Model Calculations of Higher Order Spectra of Pulsar Emission

Bachelor Thesis

in the
Degree Program
„Bachelor of Science“
in Physics

at the Faculty of Physics and Astronomy
of the Ruhr-Universität Bochum

by
Leonard Kosziol

from
Dortmund

Bochum 2021

Contents

Abbreviations

ATNF Australian Telescope National Facility.

CPU Central processing unit.

DFT Discrete Fourier Transform.

DM Dispersion measure.

FFT Fast Fourier Transform.

FFTW Fastest Fourier Transform in the West.

GPU Graphics processing unit.

ISM Interstellar medium.

PC Personal computer.

PDF Probability density function.

RAM Random access memory.

RFI Radio frequency interference.

VRAM Video random access memory.

Abstract

From a simple model of pulsar radio emission, it is investigated if different signatures can be seen in the higher-order spectra of the time series up to order four of coherent radiation and of random, incoherent noise. This is a first step to examine higher-order spectra as a tool to study the supposed coherent radio emission of pulsars, thereby providing an aid for model development of the pulsar emission mechanism, which is not well understood. Methods for the calculation of the spectra from their estimators are also presented. The models for the radiation are based on simple assumptions and on observational facts where practical, but include no propagation effects.

Visual inspection of the obtained spectra, especially the trispectra (order 4), suggests that there are minor, but visible differences following from the type of radiation which get stronger when the signal becomes more coherent and the discrete Fourier transform (DFT) resolves the indivdual pulses better. However, it is unclear if these are strong enough to study real pulsar signals which are subject to high levels of background noise. Also, it became clear that the state of the art-estimators used here are not well suited to examine pulsar signals, because they require a short coherence time scale compared to the frames of the DFT used for the processing of the signal. Maybe more general estimators have to be derived to study pulsars properly.

Lastly, an artefact of one estimator or its implementation has been found that could be related to the pulse period of the signal. This provides an opportunity for more investigation, driven by the hope to derive the pulse period or even the dispersion measure of a pulsar from an estimator of a fourth-order spectrum alone.

Zusammenfassung

Anhand eines einfachen Modells für die Radio-Emission von Pulsaren wird untersucht, ob die Spektren höherer Ordnung der Zeitreihen (bis zu vierter Ordnung) für kohärente Strahlung und zufälliges, inkohärentes Rauschen verschiedene Signaturen zeigen. Dies ist ein erster Schritt, um zu prüfen, ob Spektren höherer Ordnung als Werkzeug zur Untersuchung der vermuteten kohärenten Radioemission von Pulsaren dienen können. Damit könnten sie für die Modellentwicklung des Emissionsmechanismus von Pulsaren hilfreich sein, der nicht gut verstanden ist. Es werden auch Methoden für die Berechnung der Spektren aus ihren Schätzern vorgestellt. Die Modelle für die Strahlung basieren auf einfachen Annahmen und auf Beobachtungs-Fakten, soweit wie es zweckmäßig ist, berücksichtigen aber keine Propagationseffekte.

Eine visuelle Überprüfung der berechneten Spektren, insbesondere der Trispektren (Ordnung vier), legt nahe, dass es kleine, aber sichtbare Abweichungen abhängig vom Typ der Strahlung gibt. Diese werden stärker bei kohärenteren Signalen und wenn die diskrete Fourier-Transformation (DFT) die Pulse besser auflöst. Allerdings ist es unklar, ob die Abweichungen stark genug sind, um echte Pulsar-Signale zu untersuchen, die starkem Hintergrundrauschen ausgesetzt sind. Dafür wurde klar, dass die hier verwendeten und dem Stand der Technik entsprechenden Schätzer nicht gut geeignet sind, um Pulsar-Signale zu analysieren. Das liegt daran, dass sie verglichen mit der Fensterlänge der für die Signalverarbeitung verwendeten DFT eine kurze Kohärenzzeitskala benötigen. Vielleicht müssen allgemeiner anwendbare Schätzer hergeleitet werden, um Pulsare ordnungsgemäß zu untersuchen.

Schließlich wurde noch ein Artefakt eines Schätzers oder seiner Implementation gefunden, der mit der Puls-Periode des Signals zusammenhängen könnte. Dies bietet eine zusätzliche Möglichkeit, weiterzuforschen, motiviert von der Hoffnung, die Periode eines Pulsars oder sogar sein Dispersionsmaß allein aus einem Schätzer für ein Spektrum vierter Ordnung abzuleiten.

1 Of Pulsars and Polyspectra

1.1 What are pulsars?

Pulsars are rapidly rotating, magnetized neutron stars that show a periodic emission of usually radio waves, sometimes high-energy photons and rarely light in the optical regime [18]. They have originally been discovered as pulsating radio sources in 1968, and hence named "pulsar" (for *pulsating radio source*) [18].

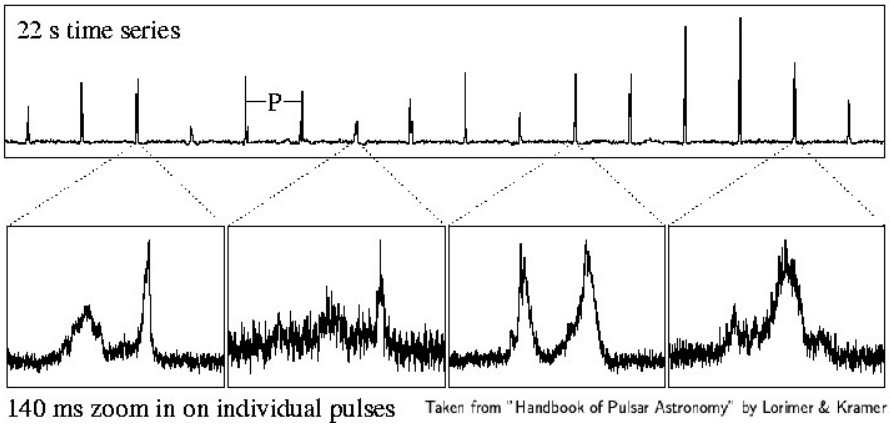


Figure 1: Relatively rare example of pulses that are individually visible from [18]. It is obvious that these pulses from PSR B0301+19 change their shape from pulse to pulse (though not totally), which is usual for pulsars [18], [33]. Recorded with the Arecibo Telescope [18]. The ordinate is an axis of electrical power measured by a pulsar-backend of the telescope.

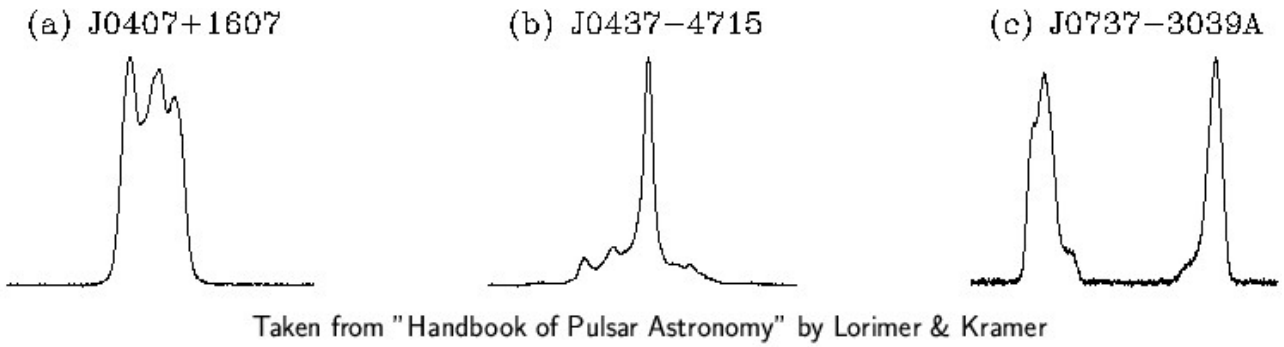


Figure 2: Three examples of so called "integrated pulse profiles" from [18]. These profiles are obtained by adding the received power of the signal of many individual pulses [18], [33]. Profile c) shows two distinct pulses within one period.

In the general picture, pulsars respectively neutron stars are created from the cores of heavy stars during core-collapse supernovae [18]. Being composed mainly of ultra-dense neutron-matter, they range in mass between white dwarfs and stellar black holes (meaning from roughly 1 to 2 solar masses [18]), but are only on the order of 10 km in diameter [18], which is only about 3 times bigger than the Schwarzschild-Radius of the corresponding mass[18].

During the collapse, the magnetic flux of the magnetic field of the star is conserved at the location of the neutron star, because the neutron matter is thought to be superconducting [18]. That means that the magnetic field lines get compressed along with the matter that forms the neutron star, which leads to very high magnetic field strengths on the order of 10^3 T for typical pulsars [18]. Similarly, a good part of the angular momentum of the whole star is transferred to the neutron star. Because the latter

has a much smaller radius than any star during its period of nuclear fusion, the neutron star can spin very rapidly, with rotation periods ranging from more than 20 seconds to less than two milliseconds [23], [19].

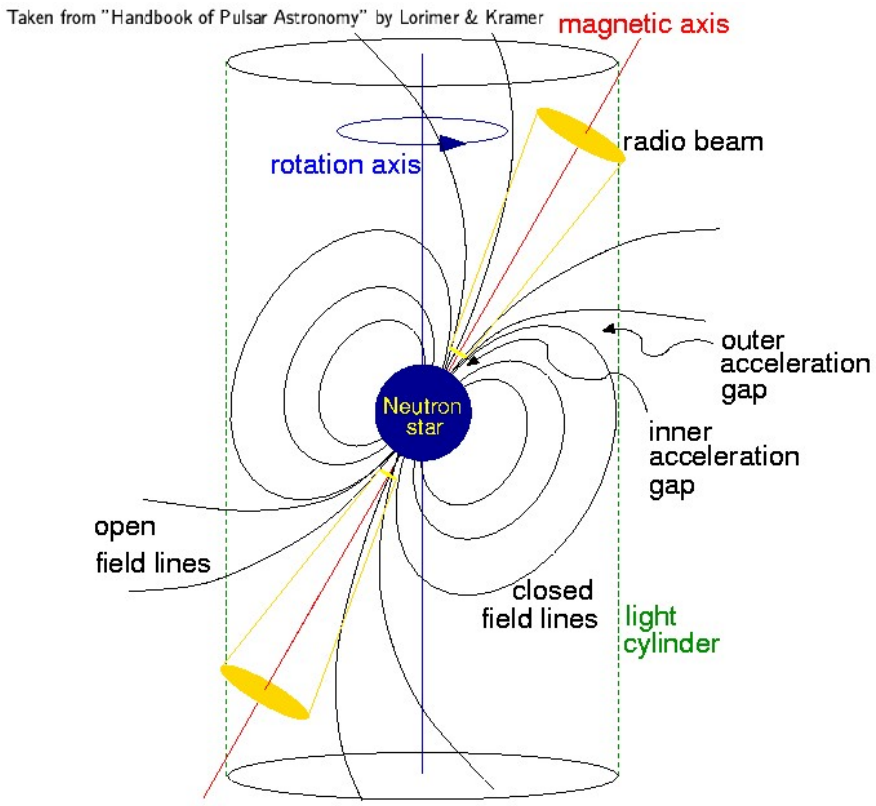


Figure 3: Illustration of a toy model from [18] showing the probable components of a pulsar that make it shine periodically in the radio, especially the geometry of the magnetic dipole field and the emission cone.

The combination of high magnetization and rapid rotation are the key factors that lead to periodic radio pulses.
From the existence of strong magnetic fields, it can be concluded that there has to be an "atmosphere" of charged particles around the pulsar, which are accelerated somehow and emit across the radio spectrum [18].
The rotation then shapes the magnetic field (which is thought to be roughly a dipole) in a way that all field lines below a certain curvature, depending on the rotation period of the pulsar, cannot be closed anymore because then their rotation speed would exceed the speed of light [18].

This leads to regions with open field lines around the magnetic poles of the pulsar, which are thought to be a prerequisite for the creation and escape of the intense radio emission that is observed from pulsars [18].

Therefore, the pulsed emission can be explained in a way that the pulsar permanently radiates in the radio regime, but its axis of rotation is not aligned with that of the magnetic dipole field [18]. As the direction of emission is determined by the magnetic field, the "emission cone" of the pulsar rotates around the rotation axis [18]. During the part of the rotation period where it intersects with the line of sight, radio emission can be observed, otherwise, the pulsar is invisible [18]. This is known as the "lighthouse effect" [18].

As of now, more than 3000 pulsars have been discovered, as can be inferred from the pulsar catalogue of the Australian Telescope National Facility [19], [23], almost all of which are situated within the milky way galaxy [8].

Figure 4: Combined image of the Crab nebula from X-ray- (blue, white), optical (purple) and infrared-observations (pink) to illustrate how the environment of a young pulsar might look [22]. The Crab nebula was produced by a supernova observed in 1054 AD [18]. In the center hides PSR B0531+21, the "Crab pulsar" with a period of 33 ms [18].

1.2 The pulsar signal

Aside from the toy-model described in section 1.1 and explanations for some of the observational characteristics of pulsars, it is not well understood how the radio emission is actually produced [18], [20], [33]. Although there are several types of models [20], [21], none of them has so far been a good enough explanation for the joint behaviour of the pulsar population [18], [20], which is demanded from a satisfying model, because many pulsars actually behave rather similarly across their population [33].

That said, there are clues that not any mechanism which generates radio waves is possible for pulsars. For example, it is generally thought that pulsar radiation must be coherent meaning groups of particles radiate together in phase [33]. This is because the so-called "brightness temperature" T_{B}, which is the temperature of a radio source of given intensity calculated with the Raleigh-Jeans-law (a standard radioastronomical measure and, as the intensity, dependent on the apparent size of the source), is way too high too be generated by an incoherent process [33]. This means that an incoherent process could only generate far less energy than observed within the supposedly tiny spacial extent of a neutron star respectively its region of open field lines.
Furthermore, it is an observational fact that pulsar radio emission is also highly polarized [18], which needs to be included in the models as well. But this polarization can look very different for different pulsars [18]. To not complicate this work unneccesarily, it will not be further considered in the model.

After the radiation is produced and has left the immediate vicinity of the pulsar, it interacts with the interstellar medium (ISM for short), and, depending on the constitution of the ISM between pulsar and observer, several distinct effects change the characteristic of the signal. “ISM” is a term for many kinds of gases and plasmas within a galaxy, and large parts of it are magnetized and ionized [18], which means that electromagnetic waves can interact with it. One of the most pronounced phenomena is a delay proportional to the inverse of the frequency squared, which is known as dispersion [18], [33]. This is due to an interaction with free electrons part of a proton-electron plasma and changes very slowly

with time, so the whole process can often be well described only by the integrated column density of the electrons along the line of sight, the "dispersion measure" or DM [18]. To observe and also to find pulsars within a survey, the signal has to be processed reversing the frequency-dependent delay, which is known as de-dispersion [18]. As this is a standard process in pulsar astronomy, this work assumes that the modeled signals are already de-dispersed to perfection.

Apart from the rather well-behaved dispersion, there are two effects that arise because the ISM is inhomogenous and also turbulent [18]. One is scintilliation, which happens when the signal is subject to phase modulations, and results in changes of the overall signal amplitude even during an observation [18]. The other is pulse scattering, which can lengthen the path to the observer resulting in (frequency dependent) delays of a part of the signal power [18]. This expresses itself in a broadening of the pulse profiles to one side.

After having successfully reached the earth, the signal from the pulsar can be detected by a radio telescope, the output of which is subject to two additional effects. One is noise from the receiver electronics (and possibly the sky surrounding the pulsar), and the other is radio frequency interference (RFI), which in radio astronomy means that the observation is contaminated by technical signals. Both of them are neglected in this work, but due to different reasons. RFI is omitted because the main reason it is studied in radio astronomy is to avoid it or remove it from existing observational data. Receiver noise, however, is only neglected because of the preliminary nature of this work. To further pursue the topic and find out if what is tried here *in principle* works *in practice* or even to compare a real pulsar signal with a simulated one to infer some of its properties, one would definitely have to include it.

1.3 Polyspectra in the pulsar context

The problem with the coherent emission in pulsars astrophysics, as pointed out in [21], that no good measure of phase coherence was employed in astrophysics until around 2017, which has probably not changed yet. At this point, polyspectra (or higher-order spectra) enter the arena: they can be thought of as n-th order (intensity) correlators of the Fourier-coefficients of a pulsar signal $\mathbf{x}(t)$ [30], and if the observed signal resolves coherent dynamics of the radiation, there is a chance that these could leave distinct traces in the higher-order spectra, although they may be not directly sensitive to phase coherence [26].

Moreover, they naturally provide information about correlation across a frequency band. This might also be helpful for model development in pulsar astrophysics, because it provides hints about how much or how little the processes producing different frequencies interact with each other, e.g. if different frequencies are produced at different distances away from the pulsar surface, which is an idea that several models pick up on [18].

Stepping back to the mathematical foundations of higher-order spectra, the signal is treated as a stochastic vector $\vec{y}$ being filled with events from stationary stochastic processes [26]. Later, it will be a set of finite time series of frequency-channels in the DFT-matrix, but now now it is regarded as infinitely long for the sake of theory.

A generalized way to define n-th order spectra $S^{(n)}$ given in [5] is then

$$S_z^{(n)}(\omega_1, ..., \omega_n) 2\pi\delta(\omega_1 + ... + \omega_n) = C_n(z(\omega_1), ..., z(\omega_n)), \tag{1}$$

which means that $S^{(n)}$ evaluated for sets of argument frequencies that sum up to zero is given by the so-called n-th order cumulant C_n of the ideal, infinitely-long time series $z_{(\omega_i)}$ of each of these frequencies ω_i. These cumulants are similar to moments in the way they are generated, but are defined

using another function, the "cumulant-generating function", which is the natural logarithm of the moment-generating function:

$$K_{\vec{y}}(\vec{k}) \;=\; \ln\langle\exp(\vec{k}\cdot\vec{y})\rangle \tag{2}$$

The vector $\vec{y}$ is, in this work, essentially a set of n time series of Fourier-coefficients from the DFT, and $\vec{k}$ is introduced to generate the cumulants by taking the derivative with respect to each component of $\vec{k}$ at $\vec{k}=\vec{0}$ [15], which means that the cumulant only depends on $\vec{y}$.

The neat thing about this derivation is that for stochastically independent vectors $\vec{y}_1$ and $\vec{y}_2$, the cumulant is linear because of the properties of the logarithm [15] (it is even multi-linear, [26]). This means that spectra based on cumulants cleanly separate the contributions of various signals that are added to each other and measured together, which is not so for spectra based on moments [15].

In addition, higher-order spectra have the property that the contribution of Gaussian noise vanishes [6]. This means in actual observations, the effects of Gaussian noise will be reduced to a small noise contribution around zero and the contribution of pulsar signals will become clearer as the observation continues (private communictaion with Daniel Hägele), which is one of the essential motivations to employ them for the analysis of pulsar signals, the pulses of which are usually drowned in noise [18].

The first cumulant based spectrum employed in this work is an alternative to the traditional power spectrum, defined as the second-order spectrum of a frequency and its complex conjugate, $S^{(2)}(\omega)\propto C_2(a_\omega,a_\omega^*)$, which is also valid for time-series that are not average-free and is therefore more general than the traditional version [15].

It is important to know that this formulation fulfills the definition in eq. 1 because of the symmetry of the Fourier-transform: the above definitions assume that the Fourier-transform is continuous and therefore includes coefficients for all positive and negative frequencies, so that one can pick frequencies for the spectra in a way that their sum equals zero. The Fourier-transform of a purely real signal is, however, symmetric around 0 [9] in a way that $a_\omega = a_{-\omega}^*$, with * denoting complex conjugation. This means that for real signals, one can also define the spectra (also e.g. to third and fourth-order) with the complex conjugate of the Fourier-coefficient of ω as a replacement for the Fourier-coefficient of $-\omega$.

The third-order spectrum $S^{(3)}(\omega_1,\omega_2)$ is called the "bispectrum" and is a correlator of the Fourier-coefficients a_{ω_1}, a_{ω_2} and $a^*_{\omega_1+\omega_2}$, which is also sensitive to time-inversion [30].

If constructed analogously to $S^{(3)}$, the trispectrum $S^{(4)}$ would depend on three independent frequencies [30], which would create a rather unhandy 3D-matrix if calculated for the whole frequency band. Instead, this work will employ a simpler alternative in the form of the correlation spectrum $S^{(4)}(\omega_1,\omega_2)$ that can be seen as a frequency-dependent intensity-intensity correlation [30] and equates to a two-dimensional cut through the three-dimensional $S^{(4)}(\omega_1,\omega_2,\omega_3)$.

This has already been handled the same way in [15], [26], [30] and [31], calculating $S^{(4)}(\omega_1,\omega_2)$ with $C_4(a_{\omega_1},a^*_{\omega_1},a_{\omega_2},a^*_{\omega_2})$ to fulfill definition 1.

For all these spectra, it must be clearly stated that in case of experimental discretized finite time series, the cumulants are not calculated directly, but via "estimators", which are similar, but modified expressions [30] that correct for the finiteness of the time series. There are often many variants of estimators valid for different types of signals [27], which are introduced to either save computational cost or to improve the statistical properties (e.g. the variance) of the results.

The specific estimators used in this work are taken from [27] and are discussed later.

1.4 Modeling the pulsar signal

While there are several questions that could be addressed in astronomy with the help of higher-order spectra, especially radio astronomy and also pulsar astronomy, the first one that will be attempted to answer is
"Do higher-order spectra of simple simulated pulsar-like radio signals look different if the pulsar radiates coherently compared to when it does not?".

This simple question allows to be answered for some very idealized cases first, simply to be able to control the properties of the pulsed emission and then infer to which contributions they lead in the higher-order spectra. This would not be possible, at least not directly, if one calculated higher-order spectra of real pulsar observations to start with, especially in the case of the type of radiation that constitutes the pulsar signal.

In general, the signal modeling the pulsar emission $\mathbf{x}(t)$ is generated by multiplying a time series $\mathbf{p}(t)$ containing regularly spaced pulses (hereafter called the "pulse filter") with one that contains a type of underlying signal that the pulsar emits:

$$\mathbf{x}(t) \;=\; \mathbf{p}(t) \cdot \mathbf{g}(t) \text{ or } \mathbf{x}(t) \;=\; \mathbf{p}(t) \cdot \mathbf{s}(t), \tag{3}$$

where $\mathbf{g}(t)$ stands for normally distributed random noise and $\mathbf{s}(t)$ means some kind of coherent radiation.

The shape of the pulses within the pulse filter is time-independent, and because of the pulse filter being multiplied to the time series, the pulse shape will also not be frequency-dependent. This is different from the behaviour of many pulsars, where only the pulse shape averaged over many periods (the integrated pulse profile) is stable, while the individual shape changes from pulse to pulse [18]. In addition, this integrated pulse profile changes with frequency [18], making it clear that this kind of pulse filter is clearly a simplification.

The properties of the pulse filter also implicitly exclude the propagation effects in the ISM mentioned in section 1.2 in addition to various kinds of pulsar dynamics aside from perfectly regularly spaced pulses, of which there are many [18].

For modeling the underlying radiation $\mathbf{g}(t)$ respectively $\mathbf{s}(t)$, one has few indications, or more positively formulated, some freedom, because not much is known about it and its degree of coherency [20]. Accordingly, radiation was modeled in a way so that it would not strongly contradict the observational knowledge, and the unknowns where filled with slightly arbitrary, but simple assumptions.

For the coherent radiation, this for example means that the power spectrum had to look at least roughly like that of Gaussian noise, and that no obvious differences should be visible from the time series alone. As for the simple assumptions, the coherence time is always the same, meaning that the phase jumps occur strictly periodically, which is also simple to calculate. Rather arbitrarily, the coherence time also exactly matches one pulse period for most of the signals analysed here.

The only observation that was violated knowingly was that pulsars usually have steep power spectra scaling with $S_{\text{mean}}(f) \propto f^{\gamma}$ with $0 \gtrsim \gamma \gtrsim -4$ [18] and a mean $\gamma_{\text{mean}} = -1.4 \pm 1$ for "normal", non-millisecond pulsars [2]. But that was only done to keep the model simpler and the results easier to analyse. That said, there are also pulsars with a nearly flat power spectrum [18], [2].

It must also be noted that all pulsar signals in this work consist of way lower radio frequencies than one would obtain if the pulsar signal would be sampled directly, e.g. with the real-time solution for obtaining correlation spectra developed in [26]. The latter would be somewhere between 100 MHz to 100 GHz [18], whereas here, the frequency bands usually end at under 1 MHz or even less.
This is mainly necessitated by the limited computational resources available for this work (standard consumer PCs with components that are far below the state of the art), but may be acceptable in

principle because in radio astronomy, the signal is anyway mixed down to baseband before further analysis [18], [33].

1.5 Outline of this work

To put into practice what has been introduced in the previous sections, the most essential task was to realize a proper signal-analysis pipeline for the simulated pulsar signals with the higher-order spectra at its end in the Julia programming language [4]. This will be discussed in detail in the following chapter 2 "Calculating polyspectra in Julia", starting with the definition of the DFT and specific implementation of a routine to calculate it efficiently. It continues with a short presentation of the estimators for the power spectrum, the bispectrum and the trispectrum and ways to calculate the spectra from them.

A short chapter 3 on "Pulsed Gaussian Noise in Julia" then picks up on random number generation and explains the technicalities of the pulse filter in more detail.

After that, the model of broadband coherent radiation is presented in 4 "Coherent Radiation in Julia", along with two possible algorithms to obtain coherent radiation with strictly periodic phase jumps. This formally includes unrealistic, but instructive examples like coherent radiation consisting of only one sine wave, which is treated in sections 5.7 and 5.8.

Finally, the higher-order spectra for each one signal consisting of pulsed Gaussian noise and another one from coherent radiation are shown and compared in the chapter 5 "Comparison of Higher-Order Spectra" for different settings regarding the DFT, the higher-order spectra themselves and the pulse filter. These comprise the results of this work and are summed up as well as discussed in the very last chapter 6 "Discussion and outlook".

Beyond the scope of this work, it may also be possible to use higher-order spectra from cumulants in other branches of pulsar astronomy, for example to study pulsar timing data, which is in turn used to study i.e. general relativity in orbital systems with at least one pulsar, planetary dynamics in the solar system and a supposed background of stochastic gravitational waves [18]. There, one also has to deal with correlated noise phenomena and account for them to take precise measurements [7], [18], [28].

Additionally, it could be possible to use higher-order spectra of time series detected by radio telescopes (or outputs of filterbank-like devices) for the discovery of new pulsars during surveys. Here, not only suppression of Gaussian noise by higher-order spectra would be advantageous, but also the experience of the authors of [15] that technical signals leave distinct signatures in the spectra [15], making them helpful for RFI-removal (pointed out by Joris Verbiest from the University of Bielefeld in private communication with Daniel Hägele).

2 Calculating Polyspectra in Julia

2.1 DFT routine

For the calculation of the spectra, a signal first constructed in the time-domain must be expressed as a set of (lower resolution) time-series of different frequencies. Here, discrete Fourier transforms (**DFTs**) will be used to construct such a set with "channels" z_{ω_i} of equal distance in frequency space. The definition of the discrete Fourier transform is chosen in accordance to [30] as

$$a_k^{(n)} = \frac{T}{N} \sum_{j=0}^{N-1} g_j z_j^{(n)} * e^{2\pi i j k / N}, \tag{4}$$

where T denotes the duration in time of one frame with N array-elements, j is the index of time in the frame, k is the index of frequencies in the resulting Fourier transform, $z_j^{(n)}$ is the value of the time-domain signal at position j, i is the imaginary unit and g_j is the value of the window function at j, which is applied to each frame. Please note that $z_j^{(n)}$ is now an element o fthe time series and not a a series of Fourier coefficient like before.

For practical reasons, this definition emphasizes the difference between the original discretized signal $z_j^{(n)}$ and the window function, although the DFT acts on both of them.

A window function is multiplied to one frame (or window) with N coefficients $z_j^{(n)}$, dampening the values near the borders while retaining the ones in the middle. This is done in practice mainly to reduce spectral leakage, which means that waves which are not exactly periodic within the window (meaning the frequency does not exactly match one that DFT "tests for") make contributions not only to their nearest frequency that the DFT accounts for, but also to all others [14]. This is because these waves will have discontinuities at the window boundaries, the effects of which can be reduced by smoothly bringing the values towards zero at the boundaries [14], which many window-functions accomplish [14].

In contrast to [30], the DFT package `FFTW.jl`, which is used in this work to calculate Fourier transforms, implies a definition like

$$a_k^{(n)}{}_{\mathrm{FFTW}} \;=\; \sum_{j=1}^{N} g_j z_j^{(n)} * e^{-2\pi i j k/N} \quad [12] \tag{5}$$

assuming that the window function g is already multiplied with the signal array. Please note that j would run from $j = 1$ to N in Julia as arrays are indexed from 1 on in this language [17]. Therefore, all Fourier coefficients obtained as a direct output of the functions within `FFTW.jl` must be modified as

$$a_k^{(n)} \;=\; \frac{T}{N} \cdot \left(a_k^{(n)}{}_{\mathrm{FFTW}}\right)^*, \tag{6}$$

where * denotes complex conjugation. Although the additional step of complex conjugation could be avoided by using an inverse (or backward) FFT-function, it is better to take it because it allows one to use the FFT-routine for real numbers, which leads to an approximate improvement of a factor of 2 in computation speed and memory usage [12]. For arrays with purely real entries, certain steps of the FFT-algorithms can be left out while retaining the full information of the time-domain signal because the DFT Y shows the Hermitian symmetry

$$Y[N - j] = Y^*[j] \quad [12]. \tag{7}$$

The routine for generating a Fourier transform with a frequency resolution determined by the window-length N out of a time-domain signal $\mathbf{x}(t)$ with n_x array entries and time duration of a single array entry δt is as follows:

1. Calculate $T = \delta t \cdot N$.

2. Calculate the number of DFT-frames M within $\mathbf{x}(t)$ with length N by truncating the expression n_x/N.

3. Prepare a column-vector $\mathbf{g}$ with lenght N and fill it with values of the discretized window function.

4. Remove the last $n_\mathrm{x} - M \cdot N$ elements from $\mathbf{x}(t)$, leaving $M \cdot N$ elements in $\mathbf{x}(t)$.

5. Use the `reshape()` command available for Julia-arrays to transform the shortened $\mathbf{x}(t)$ into an $N \times M$ matrix.

6. Multiply $\mathbf{g}$ elementwise with the reshaped signal $\mathbf{x}(t)$ by using Julia's broadcasting-syntax.

7. Perform a real-to-complex DFT (with `FFTW.rfft()`) along dimension 1 (along each column) of this two-dimensional matrix and name the result as $\mathbf{D}_{\mathrm{rfft}}$.

8. Complex-conjugate $\mathbf{D}_{\mathrm{rfft}}$ and multiply it with the scalar T/N.

Although the above process is specific for the use of the CPU, it can be rewritten for GPU-Arrays just by exchanging the `FFTW.rfft()`-command with its equivalent from a GPU-centered library, because `reshape()` is available also for GPU-arrays in Julia.

For signals with $n_{\mathrm{x}} \approx 5 \cdot 10^9$ entries of double precision floating point-numbers and short window sizes N in the order of $N \approx 1000$, this code would run in minutes on the machines used for this work. Therefore, it was not necessary to further speed up the computations by moving them to the GPU, which may be beneficial in case of substantially longer window-sizes.

The main key to achieve this reasonable performance was to compute the DFT on one two-dimensional array which has been formed by the `reshape()`-function instead of either constructing this array within a loop or calculating the FFTs of many one-dimensional arrays with one. Additionally, it should be noted that `FFTW`, which is a C-based library accessible through the `FFTW.jl`-package in Julia, has the means to perform a DFT within $a \cdot N\log_2(N)$-time complexity for every time-domain frame length N and $a \in \mathbb{R}$, because it automatically assembles a suitably fast algorithm based on the properties of the input array [12]. Therefore, it is not imperative to restrict N to powers of 2 like it is needed for some basic DFT-algorithms [9].

2.2 Approximate confined Gaussian window

The window function used exclusively in this work is the "Approximate Confined Gaussian Window" from [32], or for short **ACGW**, with a curvature parameter s of $s = 0.15$.

This window is a discretized approximation to the family of "Confined Gaussian windows" (**CGW**) [32] and can be calculated directly via the expression (taken from [26]):

$$g(j) \;=\; f(j) - f(-0.5) \cdot \frac{f(j+N+1) + f(j-N-1)}{f(0.5+N) + f(-1.5-N)}, \tag{8}$$

in which f is defined as

$$f(x) \;=\; \exp\left(-\frac{x - N/2}{2(N+1)s}\right)^2. \tag{9}$$

where N stands for the number of elements and j denotes the current index of the array representing the window.

The confined Gaussian window family interpolates between the Cosine-window and high-performance windows (like the Blackman-, de la Valle Poussin-, Bohmann-, BlackmanHarris-, Nuttall- and the Blackman-Nuttall-window), exhibiting the optimal time-bandwith product for an N-point Fourier transform [32]. For a given time resolution of a signal, this means that they "provide the best possible frequency localization that can be obtained in the context of finite discrete Fourier transforms for a given temporal resolution" [32], which makes them a natural choice when obtaining Fourier transforms of signals with fixed temporal resolution like it is done here.

2.3 Implementation of S^2

The power spectrum is defined after the conventions in [30] and [26] as

$$S_z^{(2)}(\omega_k) \;\approx\; \frac{NC_2(a_k, a_k^*)}{T\sum_{j=0}^{N-1}|g_j|^2} \tag{10}$$

with the angular frequency $\omega_k = 2\pi k/T$ and is calculated using the estimator

$$c_2^{(a)}(x,y) \;=\; \frac{m}{m-1}\left(\overline{xy} \;-\; \overline{x}\;\overline{y}\right) \tag{11}$$

from [27] for the second-order cumulant $C_2^{(\mathrm{a})}(x,y)$ by inserting the fourier-coefficient a_ω for x and its complex conjugate a_ω^* for y. Note that the labeling of the fourier-coefficients has changed to angular frequencies and that m indicates the number of elements in the set DFT-frames that $S_{\mathrm{z}}^{(2)}(\omega_k)$ will be calculated for (e.g. M, the number of all DFT-frames), whereas every other character represents the same variable as in subchapter 2.1. Overlines like in $\overline{y(t)}$ denote an average in time over a variable $y(t)$.

Therefore, the main task for a function that calculates the power spectrum lies in the calculation of

$$c_2^{(a)}(a_\omega, a_\omega^*) \;=\; \frac{m}{m-1}\left(\overline{a_\omega a_\omega^*} \;-\; \overline{a_\omega}\;\overline{a_\omega^*}\right) \;=\; \frac{m}{m-1}\left(\overline{|a_\omega|^2} \;-\; \overline{a_\omega}\;\overline{a_\omega^*}\right), \tag{12}$$

which is embedded into the following approach:

1. Calculate prefactor $p_1 = \frac{m}{m(m-1)} = \frac{1}{m-1}$ to save the step of calculating time averages explicitly.

2. For $\overline{a_\omega a_\omega^*}$, calculate the matrix $\mathbf{D}_{\mathrm{abs2}}$ by calculating the absolute square of every element in $\mathbf{D}_{\mathrm{rfft}}$ and then the vector $\mathbf{v}_1 = \texttt{sum(}\mathbf{D}_{abs2}\texttt{, along time-direction)}$ by summing over the rows of $\mathbf{D}_{\mathrm{rfft}}$.

3. Sum over the time-dimension of $\mathbf{D}_{\mathrm{rfft}}$ to obtain a column-vector $\mathbf{d}_{\mathrm{sum}}$ with N_{rfft} elements.

4. For $\frac{1}{m}\,\overline{a_\omega}\overline{a_\omega^*}$, calculate the vector $\mathbf{v}_2 = (1/m)\;.\cdot(\mathbf{d}_{\mathrm{sum}} \;\;.\cdot \mathbf{d}_{\mathrm{sum}}^*)$ via elementwise multiplication indicated by "$.\cdot$".

5. To obtain the final estimators for all angular frequencies perform the elementwise substraction of vectors $\mathbf{v}_{c_2} = p_1\;.\cdot(\mathbf{v}_1\;.-\mathbf{v}_2)$.

6. To obtain the desired spectrum S_Z^2, multiply $\mathbf{v}_{c_2}$ elementwise with $\frac{N}{T\sum_{j=0}^{N-1}|g_j|^2}$.

All operations that have not been discussed further, like transposing, taking absolute squares, broadcasting of elementwise operations via the dot-syntax (e.g. $.\cdot$ or $.-$) and summing over various dimensions are already implemented in Julia without having to use additional libraries.

2.4 Implementation of S^4

The correlation spectrum or trispectrum is defined like in [26] and [30] as

$$S_{\mathrm{z}}^{(4)}(\omega_k, \omega_l) \; =\approx \; \frac{NC_4(a_k, a_k^*, a_l, a_l^*)}{T\sum_{j=0}^{N-1}|g_j|^4} \tag{13}$$

and can be calculated using the estimator $c_4^{(\mathrm{cb})}(a, a^*, b, b^*)$ from [27] when inserting $a = a_k$ and $b = a_l$:

$$c_4^{(\mathrm{cb})}(a, a^*, b, b^*) \; = \; \underbrace{\frac{m+1}{m-1}\overline{aa^*bb^*}}_{\textbf{part 1}} \; - \; \underbrace{\frac{m}{m-1}\overline{aa^*}\,\overline{bb^*}}_{\textbf{part 2}} \; - \; \underbrace{\frac{m}{m-1}\overline{ab^*}\;\overline{a^*b}}_{\textbf{part 3}} \tag{14}$$

This reduced estimator is recommended in [27] for the analysis of time-dependent signals, as it is a simplification of the full unbiased estimator $c_4^{(a)}$ valid only for two variables and their complex-conjugates. Additionally, it is valid on the main diagonal of the correlation spectrum, where $a = b$ [27], However, it assumes that both variables a and b are average-free, meaning the average of their values over the full time of each frequency channel of $\mathbf{D}_{\mathrm{rfft}}$ is zero: $\langle a\rangle = 0$, $\langle b\rangle = 0$, and that the second-order cumulant $C_2(a, b)$ also equals zero.

For pulsed signals with gaussian noise, it has been checked for a few example signals if the above requirements can assumed to be fulfilled by calculating the time-averages of each frequency channel of the DFT-arrays and $C_2(a_k, a_l)$ where k and l denote indices of frequency channels. It was found that the absolute values of the complex time-averages were three to four orders of magnitude smaller than their standard deviation and consistent with zero. The same was true for $C_2(a_k, a_l)$, estimated by $c_2^{(\mathrm{a})}$ as defined in [27]. However, these are far from rigorous proofs for the validity of $c_4^{(\mathrm{cb})}$ for the signals to be analyzed. That can be illustrated by the fact that for pulsed signals with broad-band coherent radiation and coherence lengths two magnitudes larger than the size of the DFT-windows, the above "test" showed no immediately notable difference to pulsed gaussian noise. But $c_4^{(\mathrm{cb})}$ is a mathematically invalid estimator for such signals, as they are “too coherent” (private communication with Daniel Hägele and Markus Sifft).

It is also to be noted that for the estimators of order $n > 2$, “short-time averaging” is usually employed, meaning that the estimator is calculated for many smaller sets of frames ($m << N_{\mathrm{x}}/N$) and the results are averaged successively. On the one hand, this is practical from a computational perspective, as it can demand substantially less memory. On the other hand, it prevents contributions to the higher-order spectra from being averaged away in case of certain signals that e.g. include frequency-drifts (private communication with Daniel Hägele and Fabian Schefczik).

Like in subsection 2.3, vectorized matrix operations can used to calculate $c_4^{(\mathrm{cb})}(a_k, a_k^*, a_l, a_l^*)$ from $\mathbf{D}_{\mathrm{rfft}}$ via the following scheme, which is used almost exclusively throughout this work.

1. Choose which range of frequencies shall be taken into account when calculating S^4 and curtail $\mathbf{D}_{\mathrm{rfft}}$ accordingly, leaving only the rows that one is interested in. Name the number of remaining frequency channels as N_{chan}, which would be equal to $N_{\mathrm{rfft}} = N$ if $\mathbf{D}_{\mathrm{rfft}}$ is not curtailed.

2. Choose an m to specify how may DFT-frames are used for short time averaging.

3. Calculate the prefactors $p_{\mathrm{t1}} = \frac{m+1}{m(m-1)}$ and $p_{\mathrm{t2}} = \frac{m}{m^2(m-1)}$ to save a bit of time compared to calculating the averages over time explicitly.

4. Initialize an $N_{\mathrm{chan}} \times N_{\mathrm{chan}}$ -matrix $\mathbf{S}_{\mathrm{init}}^4$ on which the values of the estimator for all frequency pairs will be added. This can be done with a matrix-construction method provided by Julia like `zeros(data type, rows, columns)` or by calculating the estimator for e.g. the first set of

frames of $\mathbf{D}_{\text{rfft}}$, which yields an array of the appropriate data type.

5. Iterating through adjoining sets of DFT-frames $\mathbf{D}_i^{\text{frames}}$ (consisting of m frames each) within $\mathbf{D}_{\text{rfft}}$, calculate the estimator $c_4^{(cb)}(a_k, a_k^*, a_l, a_l^*)$ as follows. This loop can easily be parallelized when working on the CPU (e.g. with the `Threads@threads`-macro natively implemented into Julia), as the results are the same (at least mathematically) regardless of the order of the iteration.

   a) Calculate the matrix $\mathbf{D}_{\text{abs2}}$ by taking the absolute value squared of every element in $\mathbf{D}_i^{\text{frames}}$.

   b) For **part 1**, calculate the $N_{\text{chan}} \times N_{\text{chan}}$ -matrix $\mathbf{C}_1 = p_{\text{t1}} \mathbin{.\cdot} (\mathbf{D}_{\text{abs2}}\, \mathbf{D}_{\text{abs2}}^T)$ via matrix multiplication.

   c) Calculate the column-vector $\mathbf{d}_{\text{abs2}} =$ `sum(`$\mathbf{D}_{\text{abs2}}$`, along time-direction)` by summing over the rows of $\mathbf{D}_{\text{abs2}}$.

   d) For **part 2**, calculate the $N_{\text{chan}} \times N_{\text{chan}}$ matrix $\mathbf{C}_2 = p_{\text{t2}} \cdot \left(\mathbf{d}_{\text{abs2}} \otimes \mathbf{d}_{\text{abs2}}^T\right)$ via a dyadic product.

   e) Calculate the matrix $\left(\mathbf{D}_i^{\text{frames}}\right)^*$ by taking the elementwise complex conjugate of $\mathbf{D}_i^{\text{frames}}$.

   f) For **part 3**, calculate the $N_{\text{chan}} \times N_{\text{chan}}$ matrix $\mathbf{C}_3 = p_{\text{t2}} \cdot \left(\mathbf{D}_i^{\text{frames}} \left(\left(\mathbf{D}_i^{\text{frames}}\right)^*\right)^T \mathbin{.+} \left(\mathbf{D}_i^{\text{frames}}\right)^* \left(\left(\mathbf{D}_i^{\text{frames}}\right)^*\right)^T\right)$ via matrix multiplication and then elementwise addition and multiplication.

   g) Add the elementwise sum $(\mathbf{C}_1 \mathbin{.-} \mathbf{C}_2 \mathbin{.-} \mathbf{C}_3)$ onto $\mathbf{S}_{\text{init}}^4$. Of course it is also possible to not store $\mathbf{C}_1$, $\mathbf{C}_2$ and $\mathbf{C}_3$ as individual variables, but instead add respectively substract their contents directly to or from $\mathbf{S}_{\text{init}}^4$.

6. Take the real part of $\mathbf{S}_{\text{init}}^4$, divide it elementwise by the number of loop iterations and obtain the correlation spectrum $\mathbf{S}_{\text{z}}^{(4)}$ for all pairs of frequencies through further elementwise division by $\frac{N}{T \sum_{j=0}^{N-1} |g_j|^4}$.

To provide a better understanding why this works respectively how the estimator can be calculated with matrix multiplications, some of above operations explicitly written down in appendix A.

To optimize the performance of this algorithm, one can try to exchange the matrix multiplications for **part 1** in 5b and **part 3** in (5f) with faster implementations, for example on the GPU, or, like tried here, from the package `Octavian.jl` [10], which is based on the package `LoopVectorization.jl`. Because of this dependency, `Octavian.jl` does not support complex numbers, and this means that **part 3** cannot calculated with the same matrix multiplication as **part 1**.

However, it is possible to split up the frames into their real and imaginary parts, and then multiply other matrices leading to the same results. To find these matrices, one first writes the Fourier coefficents with their real and imaginary parts and calculates the two factors of **part 3** (see next page).

$$\langle a_k a_l^* \rangle = \frac{1}{m}\sum_{j=1}^{m}(a_{kj}+ib_{kj})(a_{lj}-ib_{lj}) = \frac{1}{m}\sum_{j=1}^{m}\underbrace{(a_{kj}a_{lj}+b_{kj}b_{lj})}_{\mathbf{Re}} + i\underbrace{(b_{kj}a_{lj}-a_{kj}b_{lj})}_{\mathbf{Im}} \tag{15}$$

$$\langle a_k^* a_l \rangle = \frac{1}{m}\sum_{j=1}^{m}(a_{kj}-ib_{kj})(a_{lj}+ib_{lj}) = \frac{1}{m}\sum_{j=1}^{m}\underbrace{(a_{kj}a_{lj}+b_{kj}b_{lj})}_{\mathbf{Re}} - i\underbrace{(b_{kj}a_{lj}-a_{kj}b_{lj})}_{\mathbf{Im}} \tag{16}$$

$$\tag{17}$$

To obtain $\langle a_k a_l^* \rangle \langle a_k^* a_l \rangle$, one can simply multiply out:

$$\textbf{part 3} = \frac{m}{m-1}\langle a_k a_l^* \rangle \langle a_k^* a_l \rangle = \frac{m}{m^2(m-1)}\left(\sum_{j=1}^{m} a_{kj}a_{lj} + \sum_{j=1}^{m} b_{kj}b_{lj}\right)^2 + \left(\sum_{j=1}^{m} b_{kj}a_{lj} - \sum_{j=1}^{m} a_{kj}b_{lj}\right)^2 \tag{18}$$

The two sums within the brackets are mathematically unneccessary, but are helpful to see where the four matrix multiplications happen. With this, one can rewrite the calculations of **part 3** like this:

Before the loop:

1. Prepare two $N_{\text{chan}} \times m$ matrices $\mathbf{M}_{\text{Re}}$ and $\mathbf{M}_{\text{Im}}$ to store the data from each pair of matrix-multiplications within the brackets in eq. 18.
   This is done because `Octavian.jl` allows to specify pre-allocated arrays to save the results of a matrix-multiplication, which increases the performance.

Within the loop:

1. Take the real and imaginary part of the set of m frames:
   $\mathbf{R}_{\text{frames}} = \text{Re}\left(\mathbf{D}_i^{\text{frames}}\right)$ and $\mathbf{I}_{\text{frames}} = \text{Im}\left(\mathbf{D}_i^{\text{frames}}\right)$.

2. Use the `matmul!()`-function from `Octavian.jl` to perform the matrix multiplications and save the results :
   $\mathbf{M}_{\text{Re}}$ `.+=` $\mathbf{R}_{\text{frames}}\mathbf{R}_{\text{frames}}^T$ `.+` $\mathbf{I}_{\text{frames}}\mathbf{I}_{\text{frames}}^T$
   $\mathbf{M}_{\text{Im}}$ `.+=` $\mathbf{I}_{\text{frames}}\mathbf{R}_{\text{frames}}^T$ `.-` $\mathbf{R}_{\text{frames}}\mathbf{I}_{\text{frames}}^T$

3. Take the squares of every element in $\mathbf{M}_{\text{Re}}$ and $\mathbf{M}_{\text{Im}}$, multiply them with the prefactor and substract the results elementwise from S^4_{init}:
   S^4_{init} `.+=` $p_{\text{t2}} \cdot$ (`abs2.`$(\mathbf{M}_{\text{Re}})$ `.+` `abs2.`$(\mathbf{M}_{\text{Im}})$)

4. Fill $\mathbf{M}_{\text{Re}}$ and $\mathbf{M}_{\text{Im}}$ with zeros.

If one takes additional care to pre-allocate memory for all of the matrices used during the calculation, this implementation could be several times faster than the above one (with its loop parallelized to four threads and running on a quadcore processor), assumed N_{chan} was big enough, that means on the order of 10^3 or more.
However, the Julia-code implementing the above had at least one mistake: the last term in point 2, $\mathbf{R}_{\text{frames}}\mathbf{I}_{\text{frames}}^T$, has been added instead of substracted. This was noticed only very late and could not be corrected or examined more closely. However, this did not even pose a problem, because most of the correlation spectra presented here could be calculated with the earlier implementation except those with $m = 2$, and even for those an artefact occured that is at least interesting or might even be of use in practice (see section 5.9).

2.5 Implementation of S^3

The bispectrum, which shows the correlation between two frequencies and their harmonic sum, is defined in [30] and [26] as

$$S_z^{(3)}(\omega_k, \omega_l, \omega_{k+l}^*) \approx \frac{NC_3(a_k, a_l, a_{k+l}^*)}{T\sum_{j=0}^{N-1}|g_j|^3} . \tag{19}$$

In this work, the general estimator for C_3 will be used, which is referred to as $c_3^{(a)}$ by the terminology from [27]. However, this time the estimator will not be calculated by explicitly multiplying matrices, but instead by specifying the operations in the form of index-notation and Einstein summation. From this, the Julia package `Tullio.jl` will automatically provide a vectorized and multi-threaded routine for the desired operations, if it is paired with the additional package `LoopVectorization.jl` [1]. If `LoopVectorization.jl` is not available, `Tullio.jl` would simply output nested `for`-loops [1], which would in principle not result in a particularly notable performance increase over any `for`-loops written explicitly into the code by hand.

However, `LoopVectorization.jl` does not support the use of complex numbers, which make up the DFT that is the input of any S^3-routine [11], [1]. Nevertheless, it is possible to split up the DFT into its complex and imaginary parts and compute the real and imaginary part of the bispectrum with elements of both of those. To accomplish this, first write $c_3^{(a)}(x, y, z)$ with $x = a_k + ib_k$, $y = a_l + ib_l$, $z = a_{k+l} - ib_{k+l}$, i being the imaginary unit, and the abbreviation $p_3 = \frac{m^2}{(m-1)(m-2)}$:

$$\begin{aligned} c_3^{(a)}(x, y, z) &= \frac{m^2}{(m-1)(m-2)} \cdot \overline{(x - \overline{x})(y - \overline{y})(z - \overline{z})} && (20)\\ &= p_3 \cdot \overline{(a_k + ib_k - \overline{a_k + ib_k})(a_l + ib_l - \overline{a_l + ib_l})(a_{k+l} - ib_{k+l} - \overline{a_{k+l} - ib_{k+l}})} && (21)\\ &= p_3 \cdot \overline{\underbrace{(a_k + ib_k - \overline{a_k} - i\overline{b_k})}_{\textbf{part I}} \underbrace{(a_l + ib_l - \overline{a_l} - i\overline{b_l})}_{\textbf{part II}} \underbrace{(a_{k+l} - ib_{k+l} - \overline{a_{k+l}} + i\overline{b_{k+l}})}_{\textbf{part III}}} && (22) \end{aligned}$$

In order to split $c_3^{(a)}$ into its real and imaginary part, the expression under the longest line indicating a time-average has to multiplied out:

$$\begin{aligned} \mathbf{I} \cdot \mathbf{II} &= a_k a_l + a_k i b_l - a_k \overline{a_l} - a_k i\overline{b_l} + i b_k a_l - b_k b_l - i b_k \overline{a_l} + b_k \overline{b_l} && (23)\\ &\quad - \overline{a_k} a_l - \overline{a_l} i b_l + \overline{a_k}\ \overline{a_l} + \overline{a_k} i \overline{b_l} - i\overline{b_k} a_l + \overline{b_k} b_l + i\overline{b_k}\overline{a_l} - \overline{b_k}\ \overline{b_l} && (24) \end{aligned}$$

From this point on the coefficients can easily be sorted to either the real or imaginary part of the bispectrum. In both cases, part **III** will be split up into real and imaginary parts first:

$$\mathrm{Re}(\mathbf{III}) = a_{k+l} - \overline{a_{k+l}} , \quad \mathrm{Im}(\mathbf{III}) = \overline{b_{k+l}} - b_{k+l} \tag{25}$$

For the real part of the bispectrum, only real coefficients multiplied with Re(**III**) and imaginary coefficients multiplied with Im(**III**) make a contribution. Partly changing the symbolism for the short-term average in time over m frames from $\overline{x(t)}$ to $\langle x(t)\rangle$, one can write:

$$\begin{aligned} \mathrm{Re}\left(c_3^{(a)}\right) &= p_3\langle(a_{k+l} - \overline{a_{k+l}})(a_k a_l - a_k\overline{a_l} - b_k b_l + b_k\overline{b_l} - \overline{a_k} a_l + \overline{a_k}\ \overline{a_l} + \overline{b_k} b_l - \overline{b_k}\ \overline{b_l})\rangle && (26)\\ &\quad + \langle(\overline{b_{k+l}} - b_{k+l})(a_k b_l - a_k\overline{b_l} + b_k a_l - b_k\overline{a_l} - \overline{a_l} i b_l + \overline{a_k}\overline{b_l} + \overline{b_k}\overline{a_l} - \overline{b_k} a_l)\rangle && (27) \end{aligned}$$

This can be further summarized to

$$\begin{aligned}
\mathrm{Re}\left(c_3^{(a)}\right) &= p_3\langle(a_{k+l}-\overline{a_{k+l}})\left(a_k(a_l-\overline{a_l})+b_k(b_l-\overline{b_l})-\overline{a_k}(a_l-\overline{a_l})+\overline{b_k}(b_l-\overline{b_l})\right) && (28)\\
&\quad+(\overline{b_{k+l}}-b_{k+l})\left(a_k(b_l-\overline{b_l})+b_k(a_l-\overline{a_l})-\overline{a_k}(b_l-\overline{b_l})-\overline{b_k}(a_l-\overline{a_l})\right)\rangle && (29)\\
&= p_3\langle(a_{k+l}-\overline{a_{k+l}})\left((a_k-\overline{a_k})(b_l-\overline{b_l})+(b_l-\overline{b_l})(\overline{b_k}-b_l\right) && (30)\\
&\quad+(\overline{b_{k+l}}-b_{k+l})\left((a_k-\overline{a_k})(b_l-\overline{b_l})+(b_k-\overline{b_k})(a_l-\overline{a_l})\right)\rangle && (31)
\end{aligned}$$

For the imaginary part of $c_3^{(a)}$, only the real coefficients from the equation line 23 and 24 multiplied with Im(**III**) and the imaginary coefficients multiplied with Re(**III**) make a contribution. These new coefficients can then also be sorted and summarized, which yields:

$$\begin{aligned}
\mathrm{Im}\left(c_3^{(a)}\right) &= p_3\ \langle(a_{k+l}-\overline{a_{k+l}})\left((a_k-\overline{a_k})(b_l-\overline{b_l})+(b_k-\overline{b_k})(a_l-\overline{a_l})\right) && (32)\\
&\quad+(\overline{b_{k+l}}-b_{k+l})\left((a_k-\overline{a_k})(a_l-\overline{a_l})+(b_k-\overline{b_k})(b_l-\overline{b_l})\right)\rangle && (33)
\end{aligned}$$

These expressions can be translated into matrix expressions (in index-notation) easily, e.g. with matrices that already contain the results of expressions like $(a_k-\overline{a_k})$ pre-prepared.

1. Choose which range of frequencies shall be taken into account when calculating S^3 and curtail $\mathbf{D}_{\mathrm{rfft}}$ accordingly, leaving only the rows that one is interested in. Name the number of remaining frequency channels as N_{chan}, which would be equal to N_{rfft} if $\mathbf{D}_{\mathrm{rfft}}$ is not curtailed.

2. Initialize two matrices $\mathbf{S}^3_{\mathrm{Re}}$ and $\mathbf{S}^3_{\mathrm{Im}}$. Here, one is basically free in specifying their dimensions because `Tullio.jl` does not perform operations that require matching dimensions like element-wise addition or multiplication. In this work, the dimensions of S^3 will be half of those of the corresponding S^4, because S^3 is only defined for frequency pairs ω_k, ω_l with $N_{\mathrm{rfft}} \geq k+l$ and S^3 should be quadratic instead of triangular.

3. Prepare two matrices $\mathbf{A}_{\mathrm{Re}}$ and $\mathbf{B}_{\mathrm{Im}}$ with the same dimensions as $\mathbf{S}^3_{\mathrm{Re}}$ and $\mathbf{S}^3_{\mathrm{Im}}$. These will allow for more flexibility when writing the matrix expressions that `Tullio.jl` evaluates within the loop.

4. Choose an m to specify the number of DFT-frames used for short time averaging.

5. Calculate the prefactor $p_{\mathrm{t3}} = \frac{m}{(m-1)(m-2)}$. Changing the numerator from m^2 to m is due to division by m when averaging the product of parts **I**, **II** and **III** over the set of frames.

6. Iterating through frames $\mathbf{D}_i^{\mathrm{frames}}$ within $\mathbf{D}_{\mathrm{rfft}}$, calculate the estimator $c_3^{(a)}(a_k, a_l, a_{k+l})$ as follows. This loop does generally not need to be parallelized, because `Tullio.jl` already uses multiple threads within one matrix multiplication.

   a) Separate $\mathbf{D}_i^{\mathrm{frames}}$ into $\mathrm{Re}\left(\mathbf{D}_i^{\mathrm{frames}}\right)$ and $\mathrm{Im}\left(\mathbf{D}_i^{\mathrm{frames}}\right)$.

   b) Prepare the vectors
   $\mathbf{a}_{\mathrm{mean}} = \frac{1}{m} \cdot$ `sum(`$\mathrm{Re}\left(\mathbf{D}_i^{\mathrm{frames}}\right)$`, along time-direction)` and
   $\mathbf{b}_{\mathrm{mean}} = \frac{1}{m} \cdot$ `sum(`$\mathrm{Im}\left(\mathbf{D}_i^{\mathrm{frames}}\right)$`, along time-direction)`.

   c) Prepare the matrices
   $\mathbf{T}_{\mathrm{Re}} = \mathrm{Re}\left(\mathbf{D}_i^{\mathrm{frames}}\right)\ .-\ \mathbf{a}_{\mathrm{mean}}$, $\mathbf{T}_{\mathrm{Im}} = \mathrm{Im}\left(\mathbf{D}_i^{\mathrm{frames}}\right)\ .-\ \mathbf{b}_{\mathrm{mean}}$, and
   $\mathbf{O}_{\mathrm{Im}} = (-1)\ .*\ \mathrm{Im}\left(\mathbf{D}_i^{\mathrm{frames}}\right)\ .+\ \mathbf{b}_{\mathrm{mean}}$.

d) Using the macro `@tullio`, translate the estimator written in matrix notation into fast machine code. Note that k and l remain indices of frequencies and relate to rows of the matrices in step 6c, whereas j is an index of time, relating to their columns. Due to Einstein sum convention, `@tullio` sums over j.

```
@tullio B_Im[k, l] += T_Re[k+l, j]·(T_Re[k, j] · T_Im[l, j] + T_Re[k, j] · T_Im[l, j])
@tullio B_Im[k, l] += O_Im[k+l, j]·(T_Re[l, j] · T_Re[k, j] + T_Im[k, j] · T_Im[l, j])

@tullio A_Re[k, l] += O_Im[k+l, j]·(T_Re[k, j] · T_Im[l, j] + T_Im[k, j] · T_Re[l, j])
@tullio A_Re[k, l] += T_Re[k+l, j]·(T_Re[k, j] · T_Re[l, j] + T_Im[k, j] · T_Im[l, j])
```

e) Multiply $\mathbf{A}_{\mathrm{Re}}$ and $\mathbf{B}_{\mathrm{Im}}$ with p_{t3} and add them elementwise to the corresponding parts of the bispectrum:
$\mathbf{S}^3_{\mathrm{Re}} \mathrel{.}+= p_{\mathrm{t3}} \mathrel{.}\cdot \mathbf{A}_{\mathrm{Re}}$
$\mathbf{S}^3_{\mathrm{Im}} \mathrel{.}+= p_{\mathrm{t3}} \mathrel{.}\cdot \mathbf{B}_{\mathrm{Im}}$.

f) Set all elements of $\mathbf{A}_{\mathrm{Re}}$ and $\mathbf{B}_{\mathrm{Im}}$ to zero.

7. Obtain the real and imaginary parts of S^3_z by multiplying the scalar $\frac{N}{T\sum_{j=0}^{N-1}|g_j|^3}$ with $\mathbf{S}^3_{\mathrm{Im}}$ and $\mathbf{S}^3_{\mathrm{Re}}$.

The implementation of this algorithm has been tested by comparing its results with the implementation of another, matrix-multiplication based algorithm for the bispectrum developed by Markus Sifft and used in [30] given a suitable signal. The results were found to be compatible.

During the course of preparing this work, the implementation of this algorithm sadly broke down, leaving not enough time to create a working alternative. The origin of the malfunction could be traced back to the `@tullio`-macro not being executed any more, leaving the matrices $\mathbf{A}_{\mathrm{Re}}$ and $\mathbf{B}_{\mathrm{Im}}$ (and as follows also $\mathbf{S}^3_{\mathrm{Re}}$ and $\mathbf{S}^3_{\mathrm{Im}}$) filled with zeros. Interestingly, after starting Julia, the macro was executed properly the first time the function was called, but not from the second call onwards. When the same code was tested on another (considerably slower) PC, it again produced non-zero bispectra, so the problem is probably not directly due to a programming mistake.

2.6 Presentation of S^4 and S^3

The higher-order spectra obtained in this work contain values that span over various orders of magnitude, which means that some possibly interesting features can remain hidden if the colours used for plotting are only linear functions of the original data.

In [30], the problem has been solved by multiplying an arsinhyp()-function to the spectra, which can be tweaked by modifying offset and curvature to map the values almost linearly in a certain range of values around the offset and to compress the rest about logarithmically (private communication with Daniel Hägele and Markus Sifft).

As an alternative, this work makes indirect use of logarithms to compress the whole scale of values within a single plot while retaining the visual relations between regions of positive and negative correlations. As the logarithm to any basis is not defined for values equal to or smaller than zero, one cannot apply a simple log()-function to the correlation spectrum.

Instead, one changes the data to be plotted like this:

1. Prepare a new matrix $S_Z^{\log}$ of the same dimensions as the spectrum to be plotted.

2. **Make sure that positive correlations stay positive and negative stay negative:**
   Identify the entry closest to zero and calculate its absolute value $s_{\text{abs}}^{\min}$. Multiply the whole spectrum with a factor $a_{\text{scale}} \geq 1/s_{\text{abs}}^{\min}$.
   This makes sure that the logarithm of the absolute values in the spectrum is always bigger than zero (because $\log_N(1) = 0 \; \forall \; N$), so that positive and negative correlation stay visually separated. However, it also means that the values of the log-scale are not at all indicative of the original values of the spectra.
3. **Apply the logarithm separately for positive and negative values:**
   Iterating through all entries x_{S} of the spectrum, take $x_{\text{new}} = \log_N(|x_{\text{S}}|)$ with a freely chosen base N. If $x_{\text{S}} > 0$, set the value of $S_Z^{\log}$ at the corresponding position to x_{new} and to $-x_{\text{new}}$ if $x_{\text{S}} < 0$.

It must be clear that this procedure is more complicated and certainly not any better than scaling the data with the arsinhyp(). The only real difference is that with the log()-visualization, one does not have a range of linear mapping, which makes the application simpler, but also less flexible.

a) original S^4 **b)** $\log_{10}$-visualisation

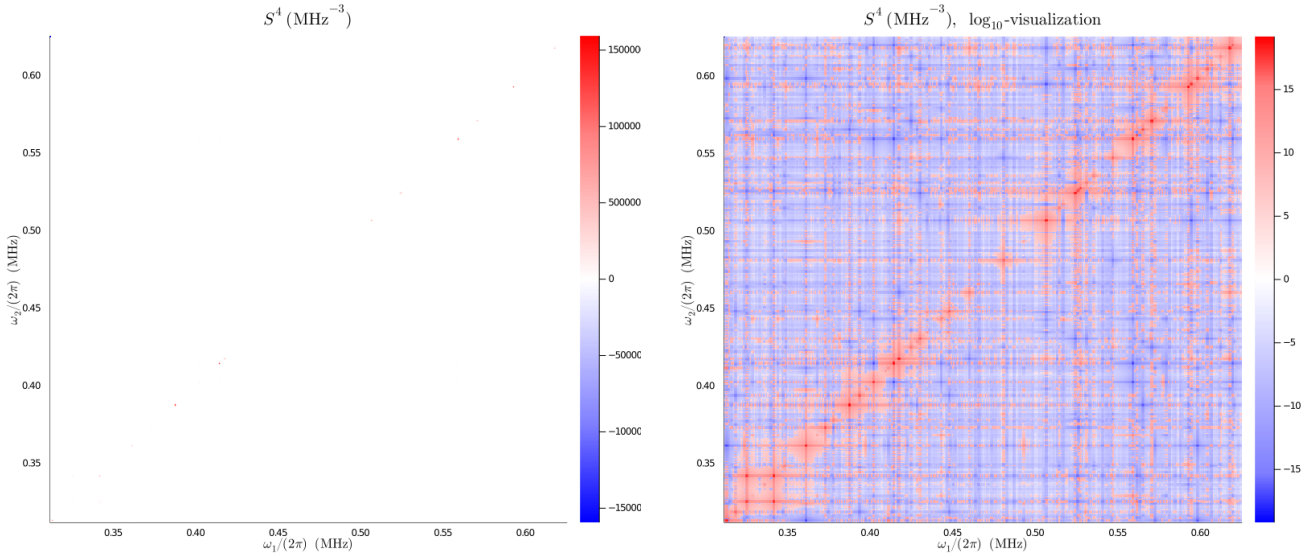


Table 1: In **a)**, one can see almost nothing, because some very small dots of high correlation on the main diagonal make the whole spectrum appear white, whereas the logarithmic visualization in **b)** allows to spot a lot of structure looking different from white noise.
Note that this specific spectrum has been obtained from a signal that was not suited to be processed by the estimators used here in a mathematically correct way. Therefore, it can only serve to demonstrate the effects of the logarithmic visualization.

3 Pulsed Gaussian Noise in Julia

The general approach to construct a time-domain signal $\mathbf{x}(t)$ followed in this work is to calculate a pulse filter function $\mathbf{p}(t)$ on a one-dimensional array, and then multiply it elementwise with another array containing the "substance" of the signal. The pulse filter represents the rotating radio-emission cone of the pulsar during its transit across the line of sight, and the "substance" is made of normally distributed random numbers in the case of Gaussian noise or an elementwise sum of trigonometric functions with certain relative phases. Accordingly, the models analysed in this work only account for emission cones with varying flux density, but not for changes of the type of the emission (e.g. concerning phase relationships between different frequency bands and coherence times) within the cone.

Additionally, it should be noted that only the so-called "integrated pulse profile" of a pulsar is stable when observed in a certain frequency band, whereas the shape of the individual pulses changes from pulse to pulse[18]. Nevertheless, for computational efficiency the single pulses will be treated like ever-stable integrated pulse profiles, which (usually) are the sums of the power measured by the radio detector, added with respect to the rotational phase of the pulsar.

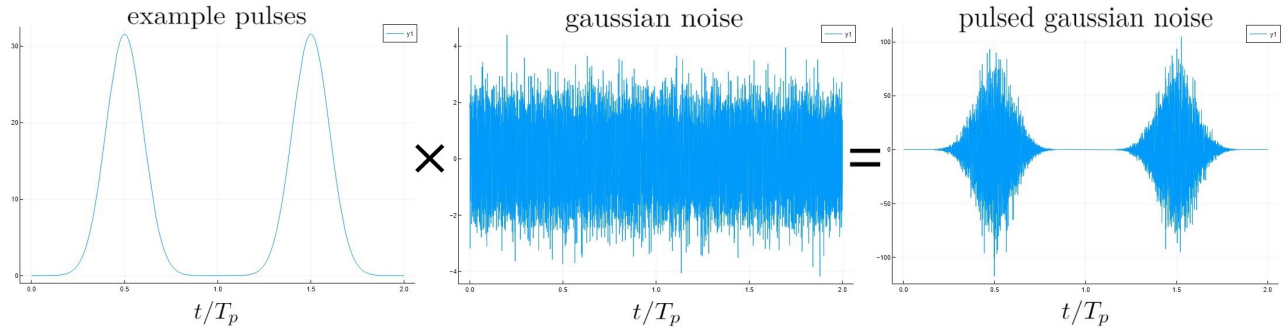


Figure 5: Scheme of how $\mathbf{x}(t)$ is produced by multiplying a pulse filter-array (example pulses) with normally distributed random numbers (gaussian noise).

The Gaussian noise $\mathbf{g}(t)$ is simply a one-dimensional array containing normally distributed random numbers with standard deviation 1 generated by the `randn()`-function implemented in Base Julia. This function utilizes a special version of the Mersenne-Twister algorithm presented in [25]. Although the seed used to calculate a sequence of random numbers can be specified in Julia, it is not always ensured that this sequence will stay reproducible in the future, because the algorithms involved in the calculation might be updated [29]. However, the exact sequences of random numbers do not need to be reproducible in this work, because the properties of the resulting spectra will most likely only be dependent on the overall statistical properties of the signals.

3.1 Functions for the pulse filter

Most integrated pulse profiles can be well fitted with one or several (analytic) Gaussians [24] [16], even to the extent of using Gaussians as templates for pulsar timing [24]. As the pulse profiles are proportional the electric power that a radio receiver outputs, which is proportional to the square of the input voltage respectively the electric field strength of the radio signal [13] that are modeled in this work, it would be the most sensible approach to use a pulse filter that is the square root of a Gaussian. However, since the Gaussian is a more common function in natural sciences and this work does not aim for maximum precision in modeling pulsar radiation, the pulses described here will be Gaussians.

In principle, any Gaussian with variables to adjust the characteristic width and possibly its time-domain shift relative to other Gaussians that are used to describe a multi-peaked pulse can be used.

More specifically, the probability density function of a normal distribution has been employed, which has the advantage that its integral is always scaled to one:

$$p(t) \;=\; \frac{1}{\sigma_\mathrm{t}\sqrt{2\pi}} \cdot \exp\left(-\frac{(t - T_\mathrm{p}/2)^2}{2\sigma_\mathrm{t}^2}\right) \tag{34}$$

The independent variable t stands for time, T_p is the time the pulsar takes to rotate once and σ_t sets the width of the pulse.

3.2 Calculating the pulse filter

In principle, it is possible to calculate the pulse-filter for one period on one short array, then copy this array and concatenate the copy with the original array as often as it is needed to reach the length of the signal, or insert the values of the short copy into a longer array periodically. As this would correspond to an observational case where the pulse-period is a whole-numbered multiple of the sampling interval, this work opts for a slightly more precise approach, where a periodic pulse filter is actually calculated as a function of an underlying time series like this:

1. Prepare a prefactor p_exp which contains every factor of the exponent except t, so e.g $p_\mathrm{exp} = -\frac{1}{2\sigma_\mathrm{t}^2}$.

2. Prepare another prefactor p_norm, e.g. $p_\mathrm{norm} = \frac{1}{\sigma_\mathrm{t}\sqrt{2\pi}}$ when the PDF of a normal distribution is used.

3. Prepare an ascending sequence of natural numbers with a step size of one and the length of $\mathbf{x}(t)$, using the `range()`-function from Base Julia, and multiply it with the sampling interval δt to obtain a time series $\mathbf{t}$.

4. Substract $\frac{T_\mathrm{p}+\delta t}{2}$ from the time series, so that the array elements contain the time marking their middle and the pulse filter does not start in the middle of one pulse.

5. Iterate through all elements t in $\mathbf{t}$ and set
   $t = t - \texttt{trunc}(t/T_\mathrm{p})\cdot T_\mathrm{p}$,
   where `trunc()` means the Julia function which rounds down its numerical argument to integer values towards zero. This step ensures that the Gaussian is repeated periodically.

6. Use the broadcasting-syntax to first calculate the elementwise absolute square `abs2.()` of $\mathbf{t}$, multiply it with p_exp, take the elementwise exponential `exp.()` and finally multiply it with p_norm:
   $\mathbf{p}(t) = \; p_\mathrm{norm} \cdot \;$ `exp.(`p_exp `·` `abs2.(t))`

Considering the performance of this algorithm and the fact that the pulse filters are saved on a hard drive instead of being calculated every time one wants to obtain higher-order spectra, it has been found very much acceptable to leave its computation on the CPU without any kind of parallelization.

4 Pulsed Coherent Radiation in Julia

4.1 General considerations

Like in the case of pulsed Gaussian noise, the pulsed coherent radiation will be calculated by multiplying the same pulse filters $\mathbf{p}(t)$ as in chapter 3 with the type of signal that the simulated pulsar emits, that is a form of broadband phase-coherent radio emission hereafter called $\mathbf{s}(t)$.

The latter will be modeled as a sum of sine-waves with varying frequencies that cover a certain bandwidth continously provided the fourier transform has short enough windows so it does not resolve the individual frequencies. The general idea is to think of a DFT with window length N_{coh} and then "fill the power spectrum" by adding sine waves with wave vectors k_i^{sin} where $1 \leq k_i^{\text{sin}} < N_{\text{coh}}/2$. This way, the actual frequencies of the sine waves are determined only later during the analysis of the pulsed signal, because the sampling frequency of the signal is first calculated when the time for one pulsar rotation is chosen.

While calculating any type of coherent radiation to be used with the estimators mentioned in [27], which includes all estimators used in this work, it is absolutely essential to bear in mind that these are only mathematically valid for coherence lengths shorter than the window length of the DFT. For radiation that looks "to coherent", they are expected to produce erratic results (private communication with Daniel Hägele and Markus Sifft).

Starting with the expression

$$\sin(\omega j \ + \phi) \qquad \text{with } \omega \ = \ 2\pi f \ = \ 2\pi \frac{k_i^{\text{sin}}}{N_{\text{coh}}} \quad \text{and} \quad \phi \ = \ 2\pi \cdot a_{\text{rand}}, \tag{35}$$

one obtains a time series of coherent radiation as a discrete sum of individual sine waves:

$$\mathbf{s}(j) \ = \ \sum_i A_i \sin\left(2\pi \frac{k_i^{\text{sin}}}{N_{\text{coh}}} j \ + \phi_\omega(k_i, j) \ + \phi_{\text{time}}(j)\right) \tag{36}$$

where j is the index of an array element within the time series, δt the time-duration of one array element, f is a formally dimensionless frequency-like variable, ω is in radians, N_{coh} is the length of one DFT-window providing a reference to the wave number k_i^{sin} and a_{rand} is a factor between zero and one that can hold arbitrarily distributed random variables. The individual maximum amplitude A_i of each sine wave is always unity for the signals examined, but it could be set to vary for different i to produce a different looking power spectrum.

$\phi_\omega(k_i^{\text{sin}}, j)$ and $\phi_{\text{time}}(j)$ are two random phases between 0 and 2π, which indicate that it is possible for a sine wave at a certain moment in time to have one somehow randomly chosen phase offset depending on the time and another one depending on frequency or wave number. Here, the phase offsets between different frequencies $\phi_\omega(k_i^{\text{sin}}, j)$ will also be randomly chosen, because if the sine waves are all in phase with each other, one would observe heavy spikes in the time-series due to interference, and also heavily u-shaped power spectra if the window-length of the DFT is below N_{coh}. However, this "statistical independence" of frequencies regarding their phases is not necessarily what could be expected from a real pulsar.

Care has been taken to optimize the choice of the k_i^{sin} between the smallest value k_{start} and the highest value k_{stop} to obtain

a) a power spectrum that looks as flat as possible and
b) to avoid a beat in the time series.

To address concern **a)**, $k_i^{\sin}$ must be chosen in a way the individual frequencies to not interfere with each other and that at least one $k_i^{\sin}$ contributes to each bin of the later DFT (otherwise the DFT would have unwanted "valleys"). Bringing this together, it means that exactly one $k_i^{\sin}$ has to make a contribution to each bin, aside from maybe spectral leakage that would also happen with Gaussian noise. That means that the initial reference length of a DFT-window N_{coh} on which the $k_i^{\sin}$ must be the same as in the later analysis of the signal, if the minimum difference between any $k_i^{\sin}$ and $k_q^{\sin}$ is 1.

For concern **b)**, it must be avoided that the difference $k_i^{\sin} - k_q^{\sin}$ is equal or very similar for a number of i and q. To assure this, a small evenly distributed random variation has been added to each $k_i^{\sin}$, which is small enough that $k_i^{\sin}$ cannot come too close to $k_q^{\sin}$ to make a contribution to its bin in the DFT (to avoid compromising on concern **a)**).

To sum it up, the wave vectors have been chosen like:

$$k_i^{\sin} = k_i + (k_{\text{rand}} - 0.5) \cdot \delta k / 2.4 \tag{37}$$

$$\text{with} \quad k_i = k_{\text{start}},\ k_{\text{start}+1},\ k_{\text{start}+2},\ k_{\text{start}+3},\ ...,\ k_{\text{stop}} \text{ and } k_{i+1} - k_i = \delta k = \text{const.} \tag{38}$$

More specifically, the random contribution to the wave-vectors are evenly-distributed random numbers between 0 and 1 generated with the `rand()`-function in Base Julia, and δk has been set to 1 due to concern **a)**.

4.2 Algorithms for coherent radiation

Finally, one can calculate coherent radiation with phase-jumps of constant periodicity (meaning individual sine waves always have $n_{\text{coh}} << N_{\text{rfft}}$ array elements before the phase changes) like this:

1. Prepare a one-dimensional array $\mathbf{s}(t)$ with length N_{s} containing zeros matching the length of the final $\mathbf{x}(t)$ or exceeding it.

2. Prepare a one-dimensional array $\mathbf{r}_1(t)$ with length n_{coh} by converting a `range()` of linearly spaced natural numbers into an explicit data type (e.g. double precision floating-point numbers for calculations on the CPU) and multiply it with 2π. Prepare a one-dimensional array $\mathbf{r}_2(t)$ with the same length.

3. Choose a window-length N_{coh} like for discrete Fourier transforms as a reference.

4. Define a set of wave numbers $k_i^{\sin}$ by choosing the smallest wave number k_{min}, the biggest one k_{max} and the interval between them δk ($\delta k = 1$ for all signals in this work).

5. Iterating through wave numbers $k_i^{\sin}$, do the following:

    a) Iterate through a counting-index i from 1 to `trunc(`$N_{\text{S}}/n_{\text{coh}}$`)` and do the following:

        i. Copy the elements from $\mathbf{r}_1(t)$ to $\mathbf{r}_2(t)$.
        Alternatively one could also write the below statements in one line. The aim is solely not to overwrite the ascending natural numbers multiplied with 2π stored in $\mathbf{r}_1(t)$.

        ii. Multiply each element of $\mathbf{r}_2(t)$ with $\frac{k_i}{N_{\text{coh}}}$.

        iii. Calculate the total phase offset $\phi = 2\pi \cdot a_{\text{rand}}$ and add it to each element of $\mathbf{r}_2(t)$. This already encompasses $\phi_\omega(k_i^{\sin}, j)$ and $\phi_{\text{time}}(j)$, because ϕ is calculated independently for

every little coherent sine wave independent of its frequency.

iv. Broadcast the `sin()`-function onto $\mathbf{r}(t)$, thereby calculating the sine of every element.

v. Add $\mathbf{r}_2(t)$ to the segment of $\mathbf{s}(t)$ from index i_{start} to i_{stop},
where $i_{\text{start}} = (i-1) \cdot n_{\text{coh}} + 1$ and $i_{\text{stop}} = i_{\text{start}} + n_{\text{coh}} - 1$.
With notation mimicking Julia syntax: $\mathbf{s}(t)\texttt{[}i_{\text{start}}\texttt{:}\ i_{\text{stop}}\texttt{]}\ \ \texttt{.+=}\ \ \mathbf{r}_2(t)$

For performance cosiderations, it is very helpful to avoid any new allocations of memory within the innermost loop, e.g. by avoiding new instances of $\mathbf{r}_2(t)$, and by directly adding a new sine wave to $\mathbf{s}(t)$. This has e.g. been way more efficient than parallelizing the outer loop onto 4 separate threads running on different cores of the same machine.

However, unless mentioned otherwise, all signals mimicking broadband coherent radiation with strictly periodic phase jumps have been calculated on a CUDA-enabled Nvidia-GPU using the package `CUDA.jl` [3] in the following manner (using single-precision floating-point numbers):

1. Choose a window length N_{coh} as a reference for the wave vectors (to be used for the DFT-windows later).

2. Choose N_{S}, the length for $\mathbf{s}(t)$, the future array to hold the sum of sine waves.

3. Choose n_{coh} like in the above algorithm.

4. Find the integer multiple $l \cdot n_{\text{coh}}$ that is closest to N_{S} and save l.

5. Prepare two $n_{\text{coh}} \times l$ matrices (as GPU-Arrays), $\mathbf{s}(t)$ containing zeros and $\mathbf{s}_{\text{help}}(t)$ containing ones.

6. Prepare a vector $\mathbf{r}_{\text{int}}$ with length n_{coh} and multiply it with $2\pi/N_{\text{coh}}$.

7. Prepare a vector ϕ_{rand} of length l containing zeros.

8. Complete the preparation of $\mathbf{s}_{\text{help}}(t)$ by elementwise multiplication of it with the column-vector $\mathbf{r}_{\text{int}}$, so that every column of $\mathbf{s}_{\text{help}}(t)$ contains the same natural numbers scaled with 2π.

9. Iterate through all wave vectors k_i^{sin} doing the following:

   a) Obtain random phases for each coherence interval:
   Fill the vector with new evenly distributed random numbers between 0 and 1 and multiply it with 2π.
   To do this within the loop assures that not only every coherence interval gets its own randomly determined phase, but also that the individual frequency channels do not have a fixed phase relationship for longer than the coherence time. In other words: $\phi_\omega(k_i^{\text{sin}}, j)$ and $\phi_{\text{time}}(j)$ are taken care of together, but are still independent.

   b) Perform an elementwise addition onto $\mathbf{s}(t)$ of the sine-function broadcasted onto the elementwise sum of the product of $\mathbf{s}_{\text{help}}(t)$ with k_i^{sin} and the row vector ϕ_{rand}^T :

$$\mathbf{s}(t)\ \texttt{.+= sin.(}\mathbf{s}_{\text{help}}(t)\ \texttt{.*}\ k_i^{\text{sin}}\ \texttt{.+ transpose(}\phi_{\text{rand}}\texttt{))} \tag{39}$$

10. Use the `reshape()`-command available in Julia for many array implementations to transform $\mathbf{s}(t)$ into a one-dimensional array.

11. Copy $\mathbf{s}(t)$ to the CPU (from VRAM to RAM).

It might be remarked that the above algorithm is only dependent on higher-level array operations provided by a programming language, but not on the specific device. Therefore, while it has been implemented and found usage with `CUDA.jl` for GPUs, one could easily employ it on the CPU as well. However, it has been found that for the same parameters of the signal to be calculated, this last algorithm was up to about 100-times faster in accomplishing its task on the machine mainly used for this work, when comparing single precision on the GPU to double precision on the CPU.

Furthermore, both algorithms also work for the case of a single sine-wave (or a small number of them) with pre-defined frequency or wave number, which has been used as a test for the correlation spectrum and the calculation of its frequency axis. However, this use case does not require the degree of optimization as the broadband coherent radiation and can therefore quickly be calculated within the analysis script after the pulse filter is loaded from the hard drive.

Lastly, to provide an even basis for the comparison of the higher-order spectra, any radiation consisting of summed sine waves is scaled to exhibit a mean absolute value of 1 like the Gaussian noise. This is simply done by calculating mean absolute value of the original coherent signal and dividing by it.

5 Comparison of Polyspectra

5.1 General remarks

The standard broadband coherent signal used in this work has the following properties:

$$k_i \in \mathbb{N},\ k_i \in [200,\ 7992], \tag{40}$$

$$k_i^{\sin} = k_i + (k_{\text{rand}} - 0.5)\ /2.4, \quad k_{\text{rand}} \in [0, 1) \tag{41}$$

$$n_{\text{k}} = 7792 \tag{42}$$

$$n_{\text{coh}} = 2500 \text{ array elements} \tag{43}$$

$$N_{\text{coh}} = 16384 \text{ array elements} \tag{44}$$

$$\phi_\omega(k_i, j), \phi_{\text{time}}(j) = 2\pi \cdot a_{\text{rand}}, \quad a_{\text{rand}} \in [0, 1) \tag{45}$$

a_{rand} and k_{rand} are evenly distributed random variables. Moreover, a_{rand} has been generated indepently for $\phi_\omega(k_i, j)$ and $\phi_{\text{time}}(j)$.

For the following scenarios, only one type of pulse shape with $\sigma_{\text{T}}/T_{\text{p}} = 0.05$ has been used. Originally, it was planned to compare additional pulse shapes, one sharper than this one and two which were substantially broader. However, no obvious change in the spectra has been observed when working with these different pulse shapes, so only this one pulse shape has been used.

Please note that if no excerpts from the time series of other signals are shown in the subsections to follow, it means that they look virtually indistiguishable to the pulses presented here.

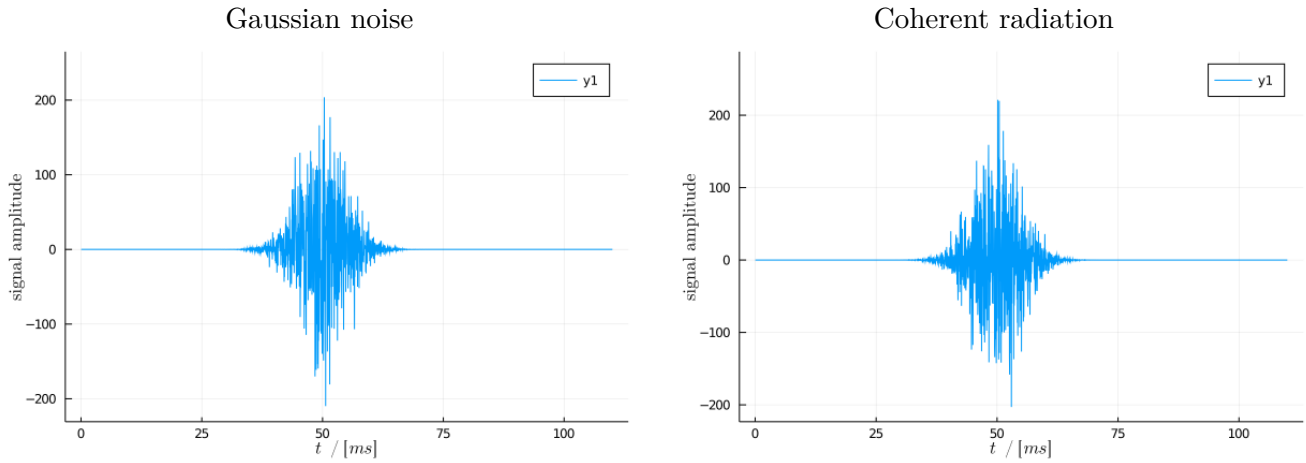


Table 2: Comparison of pulses across one period, taken from the time series of the standard case discussed in 5.2.
Plots like these are mainly saved to check if the signal that is to be analyzed shows some obvious flaws. The (dimensionless) signal amplitude is very similar for both signals, making the higher-order spectra more comparable.

5.2 Standard case

The following settings have been chosen to define the "standard case" of a signal, to which differences in the pulse filter, the DFT and the settings for calculating the higher order spectra are compared:

$T_\mathrm{p} = 0.1$ s	$\sigma_\mathrm{T}/T_\mathrm{p} = 0.05$	$m_\mathrm{p} = 2500$ bins	$t_\mathrm{obs} = 2684$ s
$\delta t = 4 \cdot 10^{-5}$ s	$N_\mathrm{win} = 16384$	$T_\mathrm{win}/T_\mathrm{p} = 6.5536$	$n_\mathrm{pulses} = 26843$

T_p is the rotation period of the pulsar, $\sigma_\mathrm{T}/T_\mathrm{p}$ is the width parameter of the pulse profile scaled to the period, m_p is the exact number of array bins needed to store one pulse, t_obs is the time duration of the whole signal, $\delta t = 4 \cdot 10^{-5}$ is the time duration of one array element of the time series, N_win is the number of array elements in one DFT-window, $T_\mathrm{win}/T_\mathrm{p}$ is the number of pulses in one DFT-window and n_pulses is the number of pulses within the whole time series.

The choice of these parameters is often somewhat arbitrary, but partly motivated by practical considerations. The DFT-windows, for example, cannot be too long to avoid the calculation of the higher-order spectra becoming unnecessarily slow. As follows, the pulse period cannot be too long, because one window should encompass several coherence lengths, which is chosen to coincide with the pulse period (in the standard case).

The values of the power spectra have very similar levels, which can be expected because of the same pulse shape, rotation period and the scaling of the coherent noise. Similarly, the correlation spectra also have comparable ranges of values.

The only immediately noticeable difference is that a subtle rectangular pattern (from here on called the checkerboard-pattern) is visible in the correlation spectrum of coherent radiation, whereas the higher-order spectra of Gaussian noise do not show any structure.

Note that there is a white area at the beginning and the end of the frequency band of the higher-order spectra of coherent radiation, which stays white because there are now sine waves in the signal contributing to it.

Regardless of the type of radiation the pulsed signal comes with, Daniel Hägele guessed that **a)** the correlation spectra would have a slight offset from zero, which would probably be positive, and **b)** there might be a contribution close to zero frequency (or to $1/T_\mathrm{p}$) in them. Concerning **a)**, the means, medians, standard deviations and normalized absolute deviations from median have been calculated for different sets of entries from the correlation spectrum of pulsed Gaussian noise shown in 3. As a reference, they have been calculated for a sequence of normally distributed random numbers with the same number of entries as the S^4 and the same mean absolute value.

The interesting results might be these for the means:

$$s^+_\mathrm{mean} \approx 0.8049703 \pm 0.6183640 \qquad r^+_\mathrm{mean} \approx 0.7979090 \pm 0.6028232 \tag{46}$$

$$s^-_\mathrm{mean} \approx -0.7891078 \pm 0.5895384 \qquad r^-_\mathrm{mean} \approx -0.7980256 \pm 0.6028442, \tag{47}$$

and for the medians:

$$s^+_\mathrm{med} \approx 0.67373801 \pm 0.59682947 \qquad r^+_\mathrm{med} \approx 0.67459576 \pm 0.59174404 \tag{48}$$

$$s^-_\mathrm{med} \approx -0.67178349 \pm 0.58466144 \qquad r^-_\mathrm{med} \approx -0.67459532 \pm 0.59173067. \tag{49}$$

The letter s stands for values derived from entries of the correlation spectrum, whereas r stands for those derived from the reference random numbers. A $+$ $(-)$ indicates that something has been calculated using only values bigger (smaller) than zero from a certain set.

What can be seen is that the means and medians of the positive values are slightly bigger than the negative values for the correlation spectrum. Comparing between the positive and the negative set, the means differ in the first digit behind the dot for the correlation spectrum, but only in the third for the reference noise. Similarly, the medians differ in the third digit for the spectrum compared to the sixth digit for the reference numbers. Additionally, also the measures of dispersion seem to be bigger for the positive values in the actual trispectrum when compared to reference noise, which has not been expected.

This slight preference of positive numbers is obviously not a strong phenomenon, as its scale is well below the measures of dispersion for the means as well as medians. If one was to round to significant digits, it would even be invisible, but it is still the best numerical hint supporting guess **a)** that could be found so far.

Regarding assumption **b)**, some of the trispectra obtained during the course of this work have been inspected visually around the origin of the two frequency axis, and nothing has been found there looking different from randomly coloured squares. But this does not necessarily mean that there is actually nothing, because there has been not enough time for a systematic search of any low-frequency features trying different settings for the signal and its analysis.

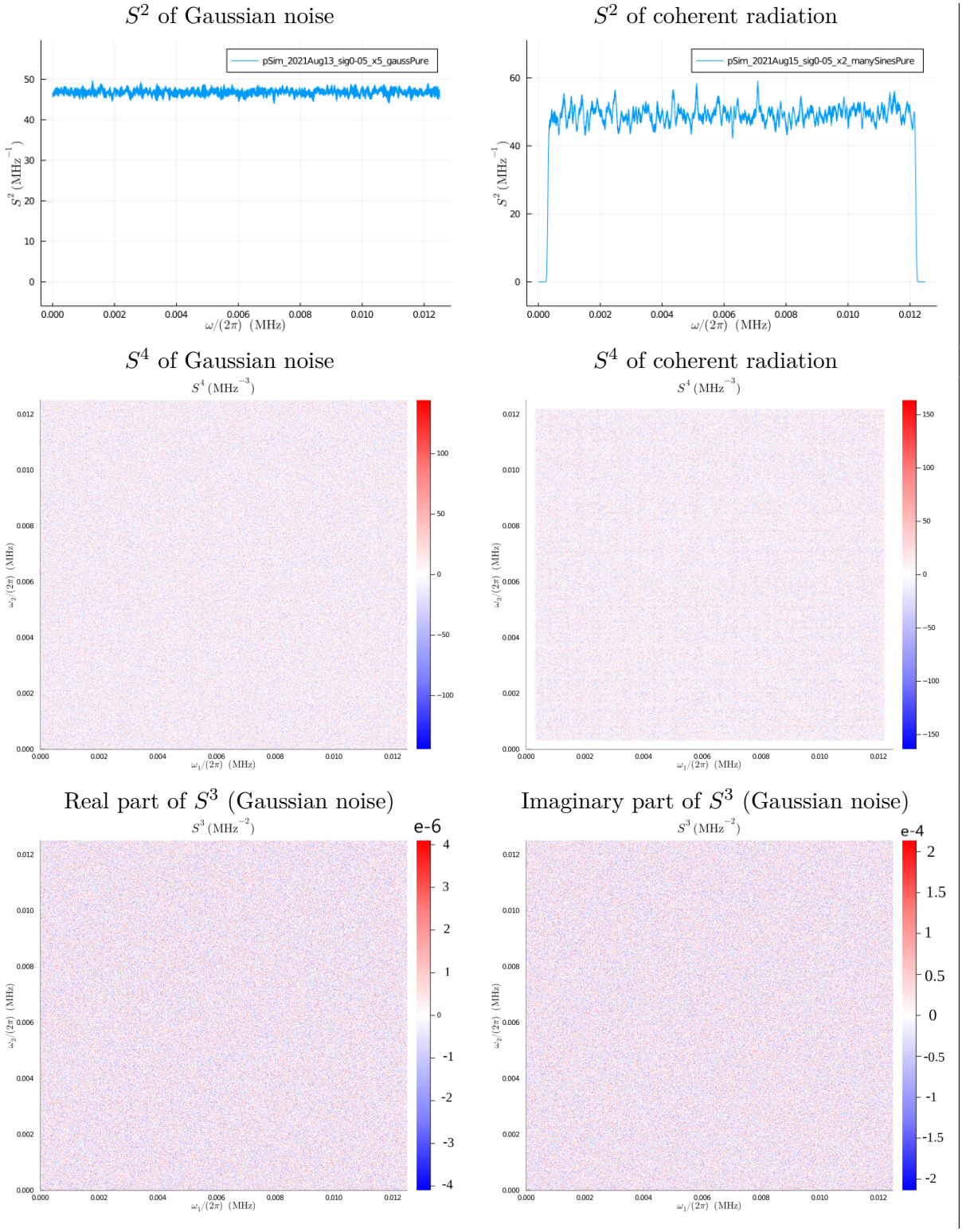


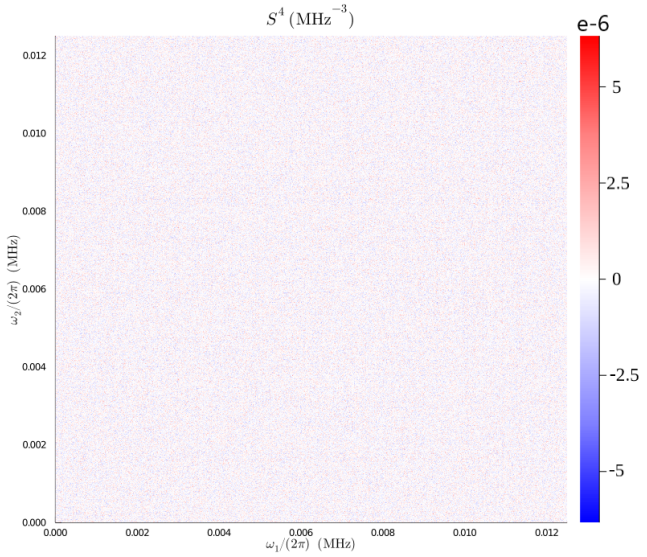
Table 3: Comparison of power- and trispectra of pulsed Gaussian noise vs. coherent radiation, supplemented with the bispectrum for pulsed Gaussian noise.
Because the code for the bispectrum stopped working during the course of this like mentioned in section 2.5, a bispectrum is unfortunately not available for a signal with coherent radiation with these settings.

5.3 Longer period

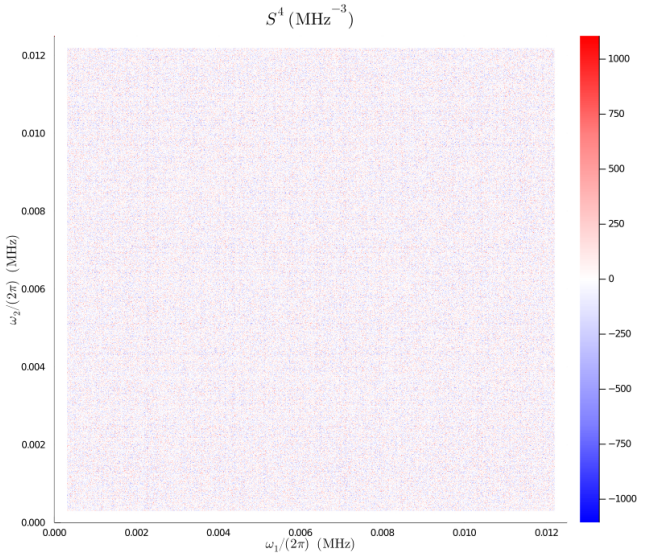
When the DFT-windows are shorter than the pulses are wide, subtle changes become visible. The settings are updated to

$T_\mathrm{p} = 7.86432$ s	$\sigma_\mathrm{T}/T_\mathrm{p} = 0.05$	$m_\mathrm{p} = 196608$ bins	$t_\mathrm{obs} = 4697.60$ s
$\delta t = 4 \cdot 10^{-5}$ s	$N_\mathrm{win} = 16384$	$T_\mathrm{win}/T_\mathrm{p} = 0.08235$	$n_\mathrm{pulses} = 597.333$

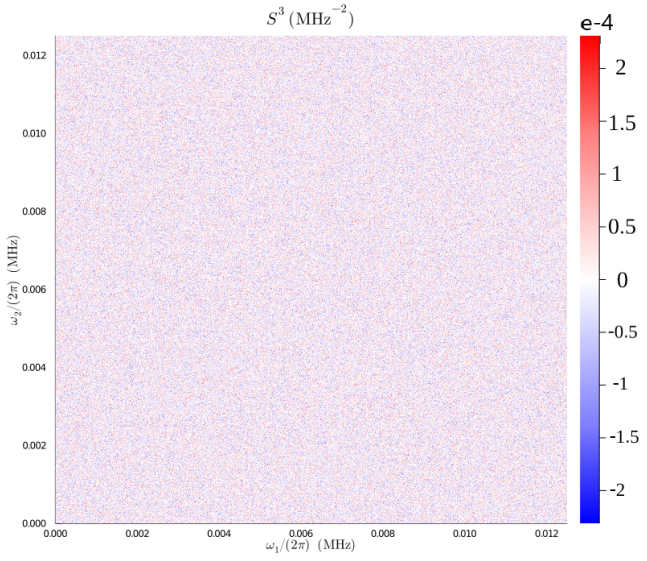
S^4 of Gaussian noise, $m = 32$



S^4 of coherent radiation, $m = 32$



Real part of S^3 of Gaussian noise, $m = 3$



Real part of S^3 of coherent radiation, $m = 32$

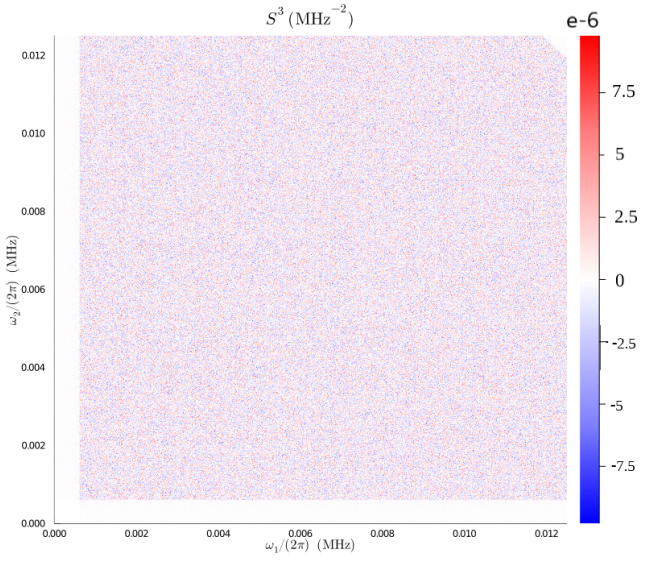


Table 4

A subtle checkerboard pattern shows up in the S^4 of coherent radiation and also in the S^3, but weaker. Furthermore, the trispectra differ about 9 orders of magnitude in their values. It could be that the pattern in the S^4 is stronger than for the other cases presented.

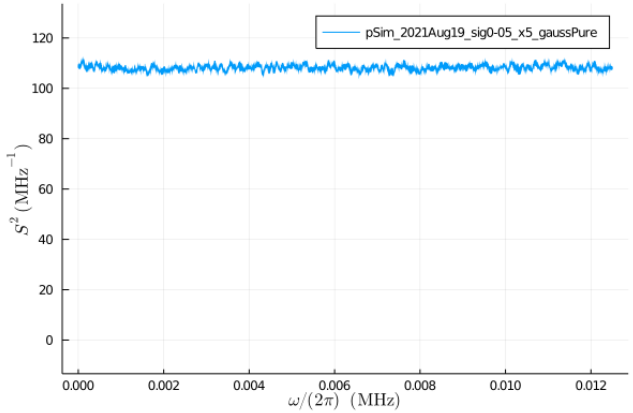
The power spectra look like the ones shown in 5.2, and the imaginary parts of the bispectra for this subsection do not differ visually from their real counterparts, meaning they also do not show any structure.

5.4 Shorter DFT

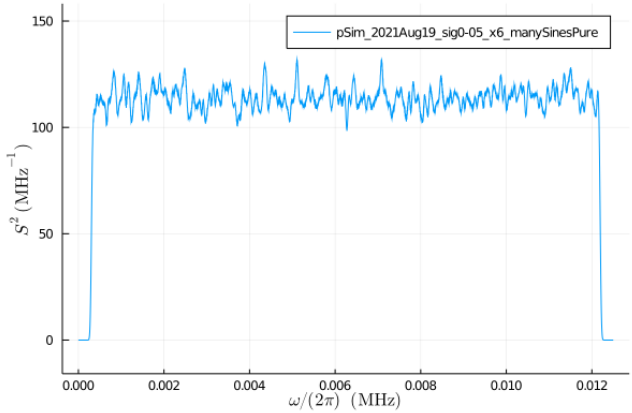
The signal is the same as in the standard case, only the window length of the DFT changes to a shorter value that is hopefully still tolerable for the coherent radiation.

$T_\mathrm{p} = 0.1$ s	$\sigma_\mathrm{T}/T_\mathrm{p} = 0.05$	$m_\mathrm{p} = 2500$ bins	$t_\mathrm{obs} = 2684.334$ s
$\delta t = 4 \cdot 10^{-5}$ s	$N_\mathrm{win} = 7000$	$T_\mathrm{win}/T_\mathrm{p} = 2.8$	$n_\mathrm{pulses} = 26843$

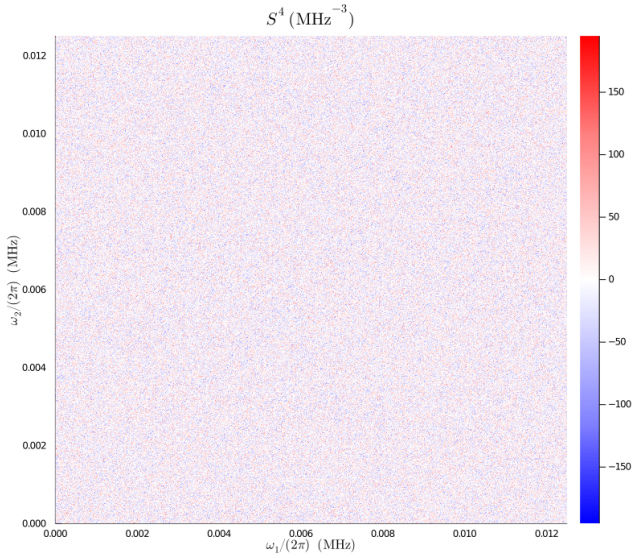
S^2 of Gaussian noise



S^2 of coherent radiation



S^4 of Gaussian noise, $m = 32$



S^4 of coherent radiation, $m = 32$

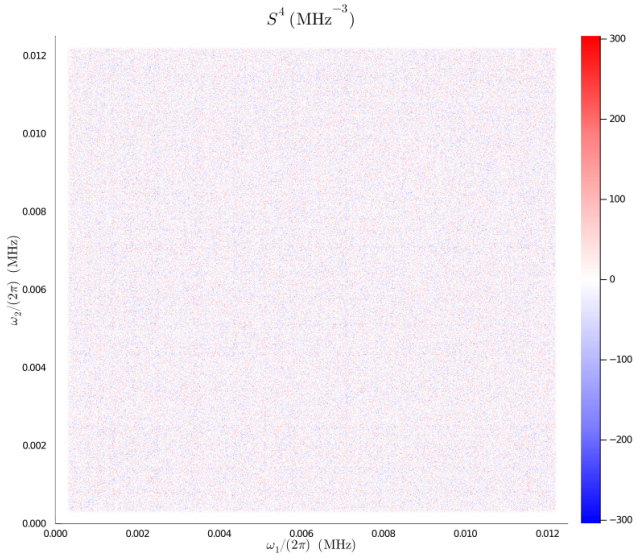


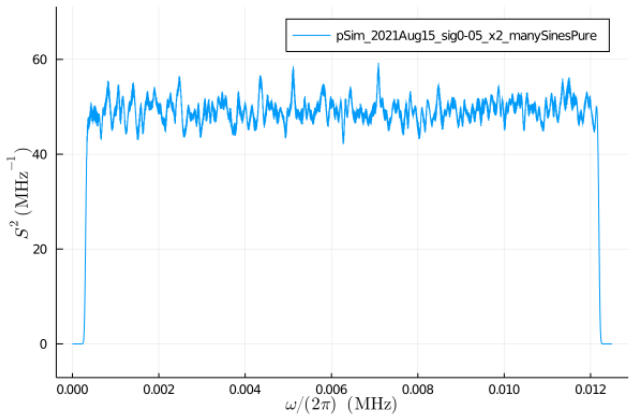
Table 5

In comparison of the two trispectra, one could maybe suppose that the one obtained from coherent radiation again shows some kind of rectangular structure.
Furthermore, the power spectrum of coherent radiation does not look more spiky than for $N_\mathrm{win} = 16344$. That means that the precaution of calculating exactly one sine wave per bin of the later DFT was not necessary in the first place, at least not for coherence lengths that are shorter than window length. The values of S^2 even scale correctly with the relative change of the window length (which is about $1/2.335$).

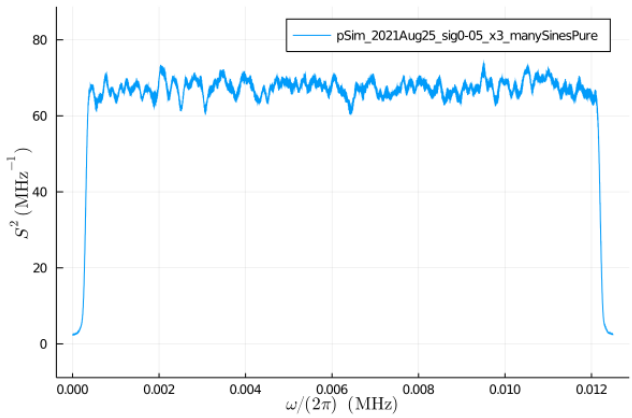
5.5 Shorter coherence time

The pulse filter is the same as in the standard case, as are the settings for the DFT and the trispectrum. Only the coherence interval is now a tenth of what it used to be, namely $n_{\text{coh}} = 250$ (amounts to 0.01 s) instead of 2500 (0.1 s).

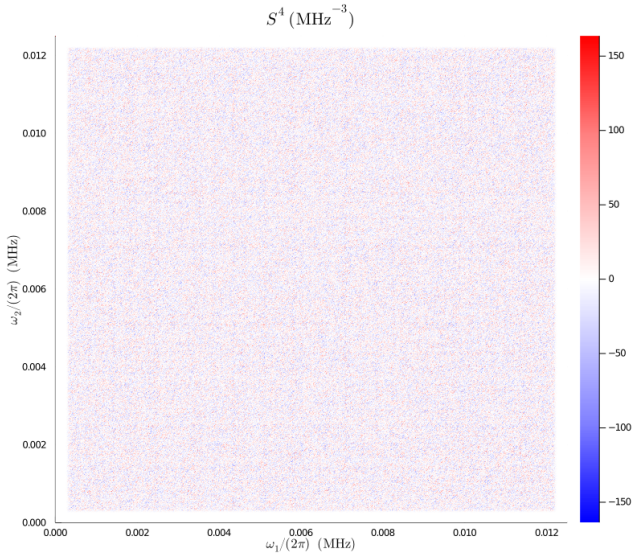
a) S^2 of with $n_{\text{coh}} = 2500$



b) S^2 with $n_{\text{coh}} = 250$



c) S^4 with $n_{\text{coh}} = 2500$, $m = 32$



d) S^4 with $n_{\text{coh}} = 250$, $m = 32$

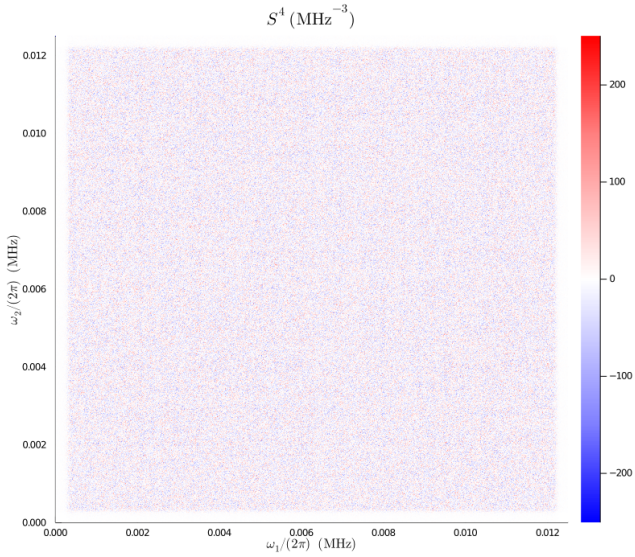


Table 6: On the left in **a)** and **b)**, the standard case of coherent radiation is shown. The spectra from the new signal with $n_{\text{coh}} = 250$ are on the right in **c)** and **d)**.

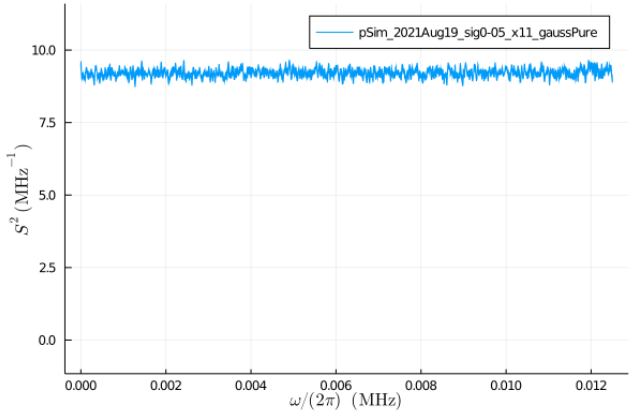
It seems that the "checkerboard-pattern" is less pronounced or even invisible in the correlation spectrum of the signal with shorter coherence time, which makes sense in a way that this signal looks more random to the DFT because 10 times more random phases have effect on a single Fourier-coefficient. In addition, the power- and correlation spectra both have noticeably bigger values as can be seen on the ordinates of **a)** and **b)** respectively on the colour scales of **c)** and **d)**. However, it is not clear why this is so.

5.6 Less symmetry

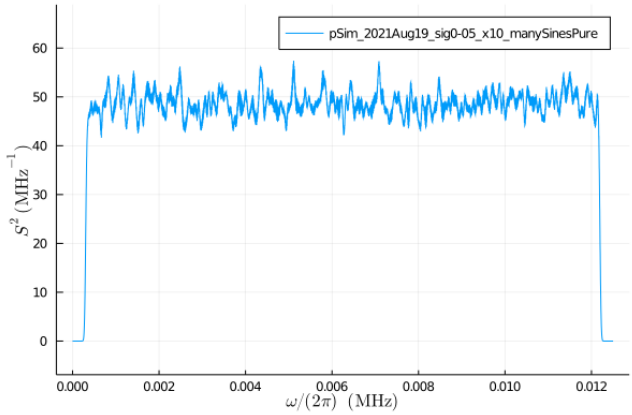
So far, the coherence time and the pulse period have been exactly the same, except in the case of extra-long periods which greatly surpassed the length of the DFT-windows. Therefore, a signal will be analysed where the pulses period is longer, but not a whole-numbered multiple of the coherence time.

$T_\mathrm{p} = 0.1$ s	$\sigma_\mathrm{T}/T_\mathrm{p} = 0.05$	$m_\mathrm{p} = 6100$ bins	$t_\mathrm{obs} = 2684.354$ s
$\delta t = 4 \cdot 10^{-5}$ s	$N_\mathrm{win} = 16384$	$T_\mathrm{win}/T_\mathrm{p} = 2.925$	$n_\mathrm{pulses} = 11983$

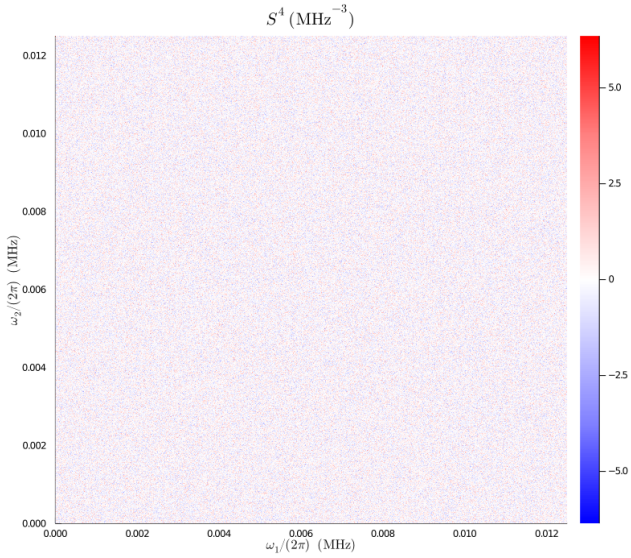
S^2 of Gaussian noise



S^2 of coherent radiation



S^4 of Gaussian noise, $m = 32$



S^4 of coherent radiation, $m = 32$

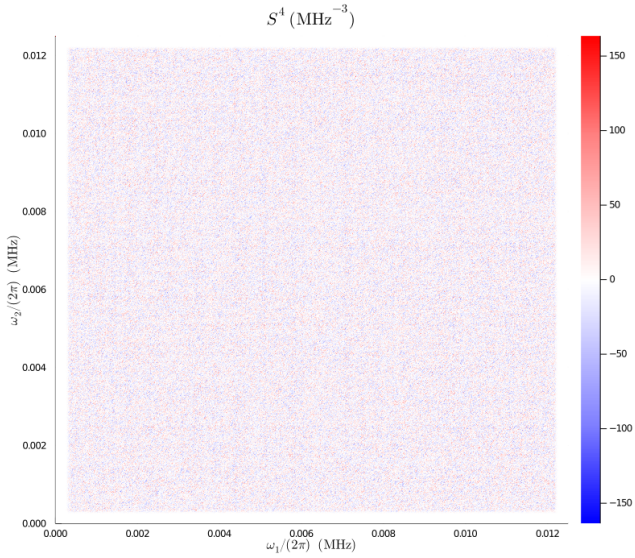


Table 7

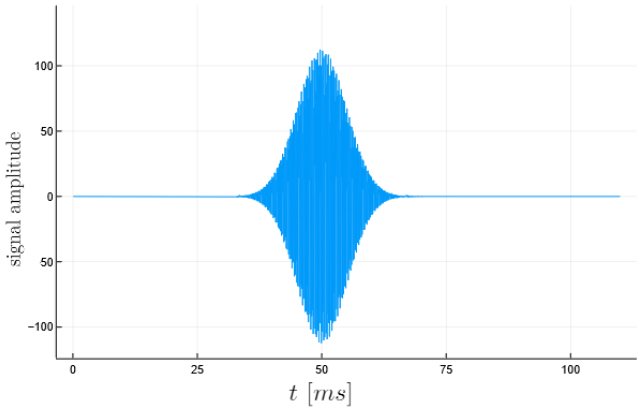
Nothing new is seen: the Gaussian does not show specific features and the coherent radiation exhibits the same checkerboard-pattern as before. The smaller values of the spectra of Gaussian noise may be due do the fact that its pulses are smaller by a factor of about 50.

5.7 A single sine wave

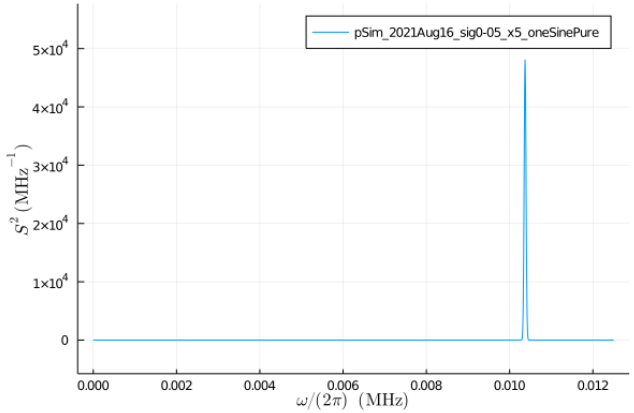
A rather simple, yet instructive example is that of a single sine wave that changes its phase every 2500 bins, here with a frequency of about 0.010375 MHz.

$T_\mathrm{p} = 0.1$ s	$\sigma_\mathrm{T}/T_\mathrm{p} = 0.05$	$m_\mathrm{p} = 2500$ bins	$t_\mathrm{obs} = 5368.689$ s
$t_\mathrm{res} = 4 \cdot 10^{-5}$ s	$N_\mathrm{win} = 7000$	$T_\mathrm{win}/T_\mathrm{p} = 2.8$	$n_\mathrm{pulses} = 53687$

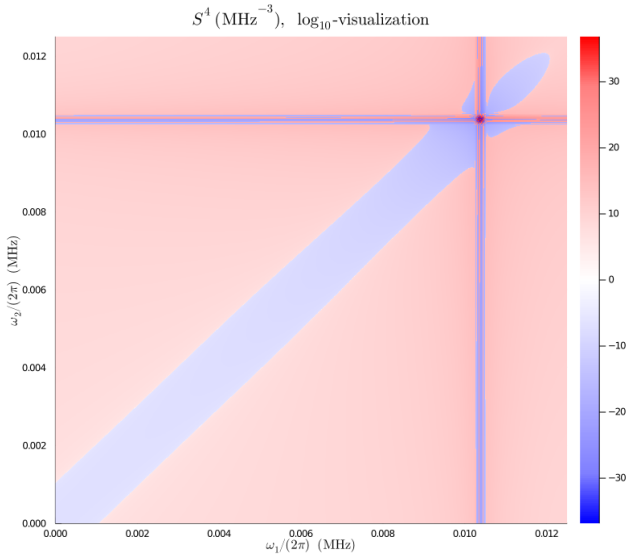
first pulse period



power spectrum



$\log_{10}$-visualization of whole S^4, $m = 32$



Magnified part of S^4 around the peak

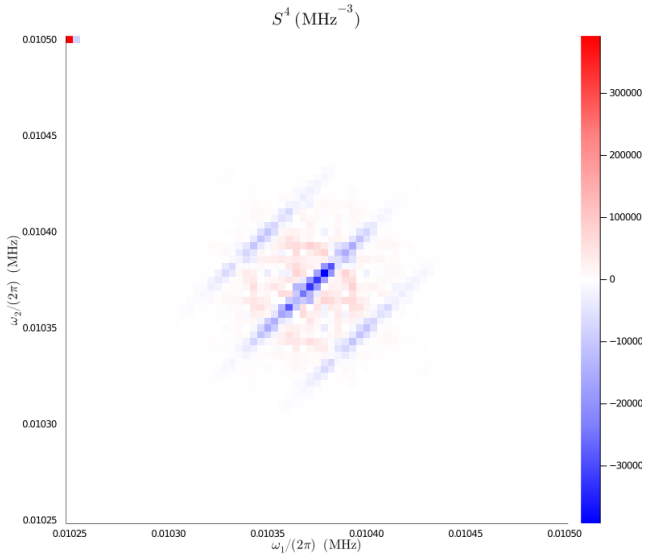


Table 8

Instead of the actual S^4, its logarithmic visualization is presented because the whole S^4 looks absolutely white with just one colourful feature that is too small to be spotted. The little red dot and light blue dots in the upper-left corner of the magnified S^4 do not belong to the data, but have been introduced for the sake of visualization.

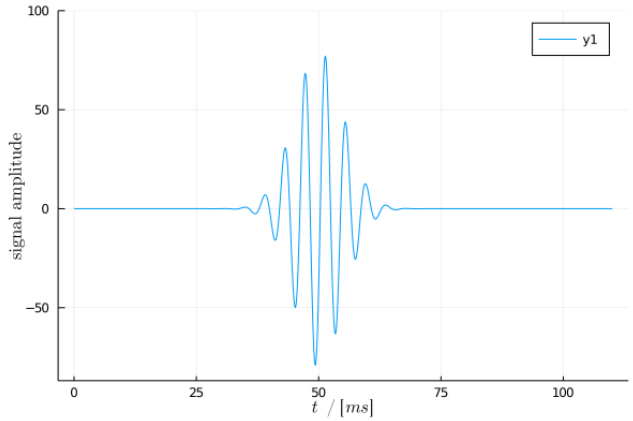
It is interesting that the strongest part of the feature is an anticorrelation. Furthermore, it is notable that there is a line of anticorrelation about $2.75 \cdot 10^{-5}$ MHz away from the main diagonal, which is close to the rotation frequency of the pulses (10^{-5} MHz). Actually, it is very close to the number of pulses per DFT-window ($T_\mathrm{win}/T_\mathrm{p} = 2.8$) times the rotation frequency.
It has already been predicted by Daniel Hägele that one should see some kind of feature in the trispectrum close to $1/T_\mathrm{p}$ (private communication), but this is the first time that it is actually seen after a reliable calculation.

5.8 A low-frequency sine wave

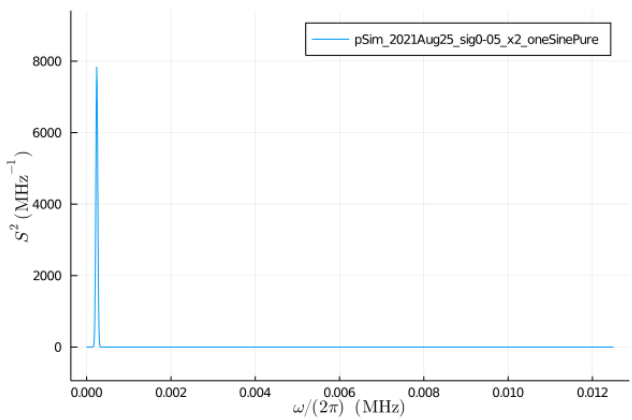
Leaving all settings the same as in section 5.7, but changing the carrier frequency to 0.00025 MHz, one obtains the following results:

$T_\mathrm{p} = 0.1$ s	$\sigma_\mathrm{T}/T_\mathrm{p} = 0.05$	$m_\mathrm{p} = 2500$ bins	$t_\mathrm{obs} = 5368.689$ s
$t_\mathrm{res} = 4 \cdot 10^{-5}$ s	$N_\mathrm{win} = 7000$	$T_\mathrm{win}/T_\mathrm{p} = 2.8$	$n_\mathrm{pulses} = 53687$

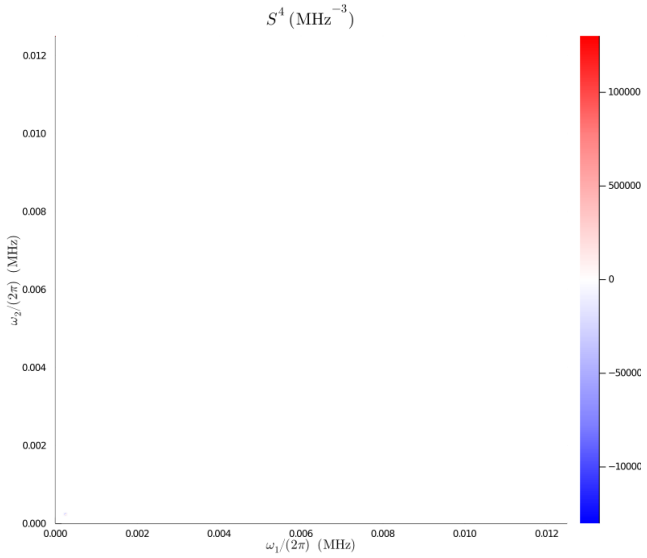
a) first pulse period



b) power spectrum



c) whole S^4, $m = 32$



d) magnified part of S^4, $m = 32$

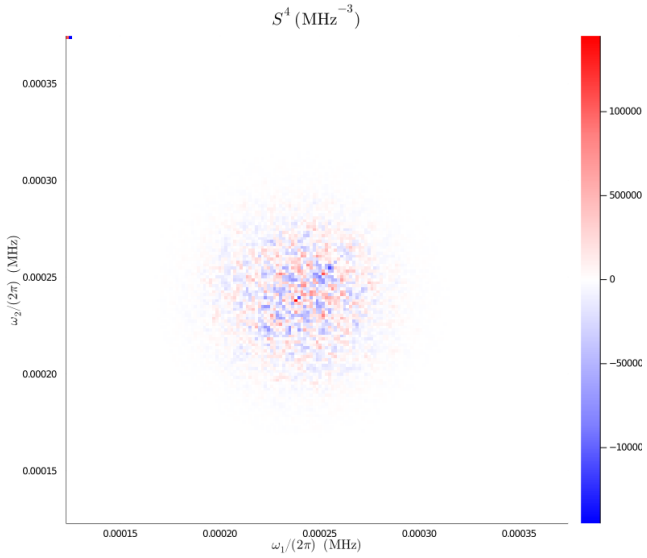


Table 9: **b)**: the power spectrum shows a single clear peak,
c): correlation spectrum with $m = 32$ across whole observed bandwith. The signature from the signal is in the lower-left corner and almost invisible.
d): magnification of **c)** to the frequency range of the injected sine. Notice that the magnification is about as strong as in section 5.7.

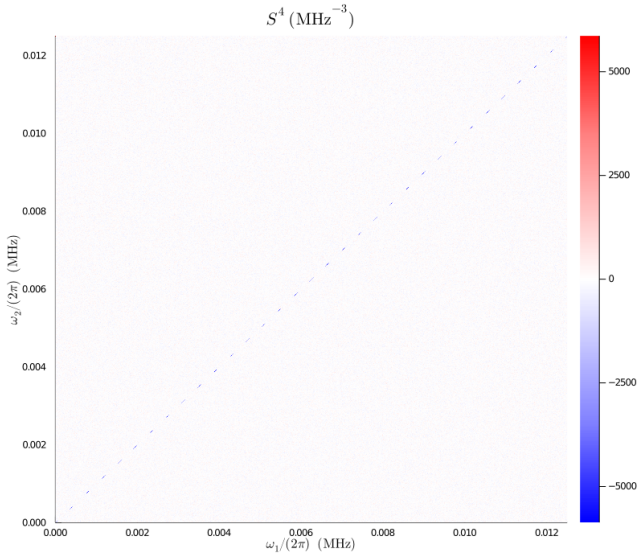
Now, the signature of the single sine wave looks much more washed out, like an almost structure-less blob around the main diagonal. Also note that the range of values within the correlation spectrum is about 3 times bigger than with the high-frequency sine.

5.9 Exploring the main diagonal

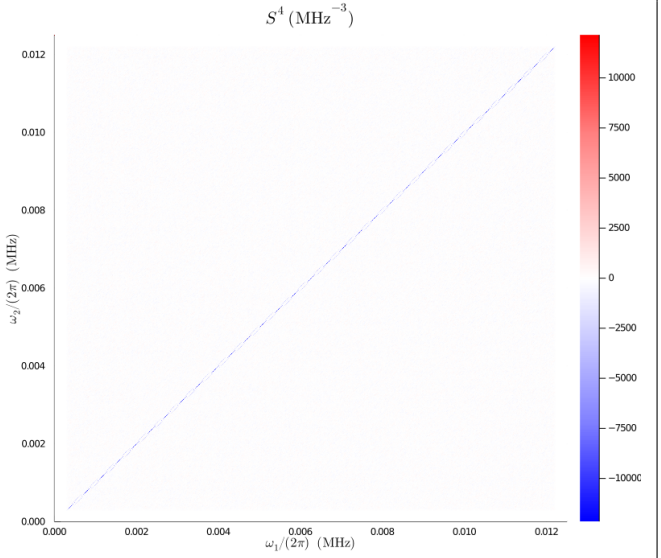
For minimal m, relatively short DFT-windows and using the alternative, faster implementation for $c_4^{(\mathrm{cb})}$ with `Octavian.jl`, one encounters a strange phenomenon not seen before.

$T_\mathrm{p} = 0.1$ s	$\sigma_\mathrm{T}/T_\mathrm{p} = 0.05$	$m_\mathrm{p} = 2500$ bins	$t_\mathrm{obs} = 4026.511s$
$\delta t = 4 \cdot 10^{-5}$ s	$N_\mathrm{win} = 7000$	$T_\mathrm{win}/T_\mathrm{p} = 2.8$	$n_\mathrm{pulses} = 40265$

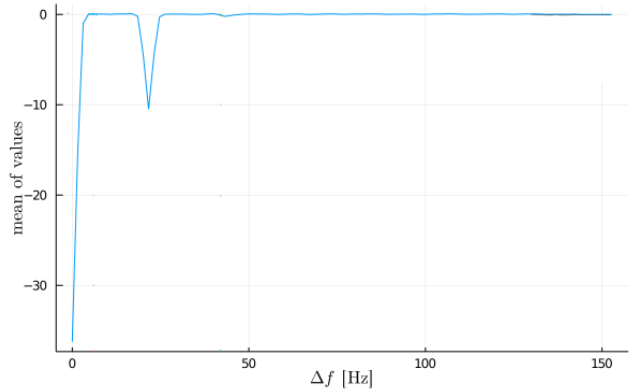
a) S^4 of Gaussian noise, $m = 2$



b) S^4 of coherent radiation, $m = 2$



c) means of diagonals from the S^4 in **b)**



d) Magnified part of S^4 of coherent radiation

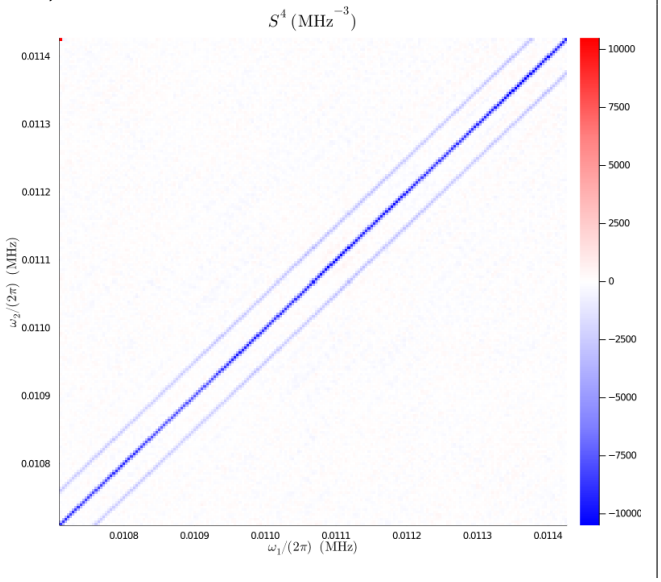


Table 10: In the full correlation spectra **a)** and **b)**, the main diagonals appear as dashed lines of anti-correlation.
In **c)**, the means of the first 150 secondary diagonals are shown, exhibiting clear dips at 0 Hz and 20-25 Hz.

The smaller anticorrelations next to the main diagonal could be estimated to be about $\Delta f = 2.5 * 10^{-5}$ MHz away from the main diagonal, which is again very close to the pulse frequency of 10^{-5} MHz (and even closer to $1/T_\mathrm{p} \cdot T_\mathrm{win}/T_\mathrm{p}$. Such a phenomenon has been predicted by Daniel Hägele (private communication) and, if found to provide reliable estimates of the latter in the future, could be helpful

also when searching for pulsars.

It has been found that these lines of anticorrelation only become prominently visible for small $m = 2$ (not for $m = 32$) and for shorter DFT-windows ($T_{\mathrm{win}} = 2.8 \cdot T_{\mathrm{p}}$). Moreover, the secondary line next to the main diagonal can move depending on $T_{\mathrm{win}}/T_{\mathrm{p}}$. Both of this means that it must be an artefact of the estimator (or its implementation in this case) rather than a real contribution to the S^4 from the signal (private communication with Daniel Hägele), which implies that it will not be helpful for studying the coherency of pulsar radiation.

However, besides being an artefact, this looks like a chance to roughly measure the pulse period of a pulsar without free parameters that have to be guessed, like most notably the dispersion measure. This means that it could potentially pay off to further explore this artefact and try to pinpoint the exact relation between its location, the pulse period and the DFT-settings in the future.

6 Discussion and Outlook

To briefly answer the question posed in section 1.4, yes, higher-order spectra of pulsed (broadband) "coherent" radiation can look different from those of absolutely noisy pulses. What could be seen consistently was **a)** a subtle checkerboard-pattern appearing in the correlation spectra from broadband coherent radiation and **b)** no visible structures in the higher-order spectra of pulsed Gaussian noise. In addition, the range of values of the higher-order spectra differed often somewhat and sometimes drastically, leaving room for more rigorous numerical analysis.

Two model parameters made a difference for the apperance of the spectra. The stronger one was the coherence length (or coherence time) of the signal, which became clear when a ten times shorter coherence time was tested. Then, the checkerboard pattern almost disappeared, which suggests that the correlation spectrum could in principle provide a measure for coherence times. Less obvious was the possible strengthening of the pattern when setting the pulse period roughly 10 times longer than the DFT-windows.

However, no visually perceptible changes where spotted when changing other analysis settings, like the length of the DFT or avoiding the exact match-up of coherence time und pulse period.

That said, the typical signatures of coherent radiation vs. pulsed Gaussian noise did not differ very strongly in any case. That makes it unclear if they could even be seen with real pulsar data, which is subject to big levels of background noise, whereas all signals analized here where practically devoid of it. But the fact that there is some difference visible indicates that it might be worthwile to experiment with more realistic signals that may include a noise background and find out under which circumstances the differences between the higher-order spectra of both radiation types are emphasized. As said before, the checkerboard-pattern became weaker with shorter coherence time and became stronger with a longer pulse period. This suggests that it could be advantageous to have short DFT-windows in comparison to the coherence time respectively the pulse period.

This is problematic, because the state of the art-estimators used here only produce correct results when the coherence time scale is considerably shorter than the DFT-windows. So ideally, one would want to use better estimators that also work for very coherent signals, or, alternatively, find a trick to manipulate the signal so that the known estimators work again (private communication with Daniel Hägele and Markus Sifft). In addition, it is absolutely unclear what the timescales for the coherence of real pulsar radiation are, so anything calculated from real data with the estimators in use here could be wrong just because the pulsar is too coherent.

Even in this work, the signals might not have been ideally suited for use with the mentioned estimators, because the strict periodicity of the pulses represents some kind of coherence and could make them

incompatible (private communication with Daniel Hägele). While working on this topic, the need for estimators that are more generally applicable became more and more clear, which was assisted by preliminary results of this work. This is already a contribution to the scientific process, and may be of value beyond pulsar astronomy, because such estimators could also be used to study certain quantum-mechanical systems and short signals from gravitational wave detectors (private communication with Daniel Hägele).

However, there is one possibility found here that even benefits from an invalid estimator (or a wrong implementation), and that is the line of anticorrelation with constant distance to the main diagonal of the correlation spectrum, possibly related to the pulse period. This could be examined more rigorously, maybe with the goal of a quick estimation of the dispersion measure (DM), which could be an aid when searching for pulsars through surveys. For now, only a straight line has been found, but it might be possible that this line becomes curved when the signal is dispersed proportional to $1/f^2$, thus revealing information about the DM. In this context, other general indications of pulsed emission could also be considered. The only thing found until now is the slight hint of a positive offset from section 5.2, but maybe there is something else indicating the presence of a pulsar.

In conclusion, it is probably more urgent to find a new class of estimators for the higher-order spectra that can handle very coherent signals than to refine the model of the signal, to process data of real pulsar observations or to analyse the higher-order spectra from this work more precisely. If one wants to work with (simulated) pulsar data before more suitable estimators are available, the artefact of the anticorrelated secondary diagonal may provide an opportunity for this.

Appendix A: Calculation of S^4 shown with matrices

Calculation for **part 1** $\frac{m+1}{m-1}\left(\overline{a_k a_k^* a_l a_l^*}\right)$ shown with actual matrices:

$$\mathbf{C}_1 = p_{\mathrm{t}1} \cdot (\mathbf{D}_{\mathrm{abs}2} \times \mathbf{D}_{\mathrm{abs}2}^T) \tag{50}$$

$$= \frac{m+1}{m(m-1)} \cdot \begin{bmatrix} |a_{11}|^2 & |a_{12}|^2 & \cdots & |a_{1M}|^2 \\ |a_{21}|^2 & |a_{22}|^2 & \cdots & |a_{2M}|^2 \\ \vdots & \vdots & \ddots & \vdots \\ |a_{N_{\mathrm{rfft}}1}|^2 & |a_{N_{\mathrm{rfft}}2}|^2 & \cdots & |a_{N_{\mathrm{rfft}}M}|^2 \end{bmatrix} \times \begin{bmatrix} |a_{11}|^2 & |a_{21}|^2 & \cdots & |a_{N_{\mathrm{rfft}}1}|^2 \\ |a_{12}|^2 & |a_{22}|^2 & \cdots & |a_{N_{\mathrm{rfft}}2}|^2 \\ \vdots & \vdots & \ddots & \vdots \\ a_{1M}|^2 & |a_{2M}|^2 & \cdots & |a_{N_{\mathrm{rfft}}M}|^2 \end{bmatrix} \tag{51}$$

$$= \left(\frac{m+1}{m-1}\right) \cdot \left(\frac{1}{m}\right) \cdot \begin{bmatrix} \sum_{j=1}^M |a_{1j}|^2|a_{1j}|^2 & \sum_{j=1}^M |a_{1j}|^2|a_{2j}|^2 & \cdots & \sum_{j=1}^M |a_{1j}|^2|a_{N_{\mathrm{rfft}}j}|^2 \\ \sum_{j=1}^M |a_{2j}|^2|a_{1j}|^2 & \sum_{j=1}^M |a_{2j}|^2|a_{2j}|^2 & \cdots & \sum_{j=1}^M |a_{2j}|^2|a_{N_{\mathrm{rfft}}j}|^2 \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{j=1}^M |a_{N_{\mathrm{rfft}}j}|^2|a_{1j}|^2 & \sum_{j=1}^M |a_{N_{\mathrm{rfft}}j}|^2|a_{2j}|^2 & \cdots & \sum_{j=1}^M |a_{N_{\mathrm{rfft}}j}|^2|a_{N_{\mathrm{rfft}}j}|^2 \end{bmatrix} \tag{52}$$

$$= \left(\frac{m+1}{m-1}\right) \cdot \begin{bmatrix} \overline{|a_1|^2|a_1|^2} & \overline{|a_1|^2|a_2|^2} & \cdots & \overline{|a_1|^2|a_{N_{\mathrm{rfft}}}|^2} \\ \overline{|a_2|^2|a_1|^2} & \overline{|a_2|^2|a_2|^2} & \cdots & \overline{|a_2|^2|a_{N_{\mathrm{rfft}}}|^2} \\ \vdots & \vdots & \ddots & \vdots \\ \overline{|a_{N_{\mathrm{rfft}}}|^2|a_1|^2} & \overline{|a_{N_{\mathrm{rfft}}}|^2|a_2|^2} & \cdots & \overline{|a_{N_{\mathrm{rfft}}}|^2|a_{N_{\mathrm{rfft}}}|^2} \end{bmatrix} \tag{53}$$

Calculation for **part 2** $\frac{m}{m-1}\left(\overline{a_k a_k^*}\ \overline{a_l a_l^*}\right)$ shown with actual matrices:

$$\mathbf{C}_2 = p_{\mathrm{t}2} \ \cdot \ \left(\mathbf{d}_{\mathrm{abs}2} \otimes \mathbf{d}_{\mathrm{abs}2}^T\right) \tag{54}$$

$$= \frac{m}{m^2(m-1)} \ \cdot \begin{bmatrix} \sum_{j=1}^M |a_{1j}|^2 \\ \sum_{j=1}^M |a_{2j}|^2 \\ \vdots \\ \sum_{j=1}^M |a_{N_{\mathrm{rfft}}j}|^2 \end{bmatrix} \otimes \begin{bmatrix} \sum_{j=1}^M |a_{1j}|^2 & \sum_{j=1}^M |a_{2j}|^2 & \cdots & \sum_{j=1}^M |a_{N_{\mathrm{rfft}}j}|^2 \end{bmatrix} = \left(\frac{m}{m-1}\right) \cdot \left(\frac{1}{m^2}\right) \cdot \tag{55}$$

$$\begin{bmatrix} \left(\sum_{j=1}^M |a_{1j}|^2\right)^2 & \sum_{j=1}^M |a_{2j}|^2 \sum_{j=1}^M |a_{1j}|^2 & \cdots & \sum_{j=1}^M |a_{N_{\mathrm{rfft}}j}|^2 \sum_{j=1}^M |a_{1j}|^2 \\ \sum_{j=1}^M |a_{1j}|^2 \sum_{j=1}^M |a_{2j}|^2 & \left(\sum_{j=1}^M |a_{2j}|^2\right)^2 & \cdots & \sum_{j=1}^M |a_{N_{\mathrm{rfft}}j}|^2 \sum_{j=1}^M |a_{2j}|^2 \\ \vdots & \cdots & \ddots & \vdots \\ \sum_{j=1}^M |a_{1j}|^2 \sum_{j=1}^M |a_{N_{\mathrm{rfft}}j}|^2 & \sum_{j=1}^M |a_{2j}|^2 \sum_{j=1}^M |a_{N_{\mathrm{rfft}}j}|^2 & \cdots & \left(\sum_{j=1}^M |a_{N_{\mathrm{rfft}}j}|^2\right)^2 \end{bmatrix} \tag{56}$$

$$= \left(\frac{m}{m-1}\right) \ \cdot \begin{bmatrix} \left(\overline{|a_1|^2}\right)^2 & \overline{|a_2|^2}\ \overline{|a_1|^2} & \cdots & \overline{|a_{N_{\mathrm{rfft}}}|^2}\ \overline{|a_1|^2} \\ \overline{|a_1|^2}\ \overline{|a_2|^2} & \left(\overline{|a_2|^2}\right)^2 & \cdots & \overline{|a_{N_{\mathrm{rfft}}}|^2}\ \overline{|a_2|^2} \\ \vdots & \cdots & \ddots & \vdots \\ \overline{|a_1|^2}\ \overline{|a_{N_{\mathrm{rfft}}}|^2} & \overline{|a_2|^2}\ \overline{|a_{N_{\mathrm{rfft}}}|^2} & \cdots & \left(\overline{|a_{N_{\mathrm{rfft}}}|^2}\right)^2 \end{bmatrix} \tag{57}$$

The calculation for **part 3** $\left(\frac{m}{m-1}\left(\overline{a_k a_k^*}\ \overline{a_l a_l^*}\right)\right)$ works very similar to the one of **part 1**, except that there are two matrix multiplications between $\mathbf{D}_{\mathrm{rfft}}$ itself and its transposed complex-conjugate $\mathbf{D}_{\mathrm{rfft}}^{*T}$ respectively the complex-conjugate $\mathbf{D}_{\mathrm{rfft}}^*$ and $\mathbf{D}_{\mathrm{rfft}}^T$, of which the results are later multiplied elementwise.

References

[1] Michael Abbott. *Tullio.jl: Github repository and documentation.* URL: `https://github.com/mcabbott/Tullio.jl`.

[2] S. D. Bates, D. R. Lorimer, and J. P. W. Verbiest. "The pulsar spectral index distribution". In: *Monthly Notices of the Royal Astronomical Society* 431.2 (May 2013), pp. 1352–1358. DOI: `10.1093/mnras/stt257`. arXiv: `1302.2053 [astro-ph.SR]`.

[3] Tim Besard, Christophe Foket, and Bjorn De Sutter. "Effective Extensible Programming: Unleashing Julia on GPUs". In: *IEEE Transactions on Parallel and Distributed Systems* 30 (4 2018), pp. 827–841. ISSN: 1045-9219. DOI: `10.1109/TPDS.2018.2872064`. arXiv: `1712.03112 [cs.PL]`.

[4] Jeff Bezanson et al. "Julia: A Fresh Approach to Numerical Computing". In: *SIAM Review* 59 (1 2017), pp. 65–98. DOI: `10.1137/141000671`.

[5] David R. Brillinger. "An Introduction to Polyspectra". In: *The Annals of Mathematical Statistics* 36.5 (1965), pp. 1351–1374. ISSN: 00034851. URL: `http://www.jstor.org/stable/2238424`.

[6] David R. Brillinger. "SOME HISTORY OF THE STUDY OF HIGHER-ORDER MOMENTS AND SPECTRA". In: *Statistica Sinica* 1.2 (1991), pp. 465–476. ISSN: 10170405, 19968507. URL: `http://www.jstor.org/stable/24304021`.

[7] W. Coles et al. "Pulsar timing analysis in the presence of correlated noise". In: 418.1 (Nov. 2011), pp. 561–570. DOI: `10.1111/j.1365-2966.2011.19505.x`. arXiv: `1107.5366 [astro-ph.IM]`.

[8] The Fermi LAT Collaboration. "An extremely bright gamma-ray pulsar in the Large Magellanic Cloud". In: *Science* 350.6262 (2015), pp. 801–805. ISSN: 0036-8075. DOI: `10.1126/science.aac7400`. eprint: `https://science.sciencemag.org/content/350/6262/801.full.pdf`. URL: `https://science.sciencemag.org/content/350/6262/801`.

[9] K. Deergha Rao and M. N. S. Swamy. *Digital Signal Processing - Theory and Practice*. Springer, Singapore, 2018. DOI: `10.1007/978-981-10-8081-4`.

[10] Chris Elrod, Dilum Aluthge, et al. *Octavian.jl: Github repository and documentation.* URL: `https://github.com/JuliaLinearAlgebra/Octavian.jl`.

[11] Chris Elrod et al. *LoopVectorization.jl: Github repository and documentation.* URL: `https://github.com/JuliaSIMD/LoopVectorization.jl`.

[12] Matteo Frigo and Steven G. Johnson. "The Design and Implementation of FFTW3". In: *Proceedings of the IEEE* 93.2 (2005). Special issue on "Program Generation, Optimization, and Platform Adaptation", pp. 216–231.

[13] Douglas C. Giancoli. *Physik Lehr- und Übungsbuch*. Pearson Deutschland, 2010, p. 1610. ISBN: 9783868940237.

[14] F.J. Harris. "On the use of windows for harmonic analysis with the discrete Fourier transform". In: *Proceedings of the IEEE* 66.1 (1978), pp. 51–83. DOI: `10.1109/PROC.1978.10837`.

[15] Daniel Hägele and Fabian Schefczik. "Higher-order moments, cumulants, and spectra of continuous quantum noise measurements". In: *Physical Review B* 98.20 (2018). ISSN: 2469-9969. DOI: `10.1103/physrevb.98.205143`. URL: `http://dx.doi.org/10.1103/PhysRevB.98.205143`.

[16] Michael Kramer. "Separation of integrated pulse shapes into unique sub-components". In: *International Astronomical Union Colloquium* 160 (1996), 215–216. DOI: `10.1017/S0252921100041518`.

[17] Ben Lauwens and Allen B. Downey. *Think Julia*. O'Reilly Media, Inc., 2019. ISBN: 9781492045038.

[18] D. R. Lorimer and M. Kramer. *Handbook of Pulsar Astronomy*. Cambridge University Press, 2004, A312. ISBN: 9783662517321.

[19] R. N. Manchester et al. "The Australia Telescope National Facility Pulsar Catalogue". In: *Astronomical Journal* 129.4 (Apr. 2005), pp. 1993–2006. DOI: 10.1086/428488. arXiv: astro-ph/0412641 [astro-ph].

[20] D. Melrose. "Coherent emission mechanisms in astrophysical plasmas". In: *Reviews of Modern Plasma Physics* 1 (5 2017). DOI: 10.1007/s41614-017-0007-0.

[21] D. B. Melrose. "Coherent emission". In: *Universal Heliophysical ProcessesProceedings IAU Symposium*. 257. International Astronomical Union. 2008.

[22] *NASA: Featured Image of the day, May 6th, 2020*. URL: https://www.nasa.gov/image-feature/the-crab-nebula-observations-through-time.

[23] *Pulsar Catalogue of the Australian National Telescope Facility*. URL: https://www.atnf.csiro.au/research/pulsar/psrcat/.

[24] M. B. Purver. "High-precision pulsar timing : the stability of integrated pulse profiles and their representation by analytic templates". PhD thesis. The University of Manchester, Faculty of Engineering, Physical Sciences, School of Physics, and Astronomy, 2010. URL: https://www.research.manchester.ac.uk/portal/files/54505886/FULL_TEXT.PDF.

[25] Mutsuo Saito and Makoto Matsumoto. "SIMD-Oriented Fast Mersenne Twister: a 128-bit Pseudorandom Number Generator". In: *Monte Carlo and Quasi-Monte Carlo Methods 2006*. Ed. by Alexander Keller, Stefan Heinrich, and Harald Niederreiter. Springer Berlin Heidelberg, 2008, pp. 607–622. ISBN: 978-3-540-74496-2.

[26] Fabian Schefczik. "Theorie kontinuierlicher Quantenmessungen höherer Ordnung und Realisierung eines GPU-Korrelationsspektrometers". PhD thesis. Fakultät für Physik und Astronomie der Ruhr-Universität Bochum, 2020.

[27] Fabian Schefczik and Daniel Hägele. "Ready-to-Use Unbiased Estimators for Multivariate Cumulants Including One That Outperforms $\overline{x^3}$". In: (2019). arXiv: 1904.12154 [math.ST].

[28] D. M. Scott, M. H. Finger, and C. A. Wilson. "Characterization of the timing noise of the Crab pulsar". In: *Monthly Notices of the Royal Astronomical Society* 344.2 (Sept. 2003), pp. 412–430. DOI: 10.1046/j.1365-8711.2003.06825.x. arXiv: astro-ph/0303582 [astro-ph].

[29] *Section on Random Number Generation in the Julia Standard Library (Julia 1.0)*. URL: https://docs.julialang.org/en/v1/stdlib/Random/.

[30] M. Sifft et al. "Quantum polyspectra for modeling and evaluating quantum transport measurements: A unifying approach to the strong and weak measurement regime". In: *Phys. Rev. Research* 3 (3 2021), p. 033123. DOI: 10.1103/PhysRevResearch.3.033123. URL: https://link.aps.org/doi/10.1103/PhysRevResearch.3.033123.

[31] Sebastian Starosielec. "Rauschspektroskopie höherer Ordnungen". PhD thesis. Fakultät für Physik und Astronomie der Ruhr-Universität Bochum, 2012.

[32] Sebastian Starosielec and Daniel Hägele. "Discrete-time windows with minimal RMS bandwidth for given RMS temporal width". In: *Signal Processing* 102 (Sept. 2014), 240–246. DOI: 10.1016/j.sigpro.2014.03.033.

[33] Thomas L. Wilson, Kristen Rohlfs, and Susanne Hüttemeister. *Tools of Radio Astronomy*. Astronomy and Astrophysics Library. Springer Verlag Berlin-Heidelberg, 2013. DOI: 10.1007/978-3-642-39950-3.