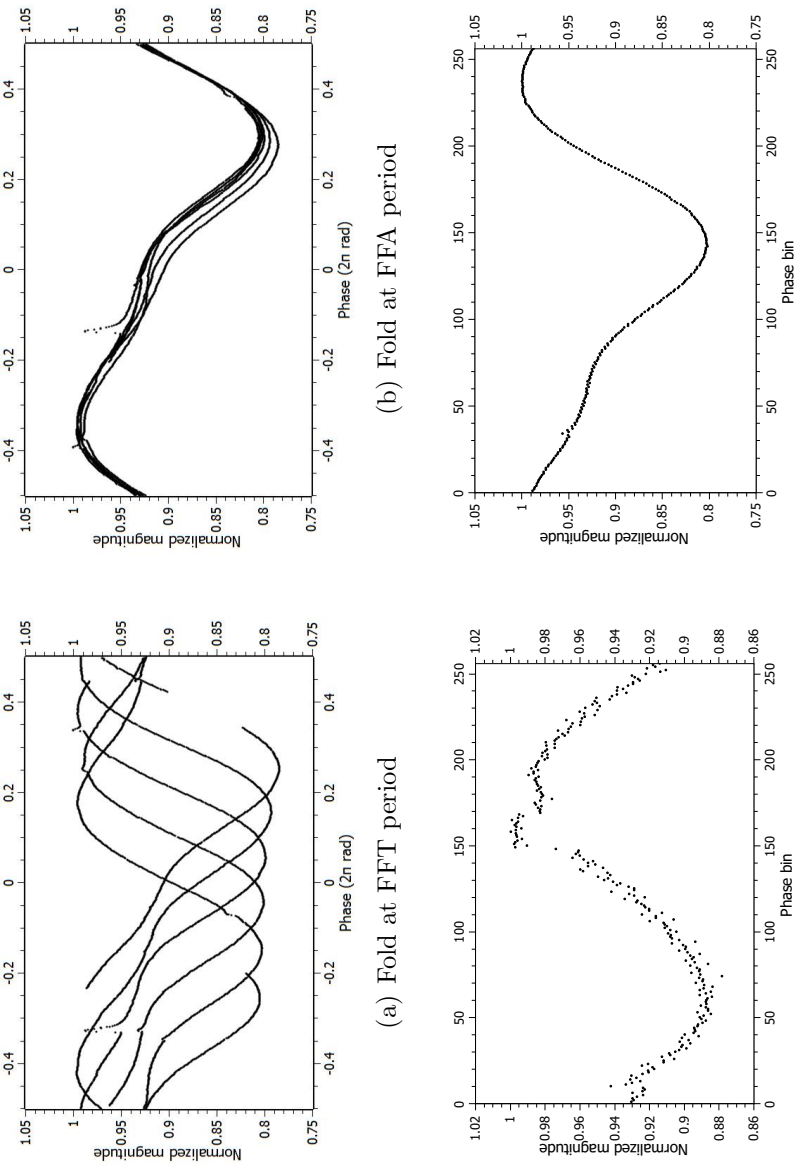


(c) FFT spectrum



(d) FFA periodogram

Figure 3.16: Results (a) for KID 12406908 — GSC 03555–00876



(a) Fold at FFT period

(b) Fold at FFA period

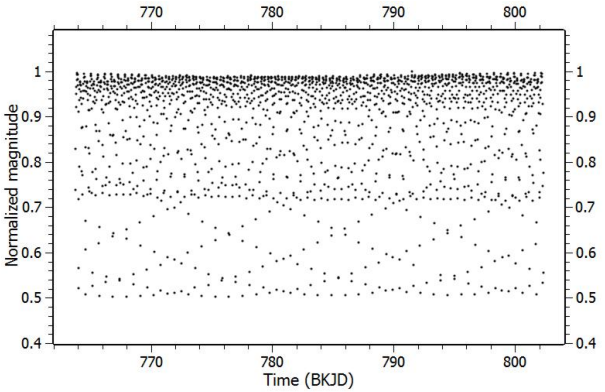
(c) Rebinned fold at FFT period

(d) Rebinned fold at FFA period

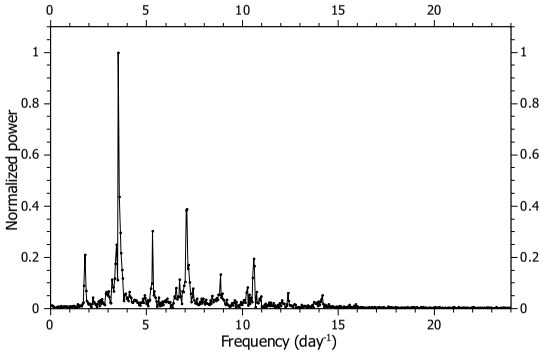
Figure 3.17: Results (b) for KID 12406908 — GSC 03555–00876

Kepler ID	4660997
Time offset	2454833 BJD
Quarter	8 (long)
Type (ASAS)	Eclipsing binary
Period (ASAS)	0.562560
Bins	1862
FFT period	0.283 346(1889)
FFA period	0.280 887(82)

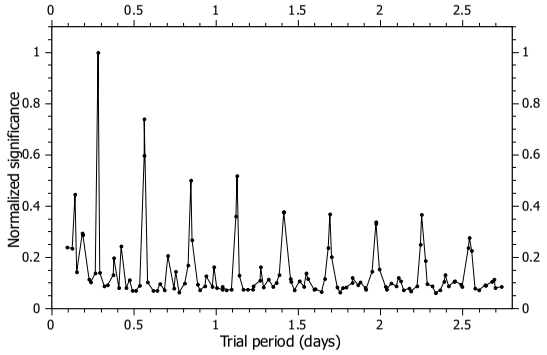
(a) Description & detected periods



(b) Lightcurve

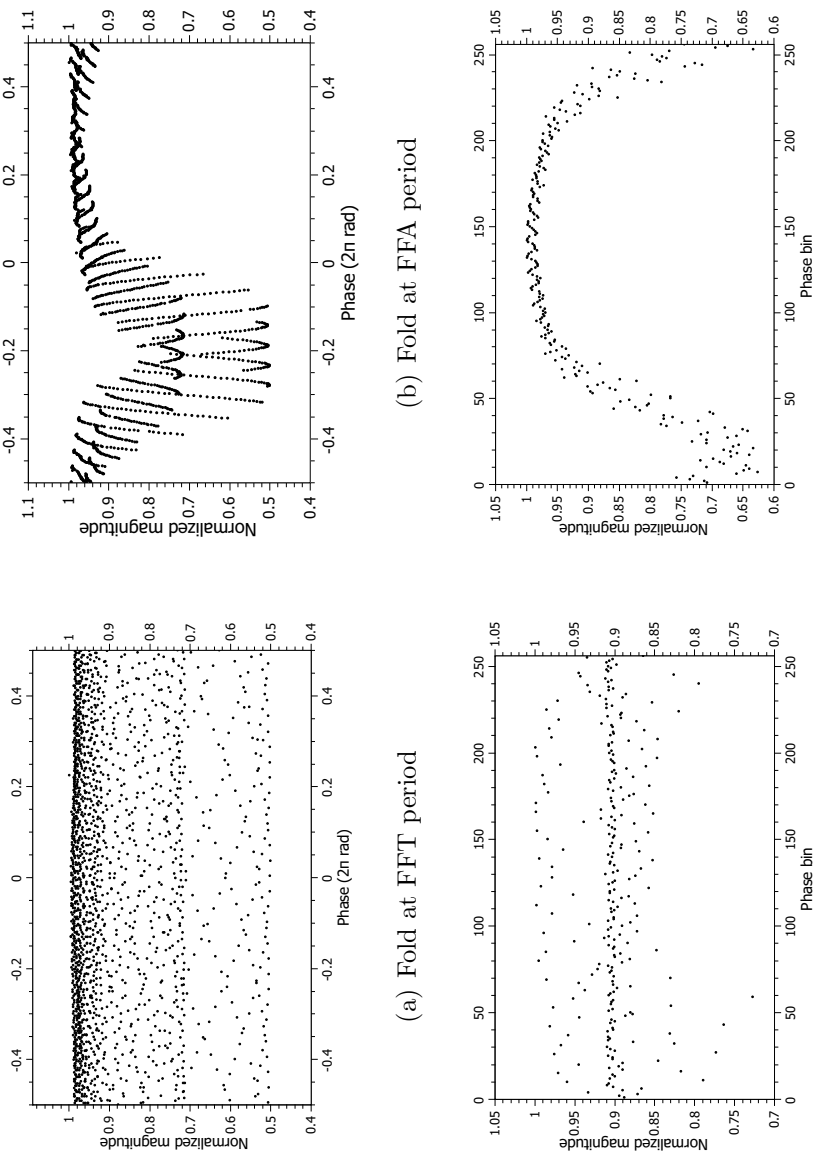


(c) FFT spectrum



(d) FFA periodogram

Figure 3.18: Results (a) for KID 4660997 — V1130 Cyg



(a) Fold at FFT period

(b) Fold at FFA period

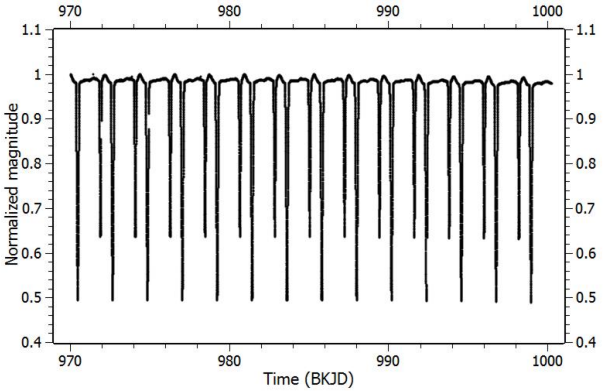
(c) Rebimed fold at FFT period

(d) Rebimed fold at FFA period

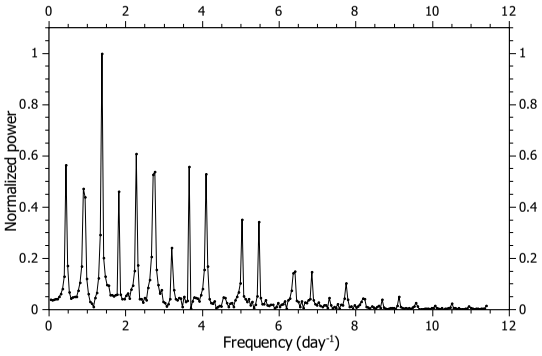
Figure 3.19: Results (b) for KID 4660997 — V1130 Cyg

Kepler ID	4544587
Time offset	2454833 BJD
Quarter	10 (short)
Type (ASAS)	Eclipsing binary
Period (ASAS)	2.1893
Bins	44062
FFT period	0.724 834(11691)
FFA period	0.730 418(11)

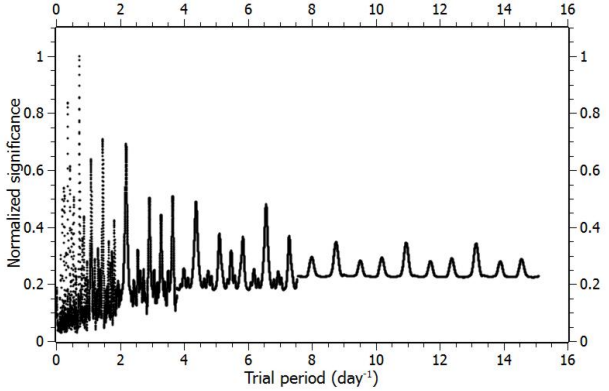
(a) Description & detected periods



(b) Lightcurve

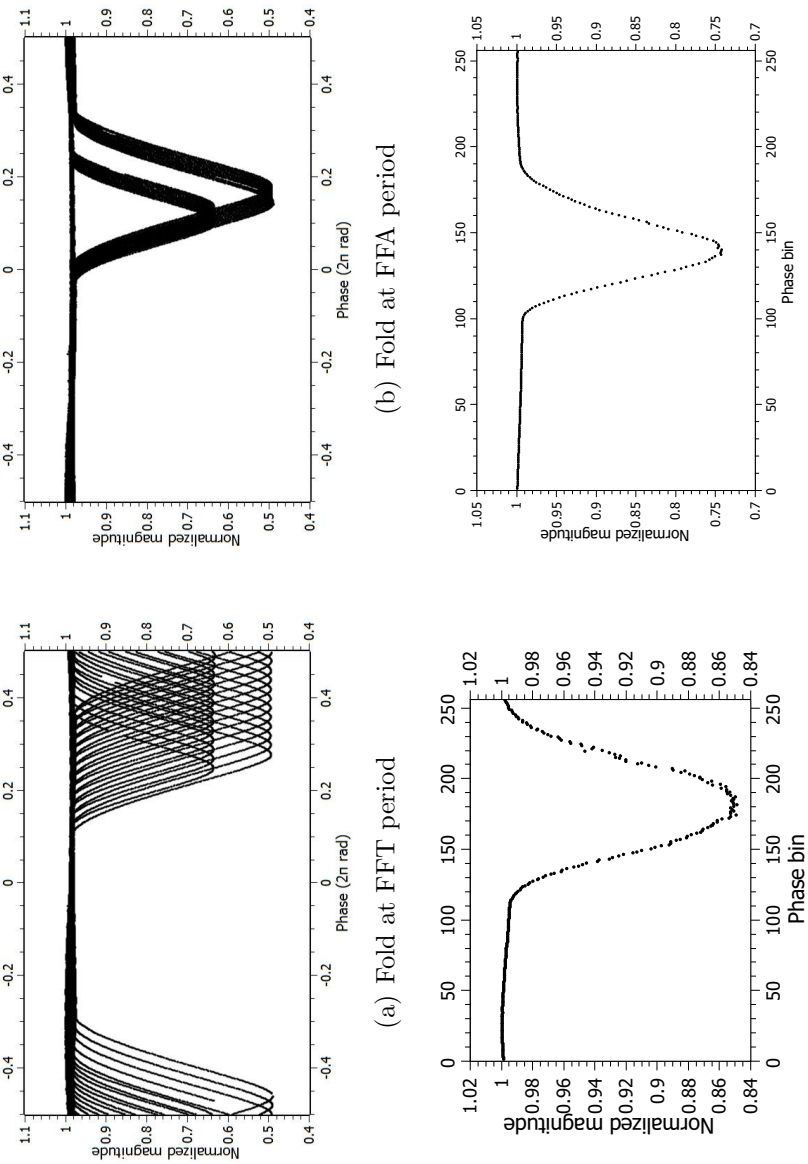


(c) FFT spectrum



(d) FFA periodogram

Figure 3.20: Results (a) for KID 4544587 —TYC 3124–1348–1



(a) Fold at FFT period

(b) Fold at FFA period

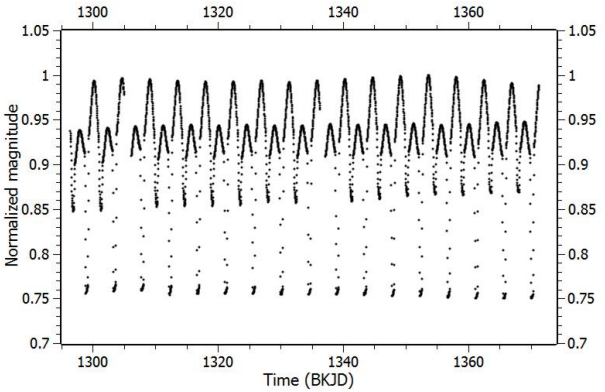
(c) Rebinned fold at FFT period

(d) Rebinned fold at FFA period

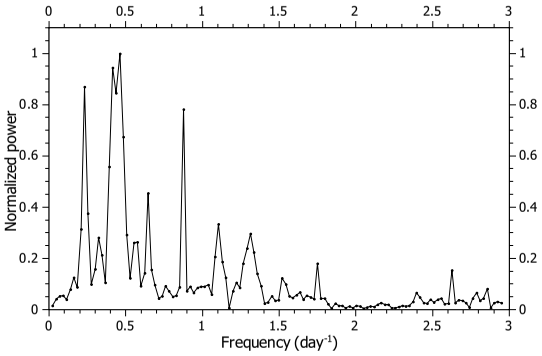
Figure 3.21: Results (b) for KID 4544587 — TYC 3124–1348–1

Kepler ID	8868650
Time offset	2454833 BJD
Quarter	14 (long)
Type (ASAS)	Eclipsing binary
Period (ASAS)	4.4472
Bins	3535
FFT period	2.172 630(54316)
FFA period	2.222 982(342)

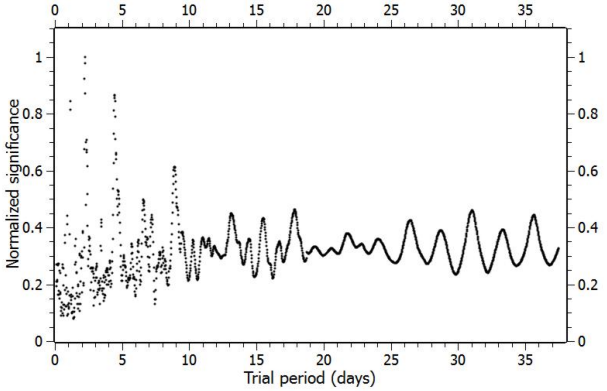
(a) Description & detected periods



(b) Lightcurve

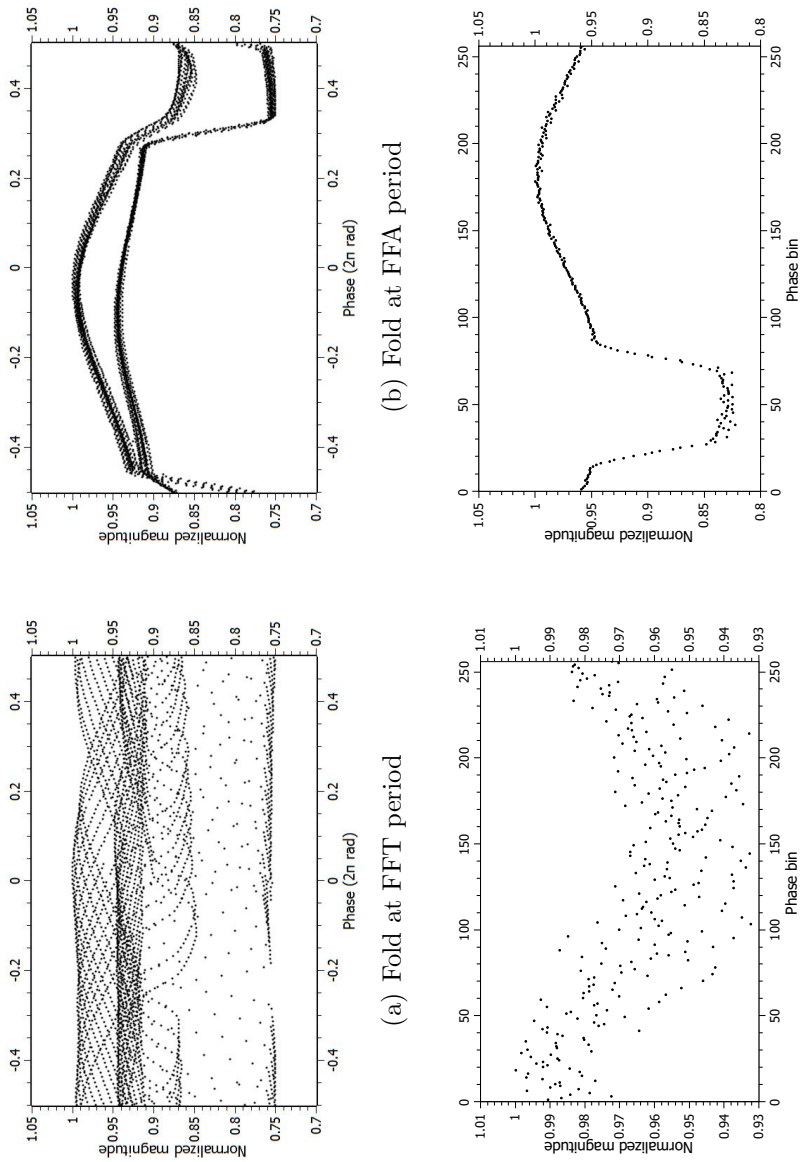


(c) FFT spectrum



(d) FFA periodogram

Figure 3.22: Results (a) for KID 8868650 — TYC 3541–2114–1



(a) Fold at FFT period

(b) Fold at FFA period

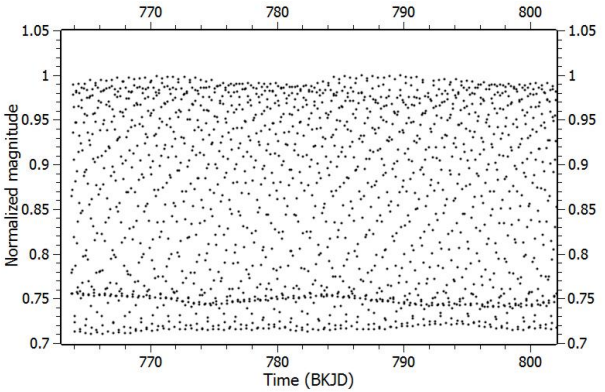
(c) Rebinned fold at FFT period

(d) Rebinned fold at FFA period

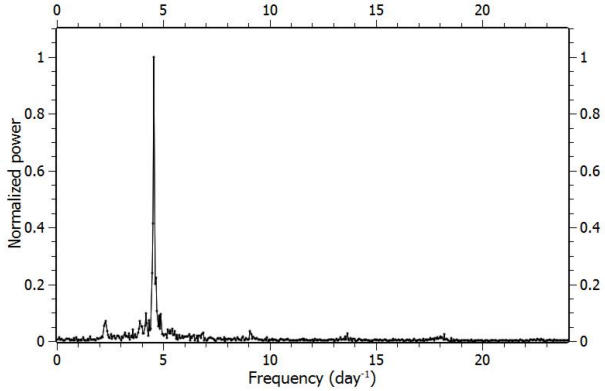
Figure 3.23: Results (b) for KID 8868650 — TYC 3541–2114–1

Kepler ID	10618251
Time offset	2454833 BJD
Quarter	8 (long)
Type (ASAS)	Eclipsing binary
Period (ASAS)	0.437417
Bins	1863
FFT period	0.219 075(1129)
FFA period	0.218 464(82)

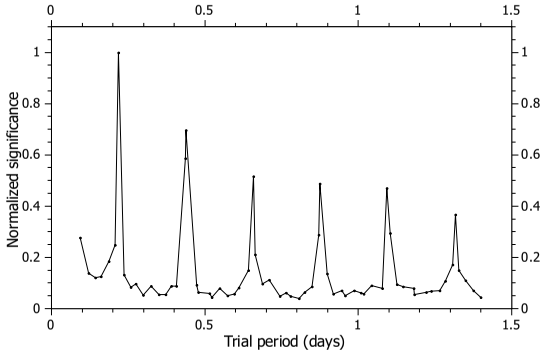
(a) Description & detected periods



(b) Lightcurve

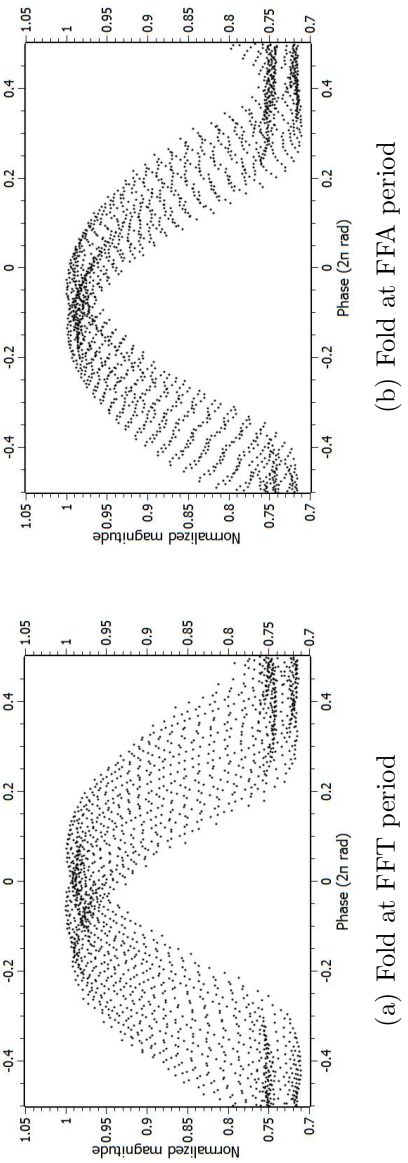


(c) FFT spectrum



(d) FFA periodogram

Figure 3.24: Results (a) for KID 10618251 — ASAS 194725+4104.0



(a) Fold at FFT period

(b) Fold at FFA period

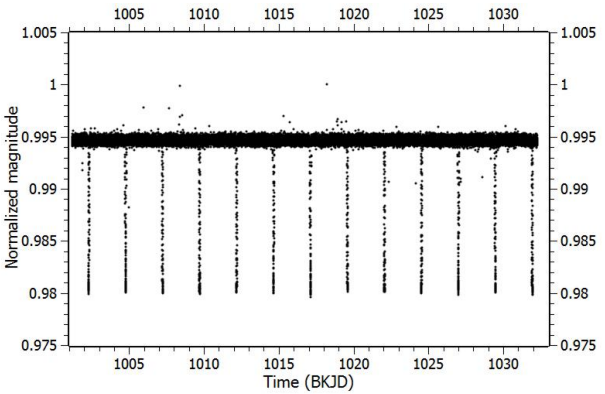
(c) Rebinned fold at FFT period

(d) Rebinned fold at FFA period

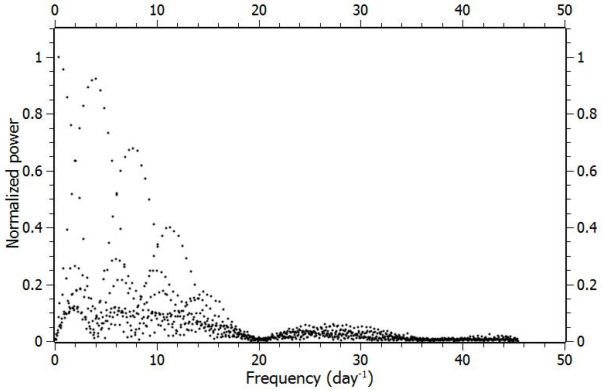
Figure 3.25: Results (b) for KID 10618251 — ASAS 194725+4104.0

Kepler ID	11446443
Time offset	2454833 BJD
Quarter	11 (short)
Type (Kepler)	Host of exoplanet
Period (Kepler)	2.47063
Bins	45183
FFT period	2.510 362(139465)
FFA period	0.003 477(1)

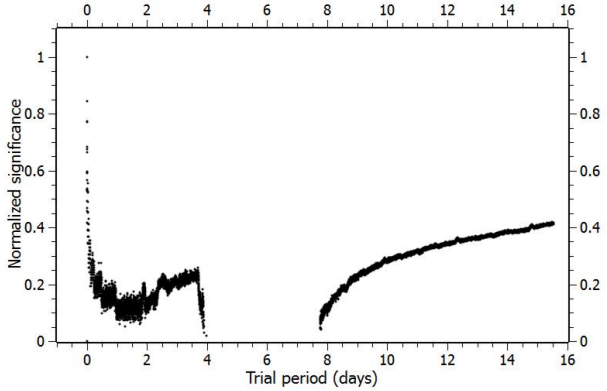
(a) Description & detected periods



(b) Lightcurve

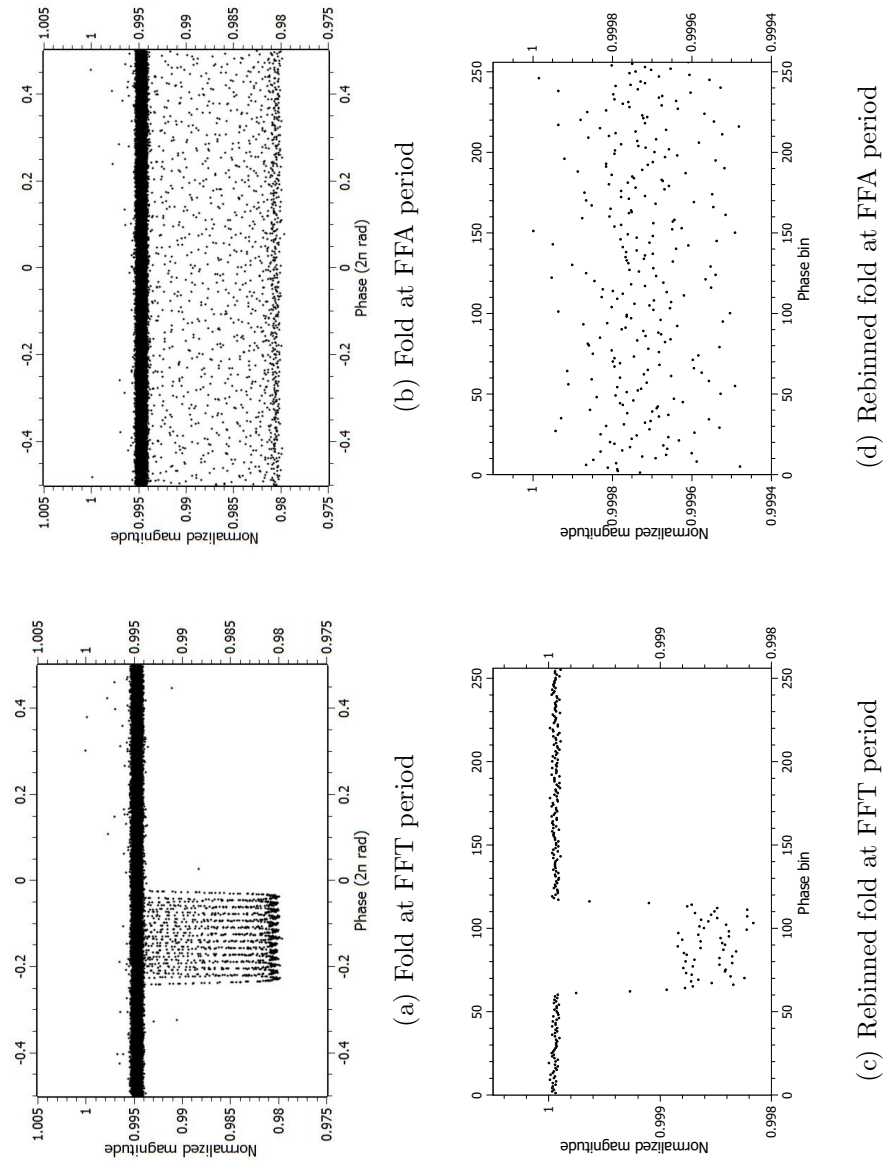


(c) FFT spectrum



(d) FFA periodogram

Figure 3.26: Results (a) for KID 11446443 — KOI–1 — TYC 3549–2811–1



(a) Fold at FFT period

(b) Fold at FFA period

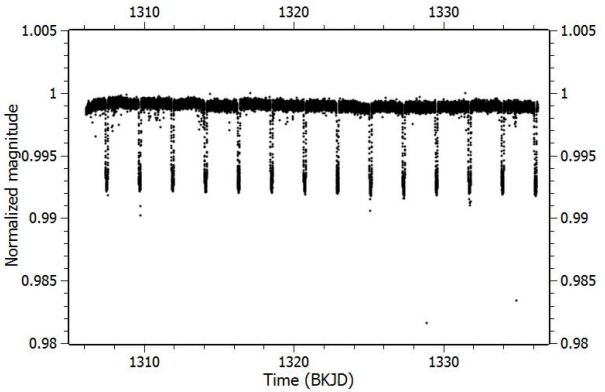
(c) Rebinned fold at FFT period

(d) Rebinned fold at FFA period

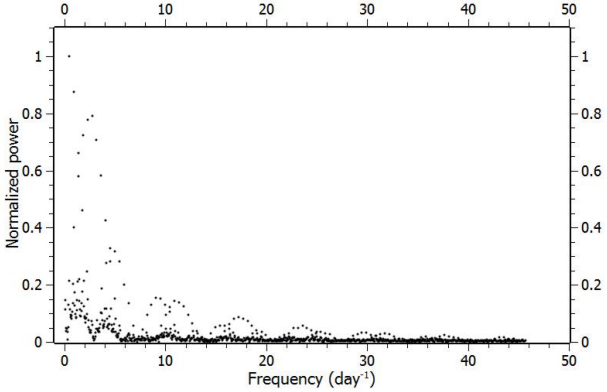
Figure 3.27: Results (b) for KID 11446443 — KOI–1 — TYC 3549–2811–1

Kepler ID	10666592
Time offset	2454833 BJD
Quarter	14.2 (short)
Type (Kepler)	Host of exoplanet
Period (Kepler)	2.204736
Bins	44149
FFT period	2.244 355(112218)
FFA period	0.003 497(1)

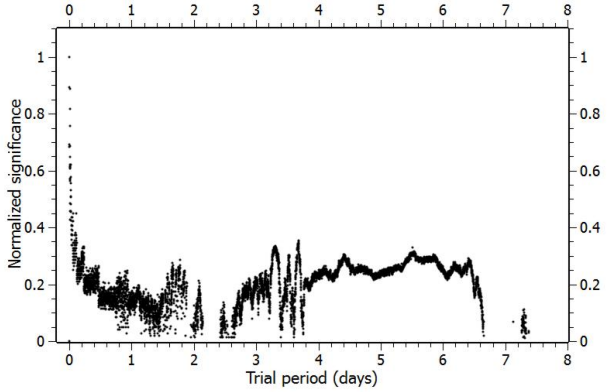
(a) Description & detected periods



(b) Lightcurve

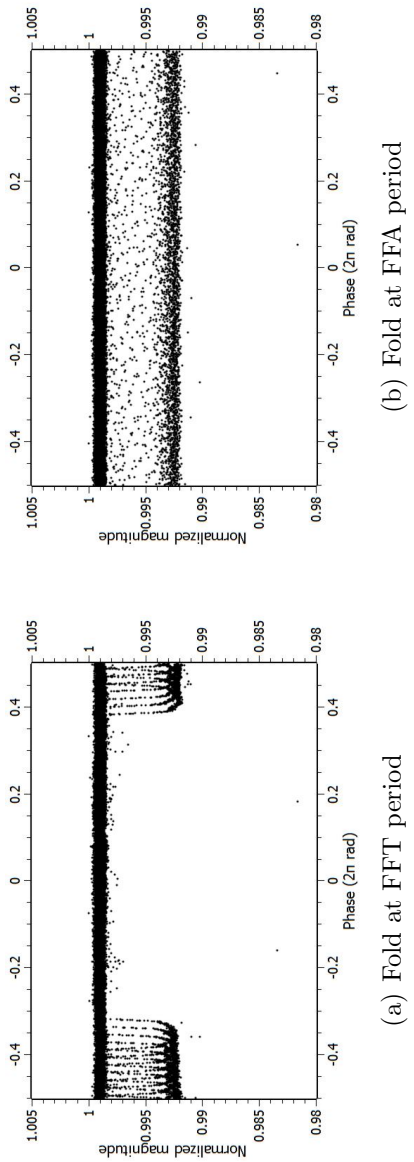


(c) FFT spectrum



(d) FFA periodogram

Figure 3.28: Results (a) for KID 10666592 — KOI–2 — BD+47 2846



(a) Fold at FFT period

(b) Fold at FFA period

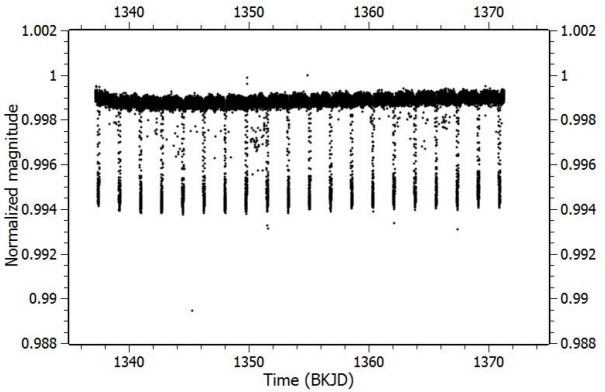
(c) Rebinned fold at FFT period

(d) Rebinned fold at FFA period

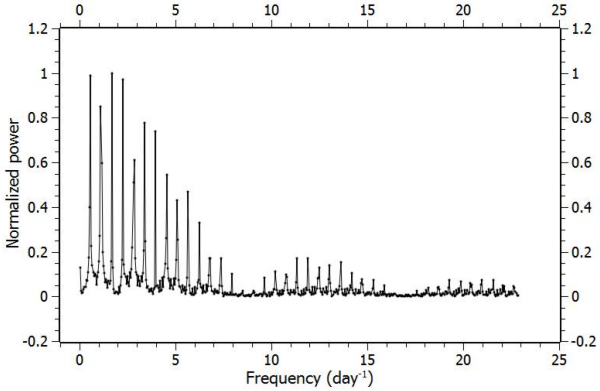
Figure 3.29: Results (b) for KID 10666592 — KOI–2 — BD+47 2846

Kepler ID	9941662
Time offset	2454833 BJD
Quarter	14.3 (short)
Type (Kepler)	Host of exoplanet
Period (Kepler)	1.7637
Bins	49958
FFT period	0.589 626(7758)
FFA period	0.002 735(1)

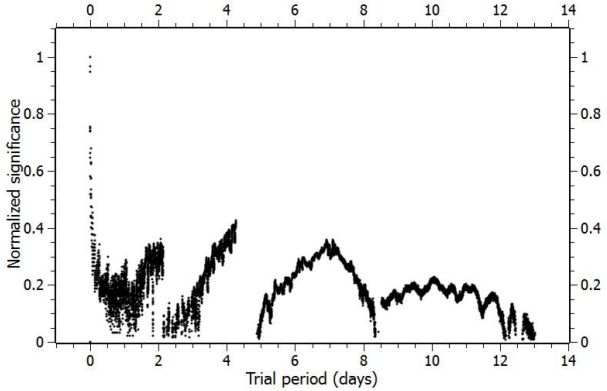
(a) Description & detected periods



(b) Lightcurve

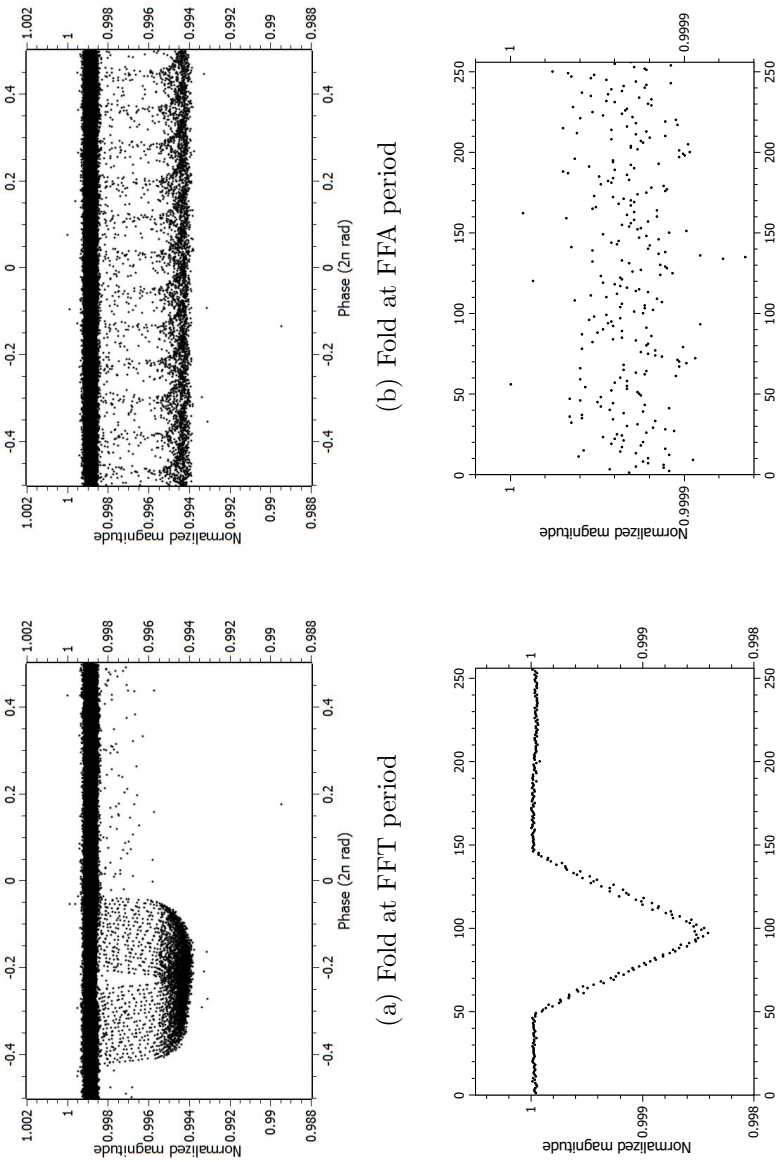


(c) FFT spectrum



(d) FFA periodogram

Figure 3.30: Results (a) for KID 9941662 — KOI–13 —BD+46 2629



(a) Fold at FFT period

(b) Fold at FFA period

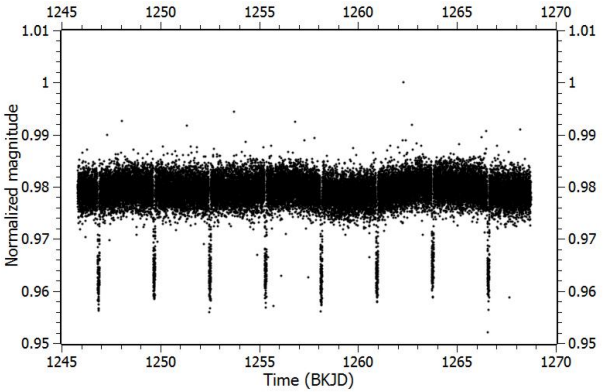
(c) Rebinned fold at FFT period

(d) Rebinned fold at FFA period

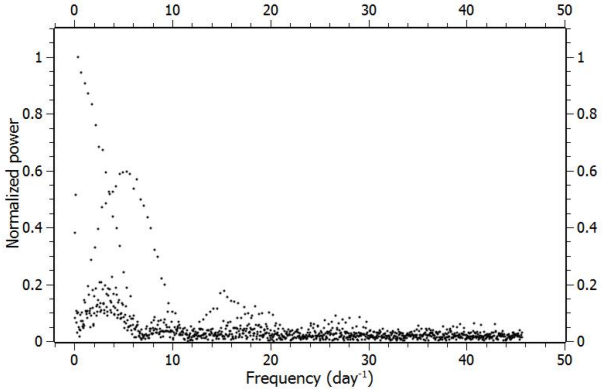
Figure 3.31: Results (b) for KID 9941662 — KOI–13 — BD+46 2629

Kepler ID	11414511
Time offset	2454833 BJD
Quarter	13.3 (short)
Type (Kepler)	Host of exoplanet
Period (Kepler)	2.816508
Bins	33464
FFT period	2.811 015(175689)
FFA period	0.002 961(1)

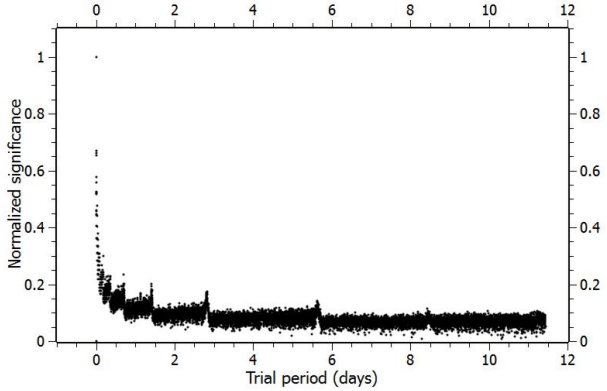
(a) Description & detected periods



(b) Lightcurve

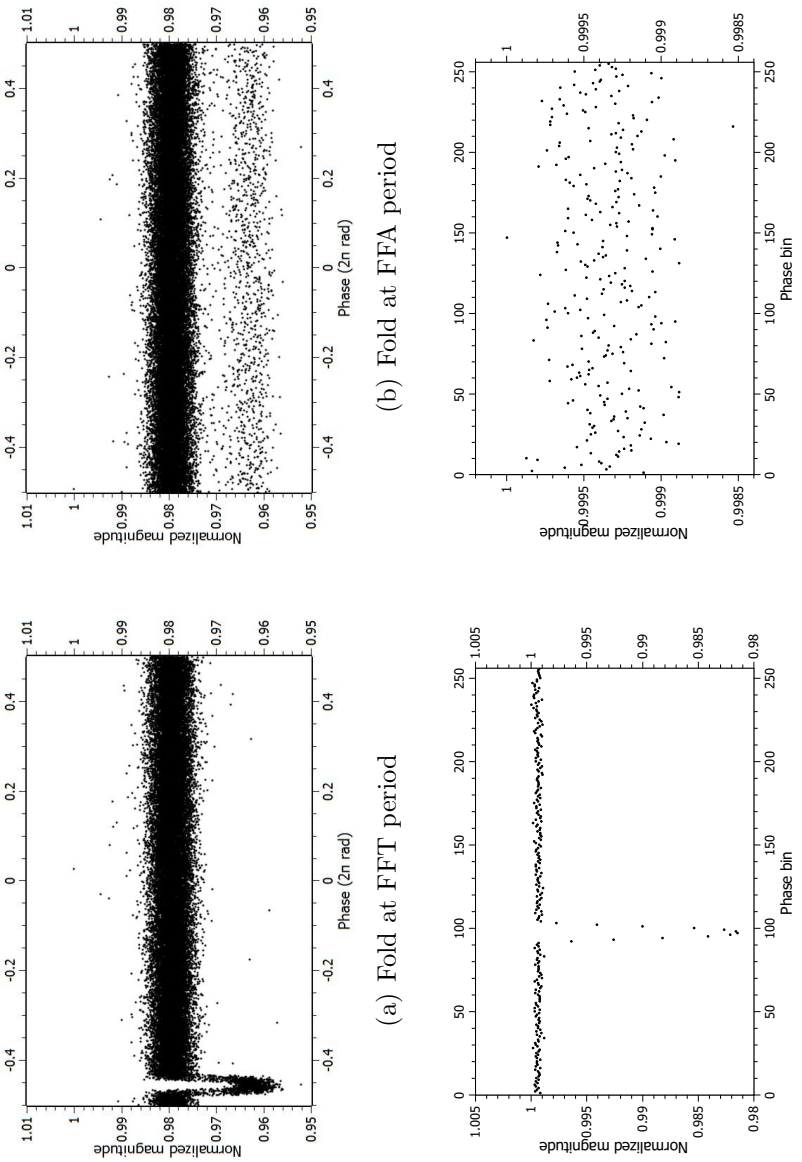


(c) FFT spectrum



(d) FFA periodogram

Figure 3.32: Results (a) for KID 11414511 — KOI–767 — 2MASS J19480371+4913310



(a) Fold at FFT period

(b) Fold at FFA period

(c) Rebinned fold at FFT period

(d) Rebinned fold at FFA period

Figure 3.33: Results (b) for KID 11414511 — KOI–767 — 2MASS J19480371+4913310

4

Discussion

4.1 Complexity test

FFT

The system that was used for the tests was relatively old and slow compared to a contemporary laboratory mainframe (Pentium® Dual–Core CPU at $2.2GHz$, under Microsoft Windows® 7 64-bit). In spite of this fact, FFT showed its powerful advantage of speed.

Even with an input of approximately 16.8 million bins (2^{24}), the calculation of the discrete Fourier transform took less than one minute. In other words, it takes a minute to analyze a collection of measurements taken every second — for about six and a half months.

The fitting of the time measurements to a curve of the type $a_1 + a_2 N \log_2 N$ returned a coefficient of determination R^2 equal to 0.99287, confirming the

complexity of the FFT algorithm and a good behavior of our implementation.

Though, a careful examination of the coefficient of the most important term ($N \log_2 N$) reveals an unexpected result: its value scales up for $N = 2^{16}$ and $N = 2^{24}$. The numbers indicate a total memory use that exceeds CPU cache levels[1] (L2, L3, RAM), resulting to the use of successively slower memory.

FFA

With an $O(n^2)$ computation complexity, FFA is extremely slow. For input sizes of thousands – which is common in observational astrophysics – FFT was completed faster by some hundred times.

Again, the fitting of the time measurements to a polynomial curve $a_0 + a_1 N + a_2 N^2$ returned a coefficient of determination R^2 equal to 0.99985 proving our theoretical estimates (see Appendix A.2).

The effect of the cache memory was not apparent in contrast to the FFT case. This probably is a result of the non-recursive implementation of the FFA algorithm which made use of only one buffer, minimizing the total memory allocation (only half of the L2 cache memory was used in the worst case).

4.2 Test on constructed light–curves

Sinusoidal signals

In the case of sinusoidal light–curves (Figure 3.5(a)), FFT successfully detected all signals with an accuracy of less than 10%. Only one exception existed out of the thousands combinations of different periods and noise levels, and corresponded to the maximum noise (standard deviation 3 times the

[1] The Cache L2 was 1MB. For four tables (input, output, two recursive buffers) of 2^{16} values, each needing a 4–byte computer word, we needed exactly 1MB of free cache

amplitude of the signal). This result does not contradict what we expected: FFT performs well on sinus-like signals.

On the other hand, FFA failed in many cases. In Figure 3.5(b) we can see a curve in the period–noise plane separating regions dominated by 'hits' or 'misses'. The good performance of FFA for low noise is trivial. The tendency to perform well on smaller periods confirms the fact that the signal–to–noise ratio is increased as the time–series are folded many times.

Examining the errors of period estimate for the 'hits' by both algorithms (Figure 3.6) we can expose a fundamental difference between the two algorithms. The FFT presents larger errors for low frequencies because of the spectral leakage and the noise hardly has any effect in detecting the signal. A possible explanation for the latter is that the spectrum of the Gaussian noise is flat, distributing the power in all frequencies and making a maximum in spectrum (due to a sinusoidal signal) clearer.

In contrast, FFA has a different behavior (Figure 3.6(b)). The folding procedure does not eliminate large part of the noise, making FFA sensitive to high noise. On the other axis, the period does not affect the success of the algorithm.

Rectangular signals

We expected to see an advantage of the FFA algorithm over the FFT in case of signals that have little resemblance to sine–wave. In Figures 3.7 to 3.9 we notice the expected fact that smaller pulses are harder to detect for both algorithms. What took us by surprise was that FFA never performed better than the FFT. Even for 5% duty cycle pulses, the latter was two times more successful.

Trying to explain the situation, we performed various tests on the algorithms using mathematical software and by–hand calculations to detect any defects of the implementations. The tests showed that the algorithms were functioning as expected.

The failure of FFA can be explained by the poor choice of significance measure. As referred in § 2.1.2, this measure is used to detect the most

prominent periodicity and depends on the power of the fold of the time–series. No calculations on the actual level of noise are performed and the measure is "by–hand" multiplied by the expected signal to noise ratio improvement (due to the central limit theorem and the nature of the Gaussian distribution of the noise).

Thus, for small duty cycles and/or high noise levels, the total power approximates the power of the noise, leading to a failure of detection. A more appropriate method for determining the signal to noise ratio should be used (like windowing functions or comparison with a filtered version of the fold).

4.3 Test on Kepler mission data

General picture

In Table 3.2 we can see that both FFT and FFA algorithms managed to detect the period of all Cepheid variables and eclipsing binaries with less than 10% error from the published values from the All Sky Automated Survey team (Pigulski et al., 2009). In the case of extra–solar planet transits, the previously discussed issue of FFA's significance measure rendered the transits undetected, whereas FFT performed quite satisfying — giving small errors in comparison to the published period in several papers (O'Donovan et al., 2006; Welsh et al., 2010; Shporer et al., 2011; Boruckiet al. , 2011).

Here we should note that the Kepler space telescope's high precision, gave high signal to noise ratio light–curves rendering FFT able to detect exoplanet transits.

Folding at estimates of period

The errors of period estimations, trusting the ASAS results(Pigulski et al., 2009), show deviations comparable to the sampling rate. Folding without

re–binning reveals that fact by showing multiple profiles close to each other. This is true for all but five case: (i) folding at FFT period for KID 4660997 (Figure 3.19(a)), were the deviation is about 5 times less than the sampling period and over 60 folds were produced because of the small period, and (ii) folding at FFA period for all host of exoplanets as detection failed.

Because of this fact, re–binned folding on a period near the real, can result to smooth and widened peaks. An good illustration of the smoothing lies on the comparison between Figure 3.11(c) and Figure 3.11(d). Example of the widening effect are all the FFT–period re-binned foldings of hosts of transiting extra–solar planets.

Though, as the errors of FFA were significantly smaller than those of FFT, the re–binned folds for the Cepheid variables and eclipsing binaries can be considered very accurate.

A good case of transit

The period of the exoplanet around KID 11414511 was detected with small error by FFT and the slightly big period (compared to the sampling rate) resulted to a very accurate profile: published (Boruckiet al. , 2011) values for period (2.8165077 days) and transit duration (156.4 minutes) suggest a duty cycle of 3.8%, whereas an examination of the Figure 3.33(d) reveals a 5.1% duty cycle (duration: 13 phase bins out of 256).

Wrong harmonic detection

In the case of the eclipsing binaries, two similar-in-form eclipses appear in one period. In most cases, the space between the two minima is about half the period, leading to a maximum in power for this value. In eccentric orbits with semi-major axes almost perpendicular to the line of sight of the observer, a different partition of time may be observed, e.g. 2:1 partition results to detection of a harmonic corresponding to one third of the actual period.

In order to face this problem, we created the non–re–binned folds where all observations where related to the observation time modulo the period. Accordingly, in case of a false harmonic, we are able to see all the profiles — especially in the eclipsing binary group.

The period estimations in Table 3.2 are the corrected ones. Though, in the output tables and figures, the original values are used. We performed re–binned folds, Figures 4.1 to 4.4 for all the four targets in the corrected period for FFA, to uncover the actual light–curve profiles of the eclipsing binaries.

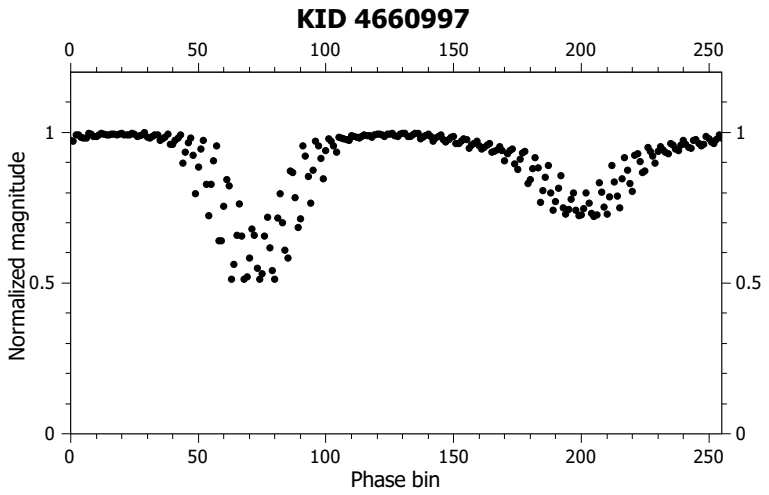


Figure 4.1: Corrected light–curve for the Algol–type eclipsing binary KID 4660997. The period of the folding was doubled to depict both the eclipses. The small period ($13^{\mathrm{h}}36^{\mathrm{m}}$) is comparable to the sampling rate (30^{m}) resulting to a certain amount of uncertainty which manifests as diffusion of the points in the plot.

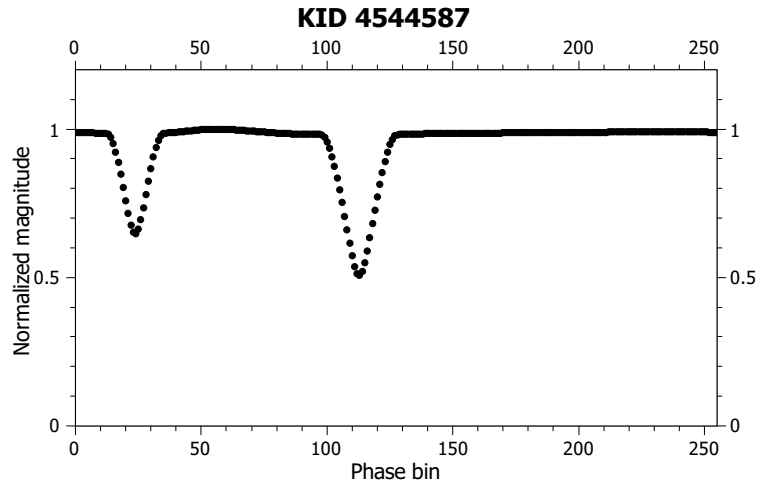


Figure 4.2: Corrected light–curve for the Algol–type eclipsing binary KID 4544587. The period of the folding was tripled to depict both the eclipses of this binary system with highly eccentric orbits. This is an example of how careful one should be interpreting the harmonics of a time–series.

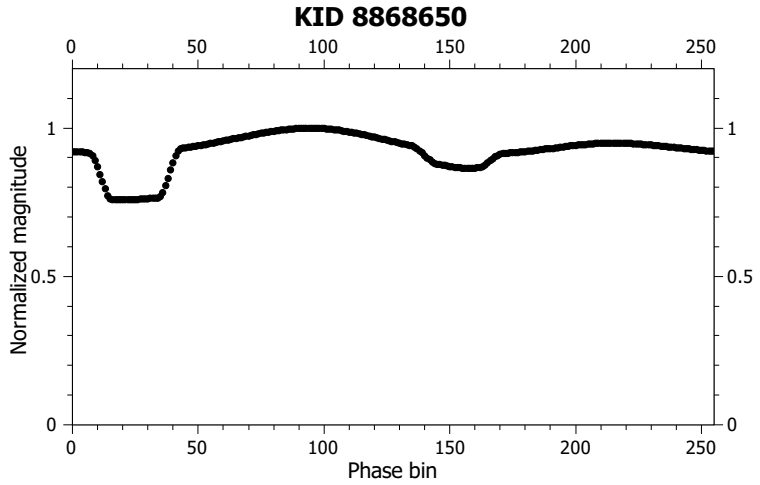


Figure 4.3: Corrected light–curve for the Algol–type eclipsing binary KID 8868650. The period of the folding was doubled to depict both the eclipses. Notice the square baseline of the minima, probably because of a very small in size companion, entering completely in the disc of the primary star during the primary eclipse and being totally obscured during the secondary eclipse.

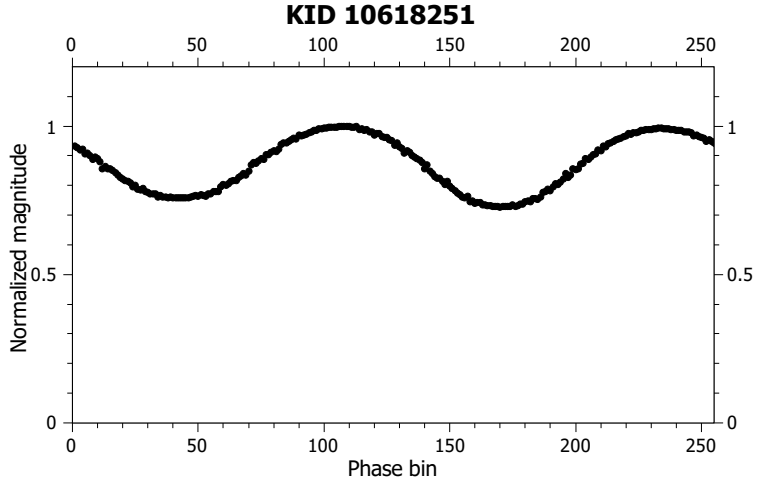


Figure 4.4: Corrected light–curve for the W UMa–type eclipsing binary KID 10618251. The period of the folding was doubled to depict both the eclipses. The high resemblance to a sinusoidal signal and the closeness of the minima points to a W Uma binary.

5

Conclusions

Reviewing the goals in § 1.6, we will investigate if they were met:

1. **Implementation of the algorithms and check of their good function**

   The algorithms were implemented and passed a series of tests that aimed to validate their output. Mathematical software and by–hand calculations were used for small inputs and we observed total convergence with our software output.

2. **Are they as time consuming as we expect them to be?**

   The algorithms run for various input sizes. The computational complexity was confirmed and the timing measurements showed the importance of 'clever' algorithms in astronomical surveys.

3. **What are their success rate when analyzing time–series of various parameters: shape, period, noise level?**

The FFT and FFA algorithms run for inputs of 10000 light–curves that were constructed by us. For sinusoidal shapes the FFT performed better, as expected. For small pulses (rectangular shape), although we expected FFA to surpass FFT, the test did not confirm this fact. We hypothesize that this is an outcome of wrong interpretation of the output rather than a problematic one.

4. **Can they be used to detect the period of three representative classes of periodic astronomical phenomena with small errors (Cepheid variables, eclipsing binaries, extra–solar planet transits)?**

   In spite of the rather simplistic methods for period searching, the combination of the two algorithms managed to detect the periods of our 12 targets. Only in the case of FFA and extra–solar planet transits we observed failure, a fact thoroughly discussed in § 4.3. Also, the effect of errors in the folded profile (elongation, smoothing etc.) and the detection of wrong harmonics were studied.

   A more careful approach for exploiting the output of both algorithms may have presented the proposed advantage of FFA in case of short pulses (Staelin, 1969; Burns & Clark, 1969; Kondratiev et al., 2009), which in turn would be useful for detecting transiting exoplanets.

6 Recommendations

Our experience raised serious issues and provoked the interest of the author in a more careful approach and even the construction of a better software package.

6.1 Changes

6.1.1 Programming conventions

The choice of floating-point variables for the bins of time–series is time-consuming. The speed of the algorithms could be improved by using a computer word to store two integer bins. Careful normalization is needed to avoid circular sign changes.

6.1.2 Proper statistical significance measures

An otherwise perfect output of a discrete Fourier transform or a periodogram, can be rendered useless if improper statistical methods are used. A method for computing the an estimate of the signal–to–noise ratio should be employed (perhaps window functions in the case of FFA and harmonic manipulation for FFT).

6.1.3 Regularization

Our method for transforming unevenly–spaced time–series to uniform induces errors in phase bins and only works for nearly–uniform times: big gaps prevent the use of FFT and FFA. Possible solutions:

- Interpolation instead of regularization.
- Interpolation for the regions of small gaps.
- Re–binning the time–series for larger intervals risking loss of information, but speeding the process.

6.2 Future extensions

6.2.1 Study of trends and colored noise

The effect of trends and other types of noise should be studied. Although careful data reduction of the observations can dispose unwanted trends and noise, sometimes these features are physical. Also, the Gaussian distribution of the noise is an approximation. In most physical procedures, the noise is Poisson distributed.

6.2.2 FFT/FFA/LS cooperation

We propose a fast method for automatic period search in light–curves:

1. Trend removal and noise reduction using smoothing/moving averages.
2. Re–binning and interpolation of the data to produce uniform–in–time time series.
3. Production of Fourier transform to detect a number of regions of possible periodicities, exploiting the speed ($O\left(nlogn\right)$).
4. Execution of FFA algorithm for the above narrow (the $O\left(n^2\right)$) won't induce significant delays) regions and new collection of possible periods with greater accuracy.
5. Use of Lomb–Scargle Periodogram method that analyzes unevenly–spaced time–series, on the original non–re–binned data in the even more narrow ranges that FFA returned.

This page intentionally left blank

A

Mathematical definitions and proofs

A.1 Big O notation

The exact running time of an algorithm, $f(n)$, depends on the length of the input, n, and in most cases is a very complex expression. As we usually are interested in the effectiveness of the performance of an algorithm when n is large (and it is likely to be time–consuming), we use asymptotic analysis.

This is done by considering only the highest order term of the expression and disregarding the coefficient of that term. For example, $\frac{2}{log2}n^2logn + 3.75n^2 + 28$ becomes n^2logn.

In the text there are some instances of the notation $O(f'(n))$ which refers to that process. The mathematical definition of the so–called 'big O notation' is:

$$f(n) = O(f'(n)) \text{ as } x \to \infty$$

if and only if a positive constant M and a large n_0 exist so that

$$|f(n)| \leq M|f'(n)| \text{ for all } n \geq n_0$$

A.2 Computational complexity of FFA for blind searches

- The complexity of FFA on a N–length–ed data for a basic period P is proportional to $N \log_2 \left\lfloor \frac{N}{P} \right\rfloor$
- In the worst case scenario, we apply the algorithm to the range of periods $[2, N/2]$
- This results to a total complexity $O(N^2)$

Proof

$$\sum_{P=2}^{N/2} \left(N \log_2 \left\lfloor \frac{N}{P} \right\rfloor \right) \simeq$$
$$\simeq \sum_{P=2}^{N/2} \left[N \log_2 \left(\frac{N}{P} \right) \right] \simeq$$
$$\simeq \int_2^{N/2} N \log_2 \frac{N}{P} dP =$$
$$= \frac{1}{\ln 2} \int_2^{N/2} N \ln \frac{N}{P} dP =$$
$$= \frac{1}{\ln 2} \int_2^{N/2} (NlnN - NlnP) =$$

$$= \frac{1}{\ln 2} \left[(NlnN)\, P - N\,(PlnP - P) \right]_{P=2}^{P=N/2} =$$
$$= \frac{1}{ln2} \left[\frac{N^2}{2} - 2ln2 \cdot N - 2\,(1 - ln2)\, N \right] =$$
$$= O\left(N^2\right)$$

A.3 Regularization formulæ

Synopsis

- Observed values: t_i
- Expected values: $\tau_i = \tau_0 + i \cdot p_s$
- Goodness–of–fit measure: $g = \sum_{i=0}^{N-1} (t_i - \tau_i)^2$
- Determination of τ_0 and p by solving $\frac{\partial g}{\partial \tau_0} = 0$ and $\frac{\partial g}{\partial p_s} = 0$

Derivation

$$\frac{\partial g}{\partial \tau_0} = 0 \qquad \frac{\partial g}{\partial p_s} = 0$$
$$\frac{\partial}{\partial \tau_0} \sum_{i=0}^{N-1} (t_i - \tau_i)^2 = 0 \qquad \frac{\partial}{\partial p_s} \sum_{i=0}^{N-1} (t_i - \tau_i)^2 = 0$$
$$\sum_{i=0}^{N-1} \left[\frac{\partial}{\partial \tau_0} (t_i - \tau_0 - i \cdot p_s)^2 \right] = 0 \qquad \sum_{i=0}^{N-1} \left[\frac{\partial}{\partial p_s} (t_i - \tau_0 - i \cdot p_s)^2 \right] = 0$$
$$\sum_{i=0}^{N-1} \left[(t_i - \tau_0 - i \cdot p_s)\,(-2) \right] = 0 \qquad \sum_{i=0}^{N-1} \left[(t_i - \tau_0 - i \cdot p_s)\,(-2i) \right] = 0$$
$$\sum_{i=0}^{N-1} t_i - \sum_{i=0}^{N-1} \tau_0 - p_s \sum_{i=0}^{N-1} i = 0 \qquad \sum_{i=0}^{N-1} (i \cdot t_i) - \tau_0 \sum_{i=0}^{N-1} i - p_s \sum_{i=0}^{N-1} i^2 = 0$$

Using that

$$\sum_{i=0}^{N-1} i = \frac{N\left(N-1\right)}{2}$$
$$\sum_{i=0}^{N-1} i^2 = \frac{N\left(N-1\right)\left(2N-1\right)}{6}$$

we conclude to the system of equations

$$N\tau_0 + \frac{N\left(N-1\right)}{2} p_s = \sum_{i=0}^{N-1} t_i$$
$$\frac{N\left(N-1\right)}{2}\tau_0 + \frac{N\left(N-1\right)\left(2N-1\right)}{6} p_s = \sum_{i=0}^{N-1} \left(i \cdot t_i\right)$$

Finally, there is a unique solution:

$$\tau_0 = \frac{2}{N^2+N}\left[\left(2N-1\right)\sum_{i=0}^{N-1} t_i - 3\sum_{i=0}^{N-1}\left(i \cdot t_i\right)\right]$$
$$p_s = \frac{6}{N\left(N^2-1\right)}\left[2\sum_{i=0}^{N-1}\left(i \cdot t_i\right) - \left(N-1\right)\sum_{i=0}^{N-1} t_i\right]$$

B

Random number generators

The artificial light–curves that were used to check the performance of FFA and FFT algorithms should be contaminated with Gaussian noise (as explained in § 1.2). The generation of such random numbers required the following functions. Note that one should avoid the use of flawed built–in functions (like `rand()`) but for the purpose of our investigations more sophisticated methods are rendered unnecessary (Press et al., 2007).

B.1 Standard uniform distribution `rnd_uniform()`

The generation of a random number of the standard uniform distribution[1] is trivial as the standard C header file `stdlib.h` provides the `rand()` function

[1] Constant probability density function in the range [0,1] and zero everywhere else

which returns an integer in the range `[0, RAND_MAX]`. The latter is a constant defined in the same header file and usually equals 32767.

Thus, we divide the return value by `RAND_MAX` and multiply with `1.0` to prevent the compiler from performing an integer division (which would subsequently result to `0.0`).

The resolution (32768 different levels) is more than enough for the 'testing' purposes we facilitate this function.

```
1 float rnd_uniform()
2 {
3     return rand()*1.0/RAND_MAX;
4 }
```

Listing B.1: A uniform random number generator

B.2 Standard normal distribution `rnd_gaussian()`

Here lies an implementation of Marsaglia's polar method (Marsaglia & Bray, 1964) for generating random values of standard normal distribution[1]. The function utilizes the `rnd_uniform()` function described above.

```
1  float rnd_gaussian()
2  {
3      float R=0,a,b;
4      do
5      {
6          a=rnd_uniform()*2-1;
7          b=rnd_uniform()*2-1;
8          R=a*a+b*b;
9      } while (R>=1 || R==0);
10
11     return a*sqrt(-2*log(R)/R);
12 }
```

Listing B.2: A normal random number generator

[1] zero mean and unit variance normal distribution

"Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live."

John F. Woods

C Main code

C.1 Data types

```
1  typedef double TIME; // type for time in BJD
2  typedef float  BIN;  // type for measurement in arbitrary units
3
4  typedef struct
5  {
6     int  KID;          // Kepler ID
7     TIME ASASperiod;   // All Sky Automated Survey Period
8     char *notes;       // NOTES
9  } STAR_INFO;
10
11 typedef struct
12 {
13    BIN  *bins;        // list of measurements
14    int  size;         // size of tables
15    TIME *times;       // list of observation times in days
```

```
16    TIME timeref;      // time reference in BJD
17 } TIME_BIN_TABLE;
18
19 typedef struct
20 {
21    BIN  *bins;        // table of measurements
22    int  size;         // size of table
23    TIME timeref;      // time reference in BJD
24    TIME timeerr;      // regularization time error
25    TIME interval;     // sampling rate
26 } BIN_TABLE;
27
28 typedef struct
29 {
30    float period;         // detected period
31    float dp;             // error in period
32    float significance;   // significance
33    int outcome;          // _SUCCESS or _FAILURE
34 } FFA_RESULT;
```

Listing C.1: Various newly defined data types

C.2 Regularize routine

```
1  int REGULARIZE(const TIME_BIN_TABLE *src,BIN_TABLE *dst)
2  {
3     printf("\nREGULARISE REPORT\n----------\n");
4     int n=(*src).size;
5     (*dst).size=n;
6     (*dst).bins=malloc(n*sizeof(BIN));
7
8     int i;
9     TIME ti=0,sumti=0,sumiti=0;
10    for(i=0;i<n;i++)
11    {
12       (*dst).bins[i]=(*src).bins[i];    // copy values
```

```
13
14          ti=(*src).times[i];       // but also do the
15          sumti+=ti;                // ...statistics for...
16          sumiti+=i*ti;             // ...the regularization.
17      }
18
19      TIME delta_tau=(2.0/(1.0*n-1)*sumiti - sumti)*6/(n*(n+1.0));
20      TIME tau=(((2.0*n-1)/3)*sumti-sumiti)*6/(n*(n+1.0));
21      TIME error=0;
22      for(i=0;i<n;i++)
23      {
24          ti=(*src).times[i]-tau-i*delta_tau;
25          ti=ti*ti;
26          error+=ti;
27      }
28      error=sqrt(error/n);
29
30      (*dst).timeref=(*src).timeref+tau;
31      (*dst).interval=delta_tau;
32      (*dst).timeerr=error;
33
34      printf("New time offset \t%lf BJD\n",(*dst).timeref);
35      printf("Sampling rate \t\t%lf min\n",(*dst).interval*24*60);
36      printf("Regularisation error \t%lf min\n",(*dst).timeerr
            *24*60);
37
38      return _SUCCESS;
39  }
```

Listing C.2: The regularization routine

C.3 FFT implementation

```
1  void RECURSIVE_FFT(complex *input,complex *output,int N,int step
       )
2  {
```

```
3     int i;
4     if (step<N)
5     {
6        RECURSIVE_FFT(output,input,N,step*2);
7        RECURSIVE_FFT(output+step,input+step,N,step*2);
8
9        for(i=0;i<N;i+=2*step)
10       {
11          complex tmp = cexp(-I*M_PI*i/N)*output[i+step];
12          input[i/2] = output[i] + tmp;
13          input[(i+N)/2]=output[i] - tmp;
14       }
15    }
16 }
17
18 float FFT(const BIN_TABLE *data)
19 {
20    printf("\nFFT REPORT\n----------\n");
21    int n=(int) pow(2,(int) log2((*data).size));
22
23    complex *input=malloc(n*sizeof(complex));
24    complex *output=malloc(n*sizeof(complex));
25
26    int i;
27    for(i=0;i<n;i++)
28    {
29       input[i]=(complex) (*data).bins[i];
30       output[i]=(complex) (*data).bins[i];
31    }
32
33    RECURSIVE_FFT(input,output,n,1);
34
35    // Save power spectrum
36    FILE *fftfile=fopen("fftspectrum.txt","w");
37    if(fftfile==NULL)
38    {
39       printf("\n\nError opening file\n\n");
40       return _FAILURE;
41    }
```

```
42      printf("> Spectrum file created.\n\n");
43      fprintf(fftfile,"Frequency,Magnitude\n");
44
45      float max=0;
46      float x;
47      int imax=0;
48
49      for(i=1;i<n/2;i++)
50      {
51          x=cabs(input[i]);
52          if(i<FFToutput) fprintf(fftfile,"%f,%f\n",i/((*data).
                interval*(1.0*n)),(x));
53          if (x>max)
54          {
55              max=x;
56              imax=i;
57          }
58      }
59
60      fclose(fftfile);
61
62      float f=imax;
63      float pmin=(*data).interval*n/(f+0.5);
64      float pmax=(*data).interval*n/(f-0.5);
65      float period=(pmax+pmin)/2;
66      float dp=(pmax-pmin)/2;
67
68      printf("Frequancy: \t\t%d bins\n",imax);
69      printf("Period: \t\t%f days \t+/- %f\n",period,dp);
70
71      return period;
72  }
```

Listing C.3: The implementation of FFT algorithm

C.4 FFA implementation

```
1  FFA_RESULT FFA(BIN *input,int bins,int P)
2  {
3     FFA_RESULT result;
4
5     // dimensions, blocks, etc.
6     int M=(int) bins/P;
7     int steps=(int) log2(M);
8     M=(int) pow(2,steps);
9     int N=M*P;
10
11
12    // allocating memory for temporary arrays
13    BIN *A=malloc(N*sizeof(BIN));
14    BIN *B=malloc(N*sizeof(BIN));
15
16    // initializing
17    int i;
18    for(i=0;i<N;i++)
19    {
20       A[i]=input[i];
21       B[i]=0;
22    }
23
24    // FFA
25    int step,k,shift;
26    for(step=1;step<=steps;step++)
27    {
28       int blocklen=(int) pow(2,step);
29       int blocks=(int) M/blocklen;
30       int block;
31       for(block=0;block<blocks;block++)
32       {
33          int row;
34          for(row=0;row<blocklen;row++)
35          {
36             for(k=0;k<P;k++)
```

```
37              {
38                  shift=(int) (row+1)/2;
39                  B[(block*blocklen+row)*P+k]=
40                  A[(block*blocklen+((int) row/2))*P+k]+
41                  A[(block*blocklen+((int) (row+blocklen)/2))*P+(k+
                        shift)%P];
42              }
43          }
44      }
45      for(i=0;i<N;i++) A[i]=B[i];
46  }
47
48  // period and significance estimate
49
50  float max=0;
51  float period_at_max=0;
52
53  int fold;
54  for(fold=0;fold<M;fold++)
55  {
56      float meanx=0;
57      float meanx2=0;
58      for(i=0;i<P;i++)
59      {
60          BIN x=A[fold*P+i];
61          meanx+=x;
62          meanx2+=x*x;
63      }
64      meanx=meanx/P;
65      meanx2=meanx2/P;
66      float SD=sqrt(P/(P-1.0))*sqrt(meanx2-meanx*meanx);
67      if(SD>max)
68      {
69          max=SD;
70          period_at_max=P+(fold/(1.0*M-1.0));
71      }
72  }
73  free(A);
74  free(B);
```

```
75
76      result.outcome=_SUCCESS;
77      result.significance=max/sqrt(M);
78      result.dp=1/(2.0*(M-1));
79      result.period=period_at_max;
80
81      return result;
82  }
83
84  FFA_RESULT FFAsearch(BIN_TABLE *data,int periodmin,int periodmax
        )
85  {
86      int bins=(*data).size;
87      printf("\nFFA REPORT\n----------\n");
88      BIN *input=malloc(bins*sizeof(BIN));
89      int i;
90      for(i=0;i<bins;i++)
91      {
92          input[i]=(*data).bins[i];
93      }
94
95      int period,count=0;
96      FFA_RESULT result;
97      result.period=0;
98      result.significance=0;
99      result.outcome=_FAILURE;
100
101     FILE *ffafile=fopen("ffaperiodogram.txt","w");
102     if(ffafile==NULL)
103     {
104         printf("\n\nError opening file\n\n");
105         return result;
106     }
107     fprintf(ffafile,"Period,Significance\n");
108     printf("> Periodogram file created.\n\n");
109
110     BIN max=0;
111     float period_at_max=0,dp_at_max,x,meanx=0,meanx2=0;
112
```

```
113     for(period=periodmin;period<=periodmax;period++)
114     {
115         count++;
116 
117         result=FFA(input,bins,period);
118         x=result.significance;
119         fprintf(ffafile,"%f,%f\n",(*data).interval*result.period,
                result.significance);
120 
121         meanx+=x;
122         meanx2+=x*x;
123 
124         if (x>max)
125         {
126             period_at_max=result.period;
127             dp_at_max=result.dp;
128             max=x;
129         }
130     }
131 
132     meanx=meanx/count;
133     meanx2=meanx2/count;
134     float stdev = sqrt(count/(1.0*count-1.0))*sqrt(meanx2-meanx*
            meanx);
135     result.significance=(max-meanx)/stdev;
136     result.period=period_at_max*(*data).interval;
137     result.dp=dp_at_max*(*data).interval;
138     result.outcome=_SUCCESS;
139 
140     printf("Period: \t\t%f days \t+/- %f\n",result.period,result.
            dp);
141 
142     fclose(ffafile);
143     return result;
144 }
```

Listing C.4: The implementation of FFA algorithm

C.5 Folding routine

```
1  int FOLD(TIME_BIN_TABLE *data,TIME p,char *path)
2  {
3     FILE *file=fopen(path,"w");
4     fprintf(file,"phase,magnitude\n");
5
6     int n=(*data).size;
7     TIME start=(*data).times[0];
8     TIME end=(*data).times[n];
9
10    BIN b,max=0;
11    TIME t,timeatmax=0;
12    int i;
13    for(i=0;i<n;i++)
14    {
15       b=fabs((*data).bins[i]);
16       t=(*data).times[i];
17
18       if(t<start) start=t;
19       if(t>end) end=t;
20
21       if(b>max)
22       {
23          max=b;
24          timeatmax=t;
25       }
26    }
27
28    for(i=0;i<n;i++)
29    {
30       t=(*data).times[i]-0.5*timeatmax;//start;
31       t=(t-trunc(t/p)*p)/p-0.5;
32       fprintf(file,"%lf,%f\n",t,(*data).bins[i]/max);
33    }
34
35    printf("\n> Folding at p=%lf saved to: %s\n",p,path);
36    fclose(file);
```

```
37 }
```

Listing C.5: The folding routine

C.6 Re–binned folding routine

```
1  int FOLD_REBINNED(TIME_BIN_TABLE *data,int REBINNING, TIME p,
       char *path)
2  {
3      int i;
4      FILE *file=fopen(path,"w");
5      fprintf(file,"phase bin,magnitude\n");
6
7      int n=(*data).size;
8      BIN *bins=malloc(REBINNING*sizeof(BIN));
9      int *bincount=malloc(REBINNING*sizeof(int));
10     for(i=0;i<REBINNING;i++)
11     {
12         bins[i]=0;
13         bincount[i]=0;
14     }
15
16     TIME start=(*data).times[0];
17     TIME end=(*data).times[n-1];
18
19     BIN b;
20     TIME t,timeatmax=0;
21
22     for(i=0;i<n;i++)
23     {
24         TIME t=(*data).times[i]-start;
25         t=t/p-trunc(t/p);
26         int pos=(int) (t*REBINNING);
27         if (pos>=REBINNING) pos=REBINNING-1;
28         bins[pos]+=(*data).bins[i];
29         bincount[pos]++;
```

```
30      }
31
32      BIN max=0;
33      for(i=0;i<REBINNING;i++)
34      {
35          int norm=bincount[i];
36          if(norm!=0) bins[i]/=norm;
37          if (fabs(bins[i])>=max) max=fabs(bins[i]);
38      }
39
40      for(i=0;i<REBINNING;i++)
41      {
42          int norm=bincount[i];
43          if (norm!=0) fprintf(file,"%d,%f\n",i+1,bins[i]/max);
44      }
45
46      printf("\n> Rebinned (%d) folding at p=%lf saved to: %s\n",
            REBINNING,p,path);
47      fprintf(outputfile,"\n> Rebinned (%d) folding at p=%lf saved
            to: %s\n",REBINNING,p,path);
48      fclose(file);
49  }
50  }
```

Listing C.6: The rebinned folding routine

Index

This page intentionally left blank

Bibliography

Borucki, W. J. et al. (2011). Characteristics of planetary candidates observed by kepler. ii. analysis of the first four months of data. *The Astrophysical Journal*, *736*(1), 19.

Burns, W. R. & Clark, B. G. (1969). Pulsar search techniques. *Astronomy & Astrophysics*, *2*, 280–287.

Christiansen, J. L., Van Cleve, J. E., Jenkins, J. M., Caldwell, D. A., Allen, C. L., Barclay, T., Bryson, S. T., Burke, C. J., Clarke, B. D., Cote, M. T., Fanelli, M. N., Gilliland, R. L., Girouard, F., Haas, M. R., Hall, J. R., Ibrahim, K., Kinemuchi, K., Klaus, T. C., Kolodziejczak, J. J., Li, J., Machalek, P., & McCauliff, S. D. (2011). Kepler data release 11 notes. *KSCI–19051–001.*

Chu, E. (2008). *Discrete and Continuous Fourier Transforms: Analysis, Applications and Fast Algorithms.* CRC Press.

Cooley, J. W. & Tukey, J. W. (1965). An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, *19*, 297–301.

Cowpertwait, S. P. P. & Metcalfe, V. A. *Introductory Time Series with R.* Springer.

Heideman, M. T., Johnson, D. H., & Burrus, C. S. (1985). Gauss and the history of the fast fourier transform. *Archive for History of Exact Sciences*, *34*(3), 265–267.

Herrero, E., Morales, J. C., Ribas, I., & Naves, R. (2011). Wasp–33: The first δ scuti exoplanet host star. *Astronomy and Astrophysics Letters*, *526*(L10), 4.

Hipel, K. W. & McLeod, A. I. (1994). *Time Series Modelling of Water Resources and Environmental Systems*. Elservier.

Horne, K. (2012). As5001: Advanced (astronomical) data analysis. Lecture Notes.

Kondratiev, V. I., McLaughlin, M. A., Lorimer, D. R., Burgay, M., Possenti, A., Turolla, R., Popov, S. B., & Zane, S. (2009). New limits on radio emission from x–ray dim isolated neutron stars. *The Astrophysical Journal*, *702*, 692–706.

Lomb, N. R. (1976). Least-squares frequency analysis of unequally spaced data. *Astrophysics and Space Science*, *39*, 447–462.

Marsaglia, G. & Bray, T. A. (1964). A convenient method for generating normal variables. *Society for Industrial and Applied Mathematics Review*, *6*(3), 260–264.

Mayor, M. & Queloz, D. (1995). A jupiter–mass companion to a solar–type star. *Nature*, *378*, 355–359.

O'Donovan, F. T., Charbonneau, D., Mandushev, G., Dunham, E. W., Latham, D. W., Torres, G., Sozzetti, A., Brown, T. M., Trauger, J. T., Belmonte, J. A., Rabus, M., Almenara, J. M., Alonso, R., Deeg, H. J., Esquerdo, G. A., Falco, E. E., Hillenbrand, L. A., Roussanova, A., Stefanik, R. P., & Winn, J. N. (2006). Tres–2: The first transiting planet in the kepler field. *The Astrophysical Journal*, *651*(1), L61–L64.

Percy, J. R. (2007). *Understanding Variable Stars*. Cambridge University Press.

Perryman, M. (2011). *The Exoplanet Handbook*. Cambridge University Press.

Pigulski, A., Pojmański, G., Pilecki, B., & Szczygieł, D. M. (2009). The all sky automated survey. the catalog of variable stars in the kepler field of view. *Acta Astronomica*, *59*(1), 33–46.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). *Numerical Recipies: The Art of Scientific Computing*. Cambridge University Press.

Sabin, W. E. (2008). *Discrete-Signal Analysis and Design*. Wiley–Interscience.

Scargle, J. D. (1982). Studies in astronomical time series analysis. ii. statistical aspects of spectral analysis of unevenly spaced data. *The Astrophysical Journal*, *263*, 835–853.

Shporer, A., Jenkins, J. M., Rowe, J. F., Sanderfer, D. T., Seader, S. E., Smith, J. C., Still, M. D., Thompson, S. E., Twicken, J. D., & Welsh, W. F. (2011). Detection of koi–13.01 using the photometric orbit. *The Astronomical Journal*, *142*(6), 7.

Shu, H. F. (1982). *Physical Universe: An Introduction to Astronomy*. University Science Books.

Spiegel, M. R. (1975). *Shaum's Outline of Probability and Statistics*. McGraw–Hill.

Staelin, D. H. (1969). Fast folding algorithm for detection of periodic pulse trains. *Proceedings of the IEEE*, *57*(4), 724–725.

Thomson, S. E., Christiansen, J. L., Jenkins, J. M., Caldwell, D. A., Barclay, T., Bryson, S. T., Burke, C. J., Clarke, B. D., Girouard, F., Haas, M. R., Hall, J. R., Ibrahim, K., Klaus, T. C., Kolodziejczak, J. J., Li, J., & McCauliff, S. D. (2013). Kepler data release 19 notes. *KSCI–19059–001*.

Wall, J. V. & Jenkins, C. R. (2003). *Practical Statistics for Astronomers*. Cambridge University Press.

Warner, B. D. (2006). *Lightcurve Photometry and Analysis*. Springer.

Welsh, W. F., Orosz, J. A., Seager, S., Fortney, J. J., Jenkins, J., Rowe, J. F., Koch, D., & Borucki, W. J. (2010). The discovery of ellipsoidal variations in the kepler light curve of hat–p–7. *The Astrophysical Journal Letters*, *713*(2), L145–L149.

Zakai, A. (2009). The rise of modern science and the decline of theology as the 'queen of sciences' in the early modern era. *Renaissance and Reformation Review*, *9*(2), 125–151.

This page intentionally left blank