C++ Beamformer Library with RFI Mitigation Version 0.1.0 – GNU GPL 3.0

Max Planck Institute for Radio Astronomy, Jan Wagner

Developed for EU FP7 ALBiUS WP6 D6.3.1 “RFI Mitigation”
Document Version 24. November 2011

1 General

This general purpose C++ library implements beamforming and RFI mitigation in the sense of interferer suppression and signal recovery. Library functions can be applied to short time integrated cross-correlation data (STI data) of multi-pixel receivers, focal plane or phased arrays and interferometers.

The library was primarily developed for radio astronomy applications under FP7 ALBiUS. It is publicly released in the hope it may prove to be useful in other applications as well. Accelerated BLAS/LAPACK linear algebra routines are used for performance. Single-core throughput on a Xeon E5430 with 64 antenna element data, double precision and complex arithmetic ranges from 100 to 9000 channels per second, depending on the type of RFI mitigation and beamforming.

2 Contents

3 Introduction

Radio frequency interference (RFI) in low frequency bands is a growing concern in radio astronomy. Research in Digital Signal Processing and Advanced Radio Communications has produced a wealth of interference mitigation methods that are widely used in military and commercial communication technology and are usually tailored for certain scenarios and radio environments.

Mitigation methods have found their way into radio astronomy, too. Thanks to advances in the performance of FPGAs and computers, digital instead of analog processing approaches have gained entrance into the early layers of radio astronomic data capture and data preprocessing. Here they improve the data quality at varying degrees of success.

For cases where signal recovery is not thought to be possible, there exist statistical methods (flaggers) that analyze time series of multi-channel data and identify parts affected by interference. Data of identified parts is then discarded during post-processing.

On the other hand, there exist also methods that aim to recover as much of the desired celestial signal as possible. Promising methods in radio astronomy include real-time adaptive filtering, adaptive beamforming, spatial filtering, subtraction of actual or reconstructed interfering signals with the help of reference antennas, and filtering of complex visibilities amongst others.

This C++ library together with Matlab reference source code is intended for certain observation setups that provide the required additional information which allows recovery of the desired signal to a higher degree, while not harming the data in those bands that are free of interference.

General requirements and a software details as well as results are given below.

4 C++ Library Requirements and Compiling

Source code is provided with this package. Code is maintained in the DiFX SVN repository (svn co https://svn.atnf.csiro.au/difx) and is in the ./libraries/beamformer/trunk/ path.

The C++ library has been tested to compile at least under **Linux** and **GCC 4.4.4** and GCC 4.5.1. You also need **autoconf, automake, libtool**.

For good performance, linear algebra operations use standard accelerated linear algebra. You need to install **Armadillo C++ Linear Algebra Library version 2.2.1 or later** (http://arma.sourceforge.net). Armadillo supports OS X, Windows and Linux and provides compile-time arithmetic expression optimizations. It also interfaces to any underlying library that provides standard BLAS/LAPACK interfaces. These may be ATLAS, Intel MKL or AMD ACML. Under Linux the Armadillo library requires **cmake, blas-devel, lapack-devel, atlas-devel, boost-devel**. You might want to install ATLAS packages that are specific for your system, e.g. **atlas-sse3** and **atlas-sse3-devel** instead of the generic atlas-devel. To install Armadillo under OS X or Windows please read the Armadillo web site.

Unpack the C++ beamformer source code package to some directory. You can compile in single instead of double precision by editing *./src/BeamformerTypeDefs.h* and defining *USE_SINGLE_PRECISION*.

To build and install the Beamformer library including example programs:

```
$ aclocal ; autoconf ; autoheader ; automake -a
$ ./configure --prefix=/usr/local
$ make ; sudo make install
```

5 C++ Library Class Overview

The library is written in C++ and documented with doxygen tags. All classes and member functions are documented both in the source code as well as the doxygen-generated PDF Reference Manual. These are useful for lower level details about the library.

Higher level details of the architecture and the algorithms are described in the current document. A brief summary of classes and what they do is found in Table 1.

The following types of processing are supported, with variations:

Beamforming
Create an *ArrayElements* object to describe antennas and their positions. Create one *Beams_t* structure with electrical pointing angles of all desired beams.
For each new multi-channel *Covariance* data loaded from file or memory, use a *BeamformerWeights* object to compute new element weights using classic beamformer, MVDR or Cox RB-MVDR.
Computed weights can be loaded into e.g. an external GPU beamformer. If raw input data that formed covariances was buffered, weights can be applied to their own data. This reduces error.

Nulling without RFI reference antennas
Load *Covariance* data, pass to a *Decomposition*, run RFI detection and nulling using a *DecompositionModifier*. This modifies data in the Decomposition object. The recompose() methods allow to generate a final cleaned Covariance.
Clean covariances can be useful as the input into external UV plane imaging software.

RFI Templating with reference antennas
Load *Covariance* data, pass it to a *CovarianceModifier* to run RFI Template generation and subtraction.
Clean covariances can be useful as the input into external UV plane imaging software.

Example source code is in the *./examples* directory. The *analysis* program is mainly intended for debugging and comparing data to Matlab. The *benchmark* program executes different processing steps on a single CPU core and reports the performance.

Table 1 - Overview of library classes

Class	Description
ArrayElements	Use this class to describe your focal plane array or antenna array. The class stores information on element positions (X,Y,Z) and the properties of each

	element (LCP, RCP polarizations) and its dedicated use (astronomy signals, or RFI reference antenna for RFI signals). Can pre-generate positions for uniform linear and uniform grid array layouts. Required by: *Beamformer* (RB-MVDR weight calculation) *BeamformerWeights* (conversion of beam angles into steering vectors) *CovarianceModifier* (RFI templating and subtraction)
typedef *Beams_t* (BeamformerData.h)	Used to list the desired electrical beam pointing angle(s). Also the output storage of computed (or "manually" edited) steering vectors and beamformer weights matching these input beam pointing angles. Required by: *BeamformerWeights* (conversion of beam angles into steering vectors) *BeamformerWeights* (weight calculation, joint with Covariance input)
BeamformerWeights	Two use cases. First, helps to convert *Beams_t* electrical beam angles into steering vectors. Second, provides functions to convert steering vectors and covariance matrices or their decompositions into beamformer weights (CBF, MVDR, Cox WNGC MVDR, other methods). Weights are stored back into *Beams_t*.
Covariance	Stores time-integrated covariance matrix data. Data can be single or multi-channel. It can be cross-correlation data (in which case it needs to be frequency domain) or covariance data (in which case it needs to be time-domain with contributing signals X having expectation value E<X>=0). Required by: *CovarianceModifier* (changes to covariance data) *Decomposition* (decompositions or recompositions of covariance data)
CovarianceModifier	Applies non-toxic RFI mitigation algorithms to a Covariance object. Currently it implements four types of RFI Template subtraction. See van der Veen [VE04] and Briggs [BRI00] and section 13 of this document. Requires that: 1) covariance data was observed with $N_{ref} \geq 1$ reference antennas 2) must have $N_{ref} \geq N_{RFI/channel}$, otherwise equation system underdetermined 3) benefits from RFI ≥10dB stronger in reference antennas; mean of RFI autocorrelations 10 larger than times mean of other autocorrelations.
DecompositionAnalyzer	Extracts features from a covariance Decomposition. Currently returns number of RFI signals in a channel, estimated from eigenvalues with MDL or AIC information criteria or 3-sigma thresholding. (Direction of arrival DOA estimation with MUSIC in C++ is TODO.)
Decomposition	Base class for decomposing a 2D covariance matrix or a 3D multi-channel Covariance object. Can also generate a new Covariance from (possibly modified) decomposition data. Child classes: *SVDecomposition, EVDecomposition, QRDecomposition*

	Required by: *DecompositionAnalyzer* (feature extraction, number of RFI, RFI DOA) *DecompositionModifier* (nulling)
DecompositionModifier	Applies automated changes to a Decomposition object. Currently editing steps are RFI interferer estimation and nulling (subspace method).

6 C++ Library and Multithreading

Multi-threading is not directly implemented in this C++ library. Basic time-division parallelism is of course possible, if you handle Covariances of different short time integration intervals on different CPU cores.

However, data channel-division parallelism is also possible. All data processing in the library is memory-in-place. Channels are processed one at a time. Those library functions that have high arithmetic cost, such as covariance data decomposition, can be invoked for just a sub-range of frequency channels.

You can thus use Parallel For on for example *CovarianceDecomposition::decompose()* and loop it over non-overlapping channel ranges, to utilize all CPU cores.

Parallel For can be found for example in the Intel Thread Building Blocks (*parallel_for*), OS X Grand Central Dispatch (*dispatch_apply*), OpenMP (*#pragma omp parallel for*), or *Boost.Thread* parallel for.

7 C++ Library Performance

The individual RFI mitigation and beamforming functions were tested in a sequence typical for normal usage in a real-time or off-line astronomic signal processing pipeline. There library source code comes together with a program called *benchmark* under the *examples*. This program was run on a single core of an Intel E5430 2.66 GHz CPU. Performance with double precision arithmetic is shown in Table 2 below. Single precision performance is shown in

Table 3.

Table 2 – Double precision performance with 1 core on dual Intel Xeon E5430 system (12MB L2, 2.66 GHz, quad core). Synthetic data, 64 phased array elements, 64x64 covariance data processed from memory.

$ # Armadillo with ATLAS, Beamformer compiled '-g –O3 -Wall' for double precision (default) $ numactl –physcpubind=0 ./benchmark	
Integrate 64-elem vector into Covariance	80300 channels/sec (better use FPGA or GPU!)
Decomposition -> recomposition (average)	230 channels/sec
SVD -> RFI detect -> null -> recomposition	150 channels/sec
EVD -> RFI detect -> null -> recomposition	230 channels/sec
1-RFI/ch, 2-reference Template subtraction	5420 channels/sec
2-RFI/ch, 2-reference Template subtraction	9100 channels/sec
64-beam classical beamformer	3600 channels/sec
64-beam MVDR (Cox b=1.0)	290 channels/sec
64-beam RB-MVDR (Cox b=1.0+1e-4)	290 channels/sec

Table 3 - Single precision performance with 1 core on dual Intel Xeon E5430 system (12MB L2, 2.66 GHz, quad core). Synthetic data, 64 phased array elements, 64x64 covariance data processed from in memory.

$ # Armadillo with ATLAS, Beamformer compiled '-g –O3 –Wall -DUSE_SINGLE_PRECISION=1' $ numactl –physcpubind=0 ./benchmark	
Integrate 64-elem vector into Covariance	156000 channels/sec (better use FPGA or GPU!)
Decomposition -> recomposition (average)	270 channels/sec
SVD -> RFI detect -> null -> recomposition	190 channels/sec
EVD -> RFI detect -> null -> recomposition	260 channels/sec
1-RFI/ch, 2-reference Template subtraction	8700 channels/sec
2-RFI/ch, 2-reference Template subtraction	21900 channels/sec
64-beam classical beamformer	5850 channels/sec
64-beam MVDR (Cox b=1.0)	390 channels/sec
64-beam RB-MVDR (Cox b=1.0+1e-4)	360 channels/sec

8 Matlab Script Overview

The source code package includes MathWorks Matlab scripts in addition to C++ source code. The Matlab scripts are essentially the reference for the numerical parts of the C++ library. The scripts are included to help you test new algorithms visually.

Matlab file(s)	Function
subspcrfi	Test program that calls some of the functions below
subspcrfi_A	Generates array steering matrix in one direction for all channels.
subspcrfi_AICrank **subspcrfi_MDLrank**	Estimate the number of independent components from a list of matrix eigenvalues; estimates the rank of the original covariance matrix. See information theory text books or Wax-Kailath [WK85].
subspcrfi_MVDR	Beamformer weights from covariance data and a set of beams. Classical, MVDR and RB-MVDR Cox Projection beamforming. For Cox see [CZO87].
subspcrfi_RtoUV	Basic UV gridding. Converts covariance matrix and antenna element positions into UV plane matrix.
subspcrfi_SNR	Beamformer weights from $R_{onsource} – R_{offsource}$ calibration covariances, Maximum SNR weights into steering direction, or conjugate field match.
subspcrfi_doa_MUSIC	Attempts RFI 3D DOA estimation from decomposed covariance, antenna element positions using the MUSIC algorithm.
subspcrfi_elemXYZ	Generates antenna positions (x,y,z) in uniform grid array in (x,y) plane.
subspcrfi_getEV	Eigenvalue decomposition of covariance matrices for all channels.
subspcrfi_getNoises	Estimates antenna noise from covariance matrices. Uses estimator described for example in Ippoliti [IPP05].
subspcrfi_loadRxxFile	Read multi-channel complex covariance data from a file that the C++ Beamformer library Covariance::store() function can generate.
subspcrfi_modelgen	Synthetic covariance data generator. Single-channel, takes a spatial array layout, element noise, estimation error noise, list of signals (2D angles and powers). Outputs covariance. Signals are assumed to be orthogonal and multipathing delays longer than integration time.
subspcrfi_modelgen2	Synthetic covariance data generator that uses RFI reference antennas. Identical to subspcrfi_modelgen, but specified antennas see RFI signals at higher gain and celestial sources at low to zero gain. Signals are assumed to be orthogonal and multipathing delays longer than integration time.
subspcrfi_nulling	Interferer nulling. Takes EVD decomposed multi-channel covariance data, estimates interferers, replaces dominant eigenvalues with mean of noise-space eigenvalues. Assembles “cleaned” output covariance. Uses standard methods, for gentle introduction see Briggs-Kocz [BK05].
subspcrfi_plotArrayResponse	Beamformer weights are converted into an array response over ±90deg phi/theta angles. The array radiation pattern is plotted in 3D.
subspcrfi_plotEVspec	Multi-channel eigenvalue spectrum, plots the dominant eigenvalues.
subspcrfi_steer	Similar to subspcrfi_A but computes steering for only one frequency.
subspcrfi_subtraction	Reference antenna method. Corrects array covariance data by subtracting RFI signal contributions, estimated by covariance between reference antennas and array elements. Methods: van der Veen [VE04], Briggs [BRI00], 2 generic, see section 13.

subspcrfi_test_subtraction	Test program for the subtraction methods.
subspcrfi_writeRxxFile	Write multi-channel complex valued covariance data into a file that the C++ Beamformer library can import.

9 Requirements on the Antenna Array data

Below are the (reasonable) requirements the input data must meet for proper operation of the RFI mitigation and analysis algorithms. The Matlab sources are quite flexible. The C++ library however expects certain additional Covariance matrix layout constraints to work efficiently.

Here are the points to consider while forming the Covariance input and while planning the technical aspects of an astronomic observation:

1. Real-time time-integrated full covariance matrices across array elements should be formed on FPGA (fixed-point) or GPU (floating point) for larger phased arrays due to CPU and I/O limits.
2. Covariances should be time-integrated over short time intervals (STI) of $1\text{ms}<T_{int}<10\text{ms}$. The T_{int} choice is a balance between increased noise at very short T_{int}, and less effective RFI mitigation at very long T_{int} due to multipathing and RFI variability.
3. Covariance matrices should not be singular. They must be time-integrated with $\geq N_{ant}$ samples.
4. If covariances are FX-type cross-correlations (visibilities) they must be frequency domain.
5. If covariances are time domain covariances they must contain time domain data. Signals of all contributing antennas need to have zero mean ($E[X] = 0$). See end of this chapter.
6. RFI reference antennas may be used to improve mitigation. Full covariances between reference and array antennas need to be formed. Reference antennas must be assigned to lowest antenna indices, that is, their data should be in the top left corner of each covariance matrix.
7. Channelizing antenna signals into a large number of narrow frequency channels may be desirable from an RFI perspective, if narrower channels reduce the likelihood that any given channel will contain more than just one RFI source.

The signal flow in an actual observation with a focal plane or phased array and RFI reference antennas is shown in Figure 1. This is merely one possible observational configuration. The figure is repeated here from van der Veen et al. [VE04]. They used a phased array as the RFI reference antenna and steered array beams towards RFI signals, then used the beams to mitigate RFI from the telescope array signals.

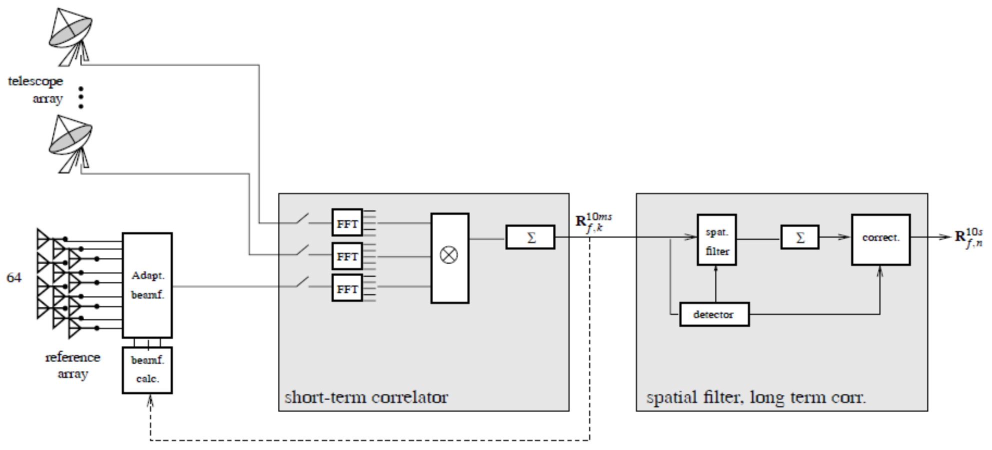


Figure 1 – Suggested processing configuration. Figure from van der Veen et al. [VE04] reinterpreted here. Left block: signal sources and RFI reference signals. Middle block: signal channelization ("FFT") into f channels, channel covariances ("X"), short-term integration ("Σ") of covariances into $R_{f,k}$. Right block: RFI processing such as C++ library reference antenna RFI subtraction or spatial filtering e.g. Nulling of covariances $R_{f,k}$ before long term integration ("Σ") into $R_{f,n}$. The C++ library can also compute RFI-rejecting beam weights ("beamf. calc.").

Data processing begins with FFT channelization (Figure 1, FFT block). This splits each antenna signal into subbands. Frequency domain data of each antenna are cross-multiplied (Figure 1, "X" block) and accumulated over a short time into a matrix (Figure 1, sum block) with one matrix per subband. Because of the high data rates involved, all of these steps should performed on FPGA or GPU.

The C++ beamformer library inputs short-term integrated (STI) covariance data. Processing plugs into the same spot as the "spatial filter, long term correlator" block in Figure 1. The library can compute new beamformer weights, estimate the amount of RFI and output new covariance data cleaned of RFI. The cleaned data can be time-integrated further according to the needs of the astronomer.

The terms "covariance" and "cross-correlation" are used interchangeably. RFI mitigation works identically with both, but we should note the definition difference. For every complex or real-valued signal pair x_i, x_j sampled from random variable processes X_i, X_j with respective means μ_i, μ_j and standard deviations σ_i, σ_j the statistical definition of the covariance is an expectation value of

$$\gamma_{ij} = cov(x_i, x_j) = E\left[(x_i(t) - \mu_i) \cdot \left(x_j(t) - \mu_j\right)^*\right] \tag{9.1}$$

The cross-correlation matrix is a scaled version defined as

$$\rho_{ij} = corr(x_i, x_j) = E\left[(x_i(t) - \mu_i) \cdot \left(x_j(t) - \mu_j\right)^*\right] / \sigma_i \sigma_j = \gamma_{ij} / \sqrt{\gamma_{ii}\gamma_{jj}} \tag{9.2}$$

Statistical cross-correlation is *not* signal processing cross-correlation $(f * g)(t) = \mathcal{F}^{-1}\{\mathcal{F}(f)\mathcal{F}^*(g)\}$. Ignoring statistics, "covariance" may also be called "complex visibility" for Fourier domain x_i, x_j.

For numerical accuracy some RFI mitigation methods require $\hat{\mathrm{C}}_{\mathrm{xx}}$ to be non-singular and invertible, i.e. that it has full rank $\mathrm{N}_{\mathrm{ant}}$. The rank of a matrix is the count of its independent column or row vectors. $\hat{\mathrm{C}}_{\mathrm{xx}}$ has full rank $\mathrm{N}_{\mathrm{ant}}$ iff the count m ($\mathrm{M} \geq m$) of linearly independent snapshots of $x(t)$ added to $\hat{\mathrm{C}}_{\mathrm{xx}}$ is $m \geq \mathrm{N}_{\mathrm{ant}}$. Each new linearly independent snapshot vector added to $\hat{\mathrm{C}}_{\mathrm{xx}}$ increases $\hat{\mathrm{C}}_{\mathrm{xx}}$ rank by 1. You may verify this experimentally with Matlab or NumPy. In the real world, antenna signals $x_i(t)$ all contain some independent noise, provided all antennas are indeed physically present and provide non-zero signal levels. In practice we then have $m = \mathrm{M}$.

For RFI mitigation it is then sufficient to use $\mathrm{M} \geq \mathrm{N}_{\mathrm{ant}}$ snapshots for one time-integrated covariance matrix each. This requirement may be problematic in pulsar observations at very short timescales with a large number of antennas.

10 Details on Subspace Methods

The C++ library and Matlab sources provide methods for interferer nulling. Nulling may involve the construction of a linear projection matrix to project the interferer subspace out of the array covariance matrix $\hat{\mathrm{C}}_{\mathrm{xx}}$. To similar effect one may form an eigendecomposition of $\hat{\mathrm{C}}_{\mathrm{xx}}$ and make certain modifications to that decomposition. Both routes are well-published standard art. A gentle introduction to the latter is given in Briggs et al. section 6 [BK05]. This and related publications do not mention certain caveats. Here we give a terse derivation of the decomposition method and raise several practical issues.

We start with a receiver array with $\mathrm{N}_{\mathrm{ant}}$ antenna elements and a single narrow-band frequency channel. Each element *i* of the array outputs a zero-mean signal $x_i(t); \langle x_i(t) \rangle = 0$ that sums receiver noise $n_i(t)$, some total signal $a_i(t)$ from astronomical sources and the possible interference $i_i(t)$ from all interferers. A snapshot at time *t* groups signals from all $i \in [0, \mathrm{N}_{\mathrm{ant}} - 1]$ antennas into a signal vector $x(t)$,

$$x(t) = \left[x_{i=0}(t), x_{i=1}(t), \dots, x_{i=N_{ant}-1}(t)\right] \; ; \; x_i(t) = n_i(t) + a_i(t) + i_i(t) \tag{10.1}$$

Observed signal vectors $x(t = 0 \dots T)$ convey information about the astronomical sources and about RFI characteristics. To retrieve that information one first forms the array covariance matrix C_{xx} as in (9.1). In practice $x(t)$ is discretely sampled and thus the true underlying C_{xx} can only be approximated. A noisy estimate of C_{xx} produced by averaging M snapshots of x is

$$\hat{\mathrm{C}}_{\mathrm{xx}}(t) = \frac{1}{M} \sum_{i=0}^{M-1} x(t - iT) \cdot x^*(t - iT) \tag{10.2}$$

where T is the sampling interval and x* denotes the complex conjugate transpose of the signal vector. The definition implies that $\hat{\mathrm{C}}_{\mathrm{xx}}(\mathrm{t}): N_{ant} \times N_{ant}$ is Hermitian conjugate symmetric.

The concept of eigendecompositions should be introduced at this point. The idea is that (nearly) any matrix A can be represented as the multiplicative product of three other matrices. Common eigendecompositions are the singular value decomposition (SVD) for general matrices $\mathrm{A}: n \times m$ and the eigenvalue decomposition (EVD) for square $\mathrm{A}: n \times n$ given below:

$$\mathrm{SVD: A} = U \Sigma V^* \quad \text{and} \quad \mathrm{EVD: A} = W \Lambda W^{-1} \tag{10.3}$$

Diagonal matrices $\Sigma: n \times m$ and $\Lambda: n \times n$ contain the scalar eigenvalues λ_i of matrix A. Square $U: n \times n$, $V: m \times m$ contain left and right SVD eigenvectors e_i of A. Columns of square matrix $W: n \times n$ contain right EVD eigenvectors e_i of A. The eigenvectors are orthogonal. Each e_i is associated to one λ_i. In fact, $Ae_i = \lambda_i e_i$, and interpreting A as a coordinate transform matrix, eigenvectors e_i are those special directions e_i that the transform $y = \mathrm{A}e_i$ simply re-scales by factor λ_i but leaves otherwise unchanged.

Some helpful facts:
1) Eigenvectors e_i of $\hat{\mathrm{C}}_{\mathrm{xx}}(\mathrm{t})$ in W are the maximum covariance directions or principal components of A. They are the “spatial footprints” of signals that entered into or originated from inside the array.
2) Projecting $x(t)$ into a principal components basis, $y(t) = Wx(t)$ (on FPGA systems for performance), gives a score vector $y(t)$ where leftmost values are large if RFI is present, low if not. W must be from an earlier eigendecomposition contaminated by the same “footprints” of RFI to be detected. Used to flag or exclude data with sporadic, spatially stationary RFI, but has not yet been applied to radio astronomy.

3) Hermitian A such as $\hat{\mathrm{C}}_{\mathrm{xx}}$ has $\forall i{:}\, \lambda_i \geq 0$ and all λ_i are real: λ_i are not complex and not negative.
4) Hermitian A also implies that A is complex normal so W is unitary i.e. $W^{-1} = W^*$ and reconstruction of A from an EVD decomposition is possible via $W\Lambda W^* = \mathrm{A}$ which is computationally faster than using the (10.3) matrix inverse.

EVD and SVD are closely related. However, EVD has numerical problems for ill-conditioned matrices and some prefer in this case to use SVD which tends to be more stable. Keep this in mind when using the C++ library. We use EVD below, mainly for compact notation.

Returning to RFI mitigation using array covariance data, we make the reasonable assumptions that there is only spatial correlation and that astronomical signal, interferer and array noise signals (subscripts A, I and N respectively) are independent processes that are not correlated during the M-snapshot averaging time, thus $\langle i \cdot a^* \rangle = \langle i \cdot n^* \rangle = \langle n \cdot a^* \rangle = 0$, whereas $\langle n \cdot n^* \rangle \neq 0$ and so on. Now $\hat{\mathrm{C}}_{\mathrm{xx}}$ (10.2) expands to a sum of its independent contributors:

$$x(t) = n(t) + a(t) + i(t)\,, \quad \hat{\mathrm{C}}_{\mathrm{xx}}(t) = \tfrac{1}{M}\textstyle\sum_{i=0}^{M-1} x(t-iT)\cdot x^*(t-iT)$$

$$\xRightarrow{n,a,i\ indep.}\ \hat{\mathrm{C}}_{\mathrm{xx}}(\mathrm{t}) \ \approx\ \hat{\mathrm{C}}_{\mathrm{N(t)}} + \hat{\mathrm{C}}_{\mathrm{A}}(\mathrm{t}) + \hat{\mathrm{C}}_{\mathrm{I}}(\mathrm{t}) \ \approx\ \sigma_{\mathrm{N}}{}^2\mathrm{I} + \hat{\mathrm{C}}_{\mathrm{A}}(\mathrm{t}) + \hat{\mathrm{C}}_{\mathrm{I}}(\mathrm{t}) \qquad (10.4)$$

Further assumptions are regarding signal powers: 1) array elements *i* follow an identical noise distribution with powers $\sigma_{\mathrm{N,i}}{}^2$ and for simplicity can assume $\forall \mathrm{i}{:}\, \sigma_{\mathrm{N,i}}{}^2 = \sigma_{\mathrm{N}}{}^2$, and 2) the *q* interferers are above noise ($\sigma_{\mathrm{I,q}}{}^2 \gg \sigma_{\mathrm{N}}{}^2$), and 3) the astronomical signal is below noise ($\sigma_{\mathrm{A}}{}^2 \ll \sigma_{\mathrm{N}}{}^2$).

The interference term $\hat{\mathrm{C}}_{\mathrm{I}}(\mathrm{t})$ in (10.4) is not known a priori and is already lumped into $\hat{\mathrm{C}}_{\mathrm{xx}}$, hence a simple subtraction from the already contaminated measurement of $\hat{\mathrm{C}}_{\mathrm{xx}}$ is not viable for removing RFI. However, an eigendecomposition is typically able to separate the RFI correlation patterns (eigenvectors) within $\hat{\mathrm{C}}_{\mathrm{xx}}$ based on the distinctly higher power levels of the RFI signals versus the noise signals.

Interferers are orthogonal $\hat{\mathrm{C}}_{\mathrm{xx}}$ eigenvectors when they are uncorrelated, and separable from noise when (and only if) interferer count $q < \mathrm{N_{ant}}$. Writing $\hat{\mathrm{C}}_{\mathrm{xx}}$ (10.4) in an EVD decomposed form:

$$\hat{\mathrm{C}}_{\mathrm{xx}} = \hat{\mathrm{C}}_{\mathrm{A}} + \hat{\mathrm{C}}_{\mathrm{I}} + \sigma_{\mathrm{N}}{}^2\mathrm{I} = W(\Lambda_A + \Lambda_I + \Lambda_N)W^* = W\Lambda W^* = W\begin{bmatrix}\Lambda_{00} & 0\\ 0 & \Lambda_{11}\end{bmatrix}W^* \qquad (10.5)$$

Matrix Λ contains eigenvalues in non-increasing order. It can be partitioned into two submatrices with interferer powers in Λ_{00} and noise powers in Λ_{11}, neglecting the weak $\hat{\mathrm{C}}_{\mathrm{A}}$ power contribution $\sigma_{\mathrm{A}}{}^2$:

$$\Lambda_{00}{:}\,(q\times q) \approx \begin{bmatrix} \acute{\sigma}_{\mathrm{I,0}}{}^2 + \acute{\sigma}_{\mathrm{N,0}}{}^2 & 0 & \cdots & 0\\ 0 & \acute{\sigma}_{\mathrm{I,1}}{}^2 + \acute{\sigma}_{\mathrm{N,1}}{}^2 & \cdots & 0\\ \vdots & \vdots & \ddots & 0\\ 0 & 0 & \cdots & \acute{\sigma}_{\mathrm{I,q-1}}{}^2 + \acute{\sigma}_{\mathrm{N,q-1}}{}^2 \end{bmatrix}$$

$$\Lambda_{11}\colon (N_{ant} - q) \times (N_{ant} - q) \approx \begin{bmatrix} \acute{\sigma}_{N,q}{}^2 & 0 & \cdots & 0 \\ 0 & \acute{\sigma}_{N,q+1}{}^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \acute{\sigma}_{N,N_{ant-1}} \end{bmatrix} \tag{10.6}$$

Submatrix Λ_{00} of Λ is associated with the $\hat{C}_{xx}$ interferer subspace and Λ_{11} with the noise subspace, spanned by RFI and noise eigenvectors respectively. Λ_{00}, Λ_{11} are asymptotically correct. However, due to short time averaging or non-stationary interferers, there is intrinsic noise in the $\hat{C}_{xx}$ estimate that also leaks into Λ_{00} and Λ_{11} and causes eigenvalues to deviate slightly from the total powers stated above.

Now consider an RFI-free environment and a faint source. In this case $\hat{C}_{xx}$ (10.4) equals the array noise covariance $\hat{C}_{xx} = \sigma_N{}^2 I$, hence Λ_{00} is empty, whereas the $\hat{C}_{xx}$ noise subspace has full rank and Λ_{11} contains all N_{ant} eigenvalues $\lambda_{0 \le j < N_{ant-1}} = \acute{\sigma}_{N,j}{}^2$ where $\acute{\sigma}_{N,j}{}^2$ is the j:th greatest antenna noise power.

Introducing RFI, submatrix Λ_{00} becomes non-empty and grows proportionally to interferers *q* added to the environment. Noise subspace eigenvalues Λ_{11} are $\lambda_{q \le j < N_{ant-1}} = \acute{\sigma}_{N,j}{}^2$ as in the RFI-free case, but interferer subspace eigenvalues Λ_{00} are $\lambda_{0 \le j < q} \approx \acute{\sigma}_{I,j}{}^2 + \acute{\sigma}_{N,j}{}^2$ which equals the total power impinging on all array elements from the j:th strongest out of *q* interferers, overlaid with antenna noise.

The effect of multipathing is that it increases the apparent number of independent interferer signals *q*, provided that the averaging time for $\hat{C}_{xx}$ is shorter than multipathing delays, ensuring no self-correlation.

As an aid to the eigendecomposition in general, consider two example cases.

- Case 1: uncorrelated equal noise $\sigma_N{}^2$ in all array elements, $N_{ant} > 3$ array elements in total. Three elements see a different interferer each, total *q*=3, with RFI powers $\sigma_{I,0}{}^2$, $\sigma_{I,1}{}^2$ and $\sigma_{I,2}{}^2$. Output signals $x_i(t); i = [0,2]$ of the 3 array elements are fully uncorrelated.
  => EVD of $\hat{C}_{xx}$ has exactly 3 interferer eigenvalues in Λ_{00}: $\lambda_0 = \sigma_N{}^2 + \sigma_{I,0}{}^2, \lambda_1 = \sigma_N{}^2 + \sigma_{I,1}{}^2$, $\lambda_2 = \sigma_N{}^2 + \sigma_{I,2}{}^2$. Noise eigenvalues in Λ_{11} are $\lambda_{j \ge 3} = \sigma_N{}^2$. Eigenvectors in W are identical to the Euclidean space standard basis ($[1\ 0\ 0 \ldots]$, $[0\ 1\ 0 \ldots]$ etc) for N_{ant} dimensions.
- Case 2: uncorrelated equal noise $\sigma_N{}^2$ in all array elements. One interferer, total *q*=1, is visible to 3 elements at a power $\sigma_I{}^2$ each. Output signals $x_i(t); i = [0,2]$ of the 3 elements are assumed to be fully correlated for the interferer part, but uncorrelated for the noise part.
  => EVD of $\hat{C}_{xx}$ has Λ_{00} with $\lambda_0 = \sigma_N{}^2 + 3\sigma_I{}^2$. Remaining Λ_{11} are $\lambda_{j \ge 1} = \sigma_N{}^2$. All eigenvectors are identical to the Euclidean space standard basis, except three that are tilted. These are a “spatial" footprint of the interferer. Values depend on array geometry and incidence angle.

Interferer count *q* is usually unknown. It could be derived from $\hat{C}_{xx}$ eigenvalues, simply counting all $\lambda_j \ne \sigma_N{}^2$. This is complicated by $\hat{C}_{xx}$ being a noisy estimate. In the end, *q* can only be estimated.

Estimating *q* is further complicated by non-equal array element noise characteristics, i.e. $\sigma_{N,i}{}^2 \ne \sigma_N{}^2$. We should explain here the quiet change of notation from σ^2 to $\acute{\sigma}^2$ in eigenvalue submatrices (10.6): all RFI publications consider only a trivial array with equal element noise, $\forall i\colon \sigma_{N,i}{}^2 = \sigma_N{}^2$. Then a scatter plot of the N_{ant}-dimensional $x(t)$ (10.1) is exactly spherical. Introducing RFI disturbs the plot towards *q*

distinct directions. These form the principal component axes of the scatter. You may refer to Principal Component Analysis (PCA) for details. Eigenvalue solution (10.6) will have $\acute{\sigma}_{\mathrm{N,j}}{}^2 = \sigma_{\mathrm{N}}{}^2,\ \acute{\sigma}_{\mathrm{I,j}}{}^2 = \sigma_{\mathrm{I,j}}{}^2$.
In a realistic antenna array however $\exists i, j\colon \sigma_{\mathrm{N,i}}{}^2 \neq \sigma_{\mathrm{N,j}}{}^2$ and correspondingly a scatter plot of $x(t)$ will be ellipsoidal. Adding *q* RFI signals stretches the ellipsoid surface and introduces new principal components tilted towards the already noise-induced asymmetries. Hence eigenvalue solution (10.6), while still exact, will have one or more $\acute{\sigma}_{\mathrm{N,j}}{}^2$ for which $\nexists k\colon \acute{\sigma}_{\mathrm{N,j}}{}^2 = \sigma_{\mathrm{N,k}}{}^2$ and for which instead $\acute{\sigma}_{\mathrm{N,j}}{}^2$, $\acute{\sigma}_{\mathrm{I,j}}{}^2$ are "biased" mixtures of pure noise powers $\sigma_{\mathrm{N,i}}{}^2$ and RFI power. This complicates determining an optimal RFI-removing subspace projection or a RFI-nulling replacement eigenvalue, discussed farther below.

A robust estimate of interferer count *q* is the Minimum Description Length (MDL, Rissanen 1978). For a set of models, MDL computes the likelihood *L* for the observed data given a model, penalized by the length of that model. This indicates the lowest model order *q* that describes the salient features of the data. For RFI detection we use eigenvalue matrix Λ as the data and *q* (*k*) as the model order.

$$q = \arg \min_{k \in [0, \mathrm{N_{ant}} - 1]} L(\Lambda | k) \tag{10.7}$$

The C++ library uses $L(\Lambda|k)$ with a ratio of the geometric to the arithmetic mean for the *k* largest eigenvalues in Λ. This determines the number of signals in Gaussian noise. For details see [WK85].

$$L(\Lambda|k) = -M(\mathrm{N_{ant}} - \mathrm{k}) \cdot \ln \frac{\left(\prod_{i=k}^{\mathrm{N_{ant}}-1} \lambda_i\right)^{\frac{1}{\mathrm{N_{ant}}-\mathrm{k}}}}{\frac{1}{\mathrm{N_{ant}}-\mathrm{k}}\left(\sum_{i=k}^{\mathrm{N_{ant}}-1} \lambda_i\right)} + \frac{1}{2} k(2\mathrm{N_{ant}} - \mathrm{k} + 1) \cdot \ln M \tag{10.8}$$

The estimated *q* is most reliable (least likely to be off-by-one) when the RFI to noise ratio is $\sigma_{\mathrm{I}}{}^2/\sigma_{\mathrm{N}}{}^2 \gg 1$.

Other methods to estimate *q* are: 1) use Akaike Information Criteria (AIC), but carefully, as AIC is easily off-by-one when array $\mathrm{N_{ant}}$ is large, or 2) count eigenvalues exceeding three standard deviations ("3σ": $\lambda_i > 3\sigma_\Lambda$) where σ is derived using all eigenvalues in Λ including outliers, or more robustly 3) count eigenvalues exceeding three median absolute deviations ("3MAD": $\lambda_i > 3 \cdot median(|\Lambda - median(\Lambda)|)$). Not implemented in the C++ library is the heuristic option 4) traversing sorted eigenvalue list along index *k* and returning *q*=*k* when data curvature within a sliding window exceeds a threshold.

Figure 2 and Figure 3 show two sets of eigenvalues: with and without RFI, including MDL detection (10.8). Data is from APERTIF and courtesy of Wim van Capellen. APERTIF is a stackable focal plane array. One tile adds ~32 dual polarization signal pairs. Each Vivaldi antenna covers non-overlapping 3°x3°. One tile was available in a telescope of the Westerbork array. A LOFAR back-end formed 71 frequency channels and their 64x64 array covariance matrices. Figure 2 shows sorted eigenvalues of channel #1 covariance with no RFI (*q*=0). Note the non-uniform array noise powers and five unconnected array elements. Figure 3 shows covariance eigenvalues of another channel with one strong and one weak RFI interferer (*q*=2).

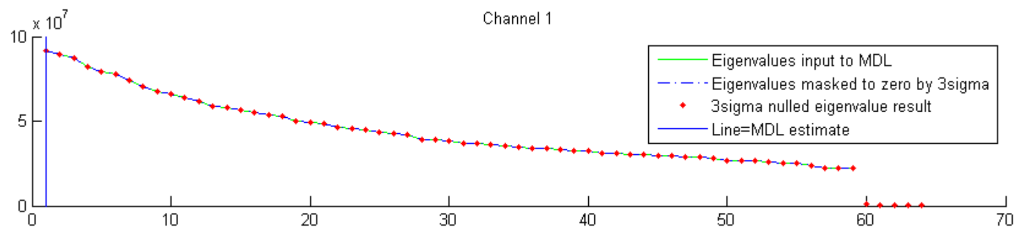


Figure 2 – Eigenvalues input to and output by RFI detectors. Data from Virgo A at 1.4830 GHz, APERTIF channel #1, 64x64 covariance. No RFI. Trailing zeroes due to 5 unconnected elements. Input to MDL detection (solid green) uses non-zero eigenvalues. MDL-estimate of *q* (vertical line) is *q*=0, identical to 3-sigma *q* estimate. Output (red dots) is identical to input.

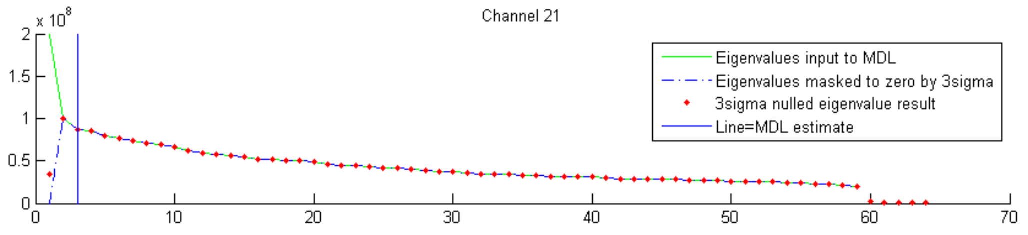


Figure 3 - Eigenvalues input to and output by RFI detectors. Data from Virgo A at 1.4830 GHz, APERTIF channel #21 64x64 covariance. Two interferers. Trailing zeroes due to 5 unconnected elements. Input to MDL detection (solid green) exhibits knee point at left of 3rd eigenvalue, MDL-estimated *q* (vertical line, exclusive) is *q*=2. The 3-sigma (dashed blue) detects *q*=1. Output (red dots) replaces flagged values with median of non-flagged values for later construction of a cleaned covariance.

After estimating *q*, the interferer subspace eigenvalues matrix $\tilde{\Lambda}_{00}: q \times q$ could be set to $\tilde{\Lambda}_{00} = 0$ ("Nulling") while retaining noise eigenvalues $\tilde{\Lambda}_{11} = \Lambda_{11}$. The RFI-cleaned new covariance matrix is:

$$\tilde{\mathrm{C}}_{\mathrm{xx}} = W_{xx} \begin{bmatrix} \tilde{\Lambda}_{00} & 0 \\ 0 & \Lambda_{11} \end{bmatrix} W_{xx}{}^{*} = W_{xx} \tilde{\Lambda}_{xx} W_{xx}{}^{*} \tag{10.9}$$

Choosing $\tilde{\Lambda}_{00} = 0$ ignores the noise contribution $\acute{\sigma}_{\mathrm{N,j}}{}^{2}$ in $\acute{\sigma}_{\mathrm{I,j}}{}^{2} + \acute{\sigma}_{\mathrm{N,j}}{}^{2}$ (10.6) and biases $\tilde{\mathrm{C}}_{\mathrm{xx}}$ by suppressing both noise, source and RFI. The traditional $\tilde{\Lambda}_{00}$ choice attempts to estimate $\acute{\sigma}_{\mathrm{N,j}}{}^{2}$ via the mean of the Λ_{11} noise power values. Using median is more robust. These are available in the C++ library.

$$\tilde{\Lambda}_{00} = diag([mean(trace(\Lambda_{11}))]) \quad \text{or} \quad \tilde{\Lambda}_{00} = diag([median(trace(\Lambda_{11}))]) \tag{10.10}$$

The approach with SVD decompositions is identical, the "nulled" singular values $\tilde{S}$ form the RFI-cleaned covariance $\tilde{\mathrm{C}}_{\mathrm{xx}} = U_{xx} \tilde{\mathrm{S}}_{\mathrm{xx}} \mathrm{V}_{\mathrm{xx}}^{*}$.

Non-zero $\tilde{\Lambda}_{00}$ is the traditional choice in radio astronomy RFI literature. It is effectively a "whitening" of the new $\tilde{\mathrm{C}}_{\mathrm{xx}}$ (10.9): all *q* RFI eigenvalues λ_{q} are attenuated by $L = \lambda_{\mathrm{q}} / \tilde{\lambda}_{\mathrm{q}}$ (usually L ≥ 7 dB) to equal the noise floor. Any real noise and astronomical power fractions within λ_{q} are attenuated likewise. They are not actually recoverable. The RFI "footprint" in the unmodified eigenvector matrix multiplied by non-zero replacement $\tilde{\Lambda}_{00}$ retains the RFI signature in the "clean" $\tilde{\mathrm{C}}_{\mathrm{xx}}$. Several $\tilde{\mathrm{C}}_{\mathrm{xx}}$ stacked over a long term accumulation period will reveal again any stationary RFI. Using instead $\tilde{\Lambda}_{00} = 0$ may be a good tradeoff.

Series of nulled covariance matrices $\tilde{C}_{xx}(t)$ can be summed for long accumulation periods, typically for imaging and spectral lines. Long accumulation will reveal the astronomical signal, but may also recover the strongly suppressed RFI if it is stationary. Nulled data can also be fed into pulsar de-dispersion. Strong pulsars need special care as they add dominant eigenvalues and may subsequently get “nulled” out. RFI template subtraction with reference antennas as described in section 13 is better in this case.

11 Subspace Method Examples

To give an intuitive feeling for the method, we provide here examples on “nulling” APERTIF and synthetic data. The APERTIF focal plane array was described in the previous section. The raw frequency domain covariance data used here is courtesy by Wim van Capellen. It contains wide-band satellite RFI from AfriStar, a EuroStar digital radio satellite with ETSI EN 302 550-1-3 signal encoding. APERTIF channels #1-71 cover the frequency range 1.4830-1.4967 GHz. AfriStar contaminates channels #17-45, approximately.

First we show how “Nulling” affects variances i.e. diagonal entries in $\hat{C}_{xx}$. To minimize standing wave effects of the WSRT dish we subtract Virgo A ON-source from OFF-source (galactic center) covariances. Virgo A is seen only in element #31 where it is present in all 71 channels. AfriStar is seen in all elements. Figure 4 shows the 64-element diagonal of the ON-OFF delta plotted across all 71 frequency channels. The left plot uses the original covariances contaminated by RFI. The right plot uses “nulled” covariances. Interferer count *q* was detected using MDL (10.8) and 3-sigma, followed by median replacement of RFI eigenvalues. ON- and OFF-source covariances were nulled independently, prior to taking their difference. The clean covariance plotted on the right exhibits reduced RFI levels and an improved Virgo A signal.

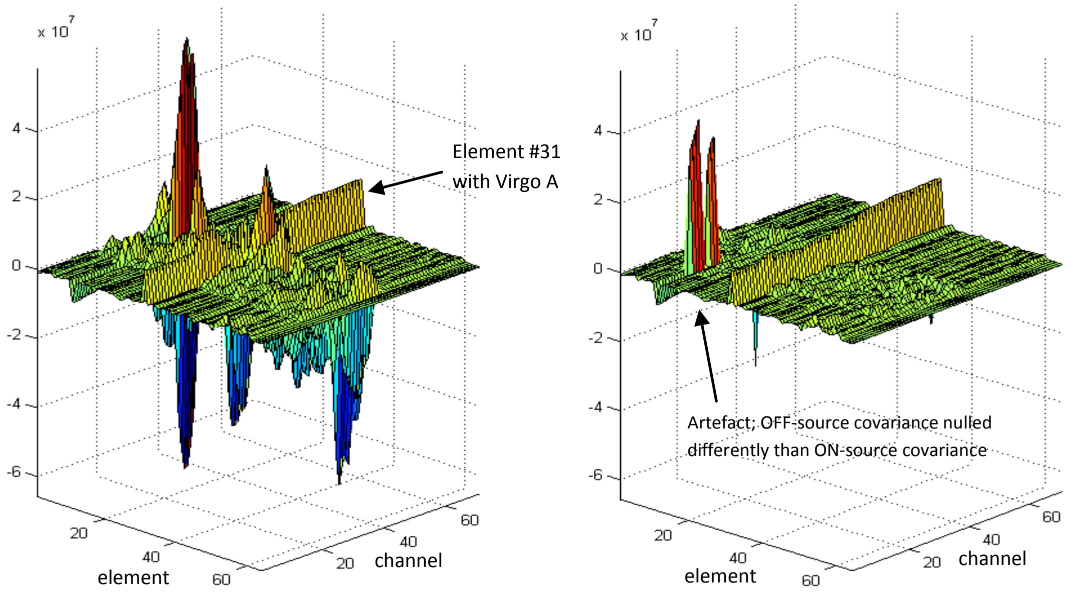


Figure 4 – Array element variances (powers) in Virgo A covariances. 64 array elements, 71 channels (1-71: 1.4830-1.4967 GHz). The strong wide-band signal seen in only one element is Virgo A. Interference present in all elements in channels 17 to 45 is RFI from the AfriStar digital radio satellite of the EuroStar satellite family. The transmission follows ETSI EN 302 550-1-3. Left image: ON-source minus OFF-source of original data. Right: same with covariance data “nulled” prior to differencing.

The ON-OFF delta also reveals an artefact: two new peaks in channel #11 are due to a difference in ON- and OFF-source "nulling". At this location OFF-source nulled data is near zero whereas ON data is not. Nulling is not always 100% robust when applied to data arithmetics in a changing RFI environment.

Next we show how "nulling" affects covariance eigenvalues. Figure 5 shows eigenvalues of the original ON-source (Virgo A) data. The 71 APERTIF frequency channel covariances are EVD decomposed. Each decomposition gives a non-decreasingly sorted list of 64 eigenvalues. The j:th largest ($j \in [1,64]$) in each channel is plotted in Figure 5. Overlaid curves have a color gradient from red to black, indicating the largest and progressively smaller eigenvalues. There are several eigenvalues below 65 dB. These are contributed by five unconnected antenna elements. Channels 17-45 contain a large eigenvalue distinctly different from the noise eigenvalues. By "nulling" the ON-source data, these large eigenvalues are simply replaced by noise-like eigenvalues. The clean eigenspectrum is shown in Figure 6.

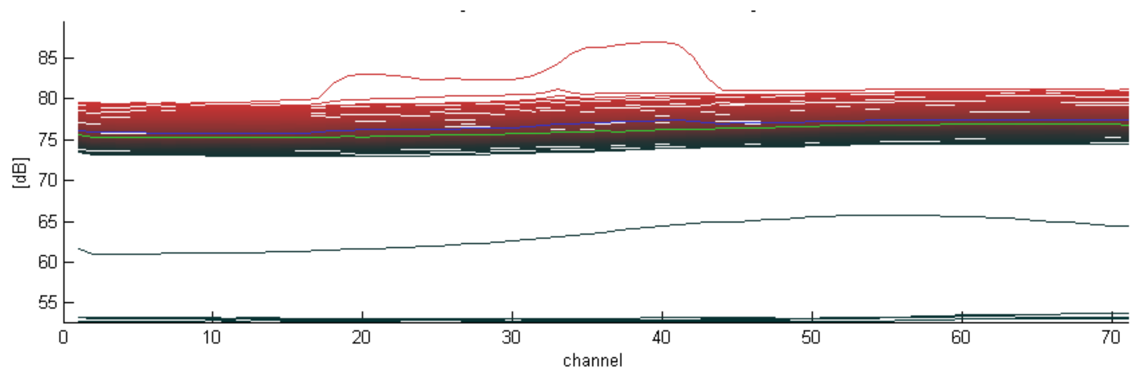


Figure 5 – Eigenspectrum of Virgo A covariances: eigenvalue power (dB) in all 71 channels (1-71: 1.4830-1.4967 GHz) is plotted as an overlay of the 1st to 64th largest eigenvalue in every channel (color gradient red to black: largest to smallest value, green: median of 64 values, blue: mean of 64 values). Covariances provided by Wim van Capellen. Five unconnected array elements contribute eigenvalues <65 dB. AfriStar causes RFI with a dominant eigenvalue in middle channels 17 to 45.

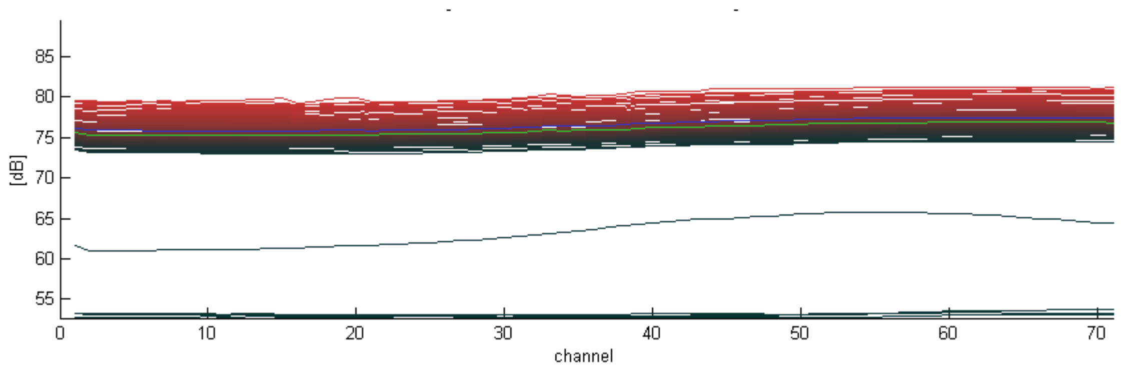


Figure 6 - Eigenspectrum of "Nulled" Virgo A 71-channel 64x64 covariances. Eigenvalue power (dB) in all 71 channels (1-71: 1.4830-1.4967 GHz) plotted as an overlay of the 1st to 64th largest eigenvalue in every channel (color gradient red to black: largest to smallest value, green: median of 64 values, blue: mean of 64 values). Covariances provided by Wim van Capellen. Five unconnected array elements contribute eigenvalues <65 dB. Interferer count *q* was estimated with MDL and 3-sigma, then *q* values were "Nulled" via median replacement. Cleaned covariance matrices were decomposed for the eigenspectrum.

An attempt to image the Virgo A source using APERTIF data was made. A crude UV gridder is provided as Matlab code in the Beamformer package. It places APERTIF antenna elements on a square grid with a 11cm spacing. The source is assumed to be at array zenith. The code computes baseline vectors B between element pairs (i,j) and places weighted cross-covariance $\tilde{C}_{xx}(i,j)$ (or rather $\tilde{C}_{xx,ON}(i,j) - \tilde{C}_{xx,OFF}(i,j)$ deltas) and its conjugate at UV plane points $+B$ and $-B$. Variances (i,i) are placed at $+B$. The complex UV plane is 2D Fourier transformed into an image. All 71 channels are stacked for a final image.

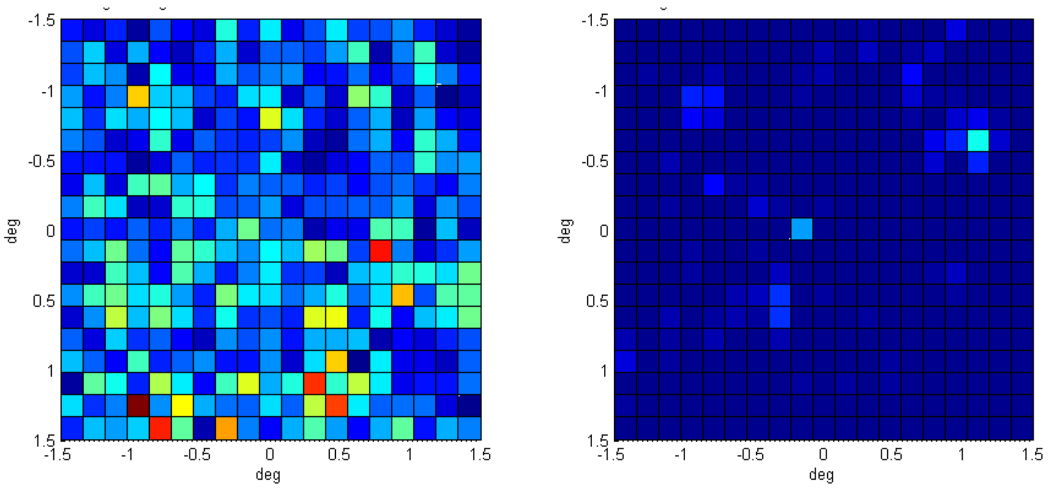


Figure 7 – UV images of On-source minus Off-source covariances. Stack of all 71 channels. Left image uses original covariances, image rms σ=3.99. Right image uses "nulled" covariances, σ=1.11. Image FOV=±1.5°. WSRT main dish f/D=0.35, D=25m subtends ±55°. Nulled data enhances Virgo A at ~(0°,0°). Point source at (-0.7°,1.1°) is wide-band RFI in WSRT spillover. Virgo A SNR increases by factor x1.12 and spillover RFI INR by factor x2.25 after nulling.

Figure 7 shows two images. The left image is derived from original APERTIF covariance data, the right from C++ library "nulled" data. No flux calibration was done. As just a single time slice was available, extensive Virgo A imaging was not possible. Nevertheless the "nulled" data image shows Virgo A as a point source near (0°,0°). The original image on the left is corrupted by RFI. Covariance nulling reduces rms noise σ=3.99 to σ=1.11 and modestly increases Virgo A SNR by factor 1.12 .

Another point source is seen near (-0.7°,1.1°). It is not present in ON or OFF data on their own. It is visible in an ON, OFF delta when stacking channels of original data unaffected by AfriStar RFI, thus unlikely to be a "nulling" artifact. The point source is outside the main dish and probably enters APERTIF elements through sidelobes. Possible origins are the Sun or wide-band RFI from the WSRT control building.

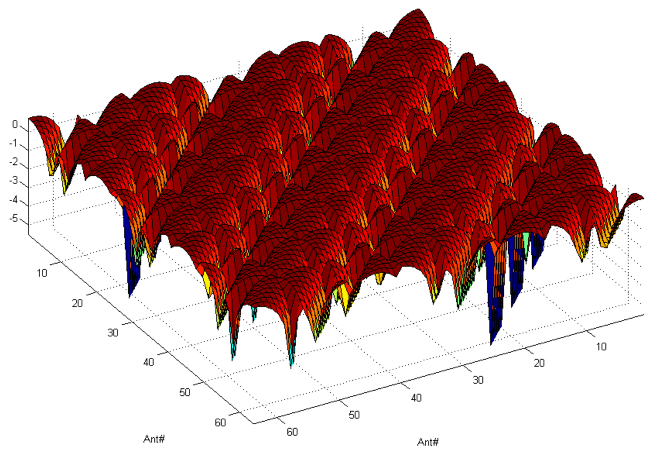


Figure 8 – Synthetic complex 64x64 covariance for a 64-element uniform grid phased array. Elements see three point sources: two RFI and one faint astronomical source (INR=10^3). Periodicity stems from the linear plot of a spatial grid layout (8*8).

As a final example we use synthetic $\hat{C}_{xx}$ covariance data for a 64-element antenna array laid out as a 8*8 uniform grid. The conjugate transpose product (10.4) of the per-antenna phase delay vector (12.1) for each incoming signal is scaled by signal power and is added to $\hat{C}_{xx}$. Figure 8 shows magnitudes of the generated complex covariance matrix $\hat{C}_{xx}$. The matrix contains three external point source signals: a faint astronomic source (${\sigma_A}^2 = 10^{-4}$) and two RFI signals (${\sigma_{I,1}}^2 = {\sigma_{I,2}}^2 = 1$) entering at distinct angles (θ,φ). Array noise power added to the $\hat{C}_{xx}$ diagonal is ${\sigma_N}^2 = 10^{-3}$. To mimic estimation errors, 64x64 random complex values $0 \le |\varepsilon_{ij}| \le {\sigma_C}^2 = 10^{-6}$ are added to all cells $\hat{C}_{xx}(i,j)$. "Nulling" of the synthetic covariance constructs a clean covariance $\tilde{C}_{xx}$. Dirty and cleaned covariances $\hat{C}_{xx}$ and $\tilde{C}_{xx}$ are UV imaged via covariance gridding and 2D Fourier transformation as described on the previous page. Figure 9 shows the resulting two images. In the left image the astronomical source is obscured by both interferers. In the right image, nulling has fully recovered the astronomical source.

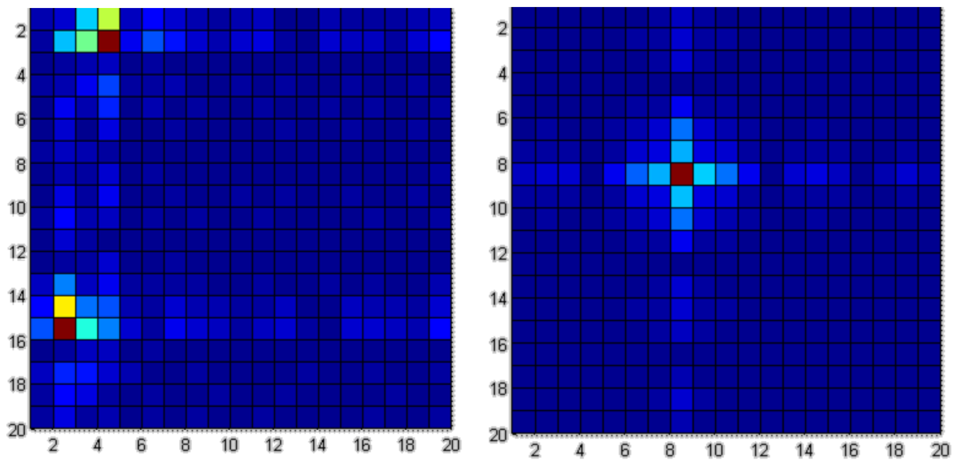


Figure 9 – UV images of synthetic data. Three signals enter at distinct 2D angles with powers ${\sigma_{I,1}}^2 = {\sigma_{I,2}}^2 = 1$, ${\sigma_A}^2 = 10^{-4}$. Array noise ${\sigma_N}^2 = 10^{-3}$ with additive covariance estimation error ${\sigma_C}^2 = 10^{-6}$. Left image uses contaminated covariance. Right image uses "nulled" covariance which reveals the astronomical source previously obscured by the two RFI signals.

12 Details on Adaptive Beamforming

Beamforming is essentially an adaptive filter process. There exists a wealth of algorithms with varying computational complexity. Each algorithm is optimal for some specific constraints. An extremely comprehensive introduction is S. Werner's PhD thesis [WE02]. The C++ library includes standard adaptive beamformers but no new developments.

Beamforming produces a set of complex weights $w = [w_0, w_1, \ldots, w_{N_{ant}-1}];\ \|w\| \equiv 1$ that phase-shifts and combines array element signals $x(t) = [x_0(t), x_1(t), \ldots, x_{N_{ant}-1}(t)]$ such that the weighted sum $y(t) = w^H \cdot x(t)$ is a spatial filter with spatial band pass towards a specific direction of an incoming plane wave with wave vector *k*. By modifying weights, the direction of the maximum response and the band pass steepness can be shifted. Deep nulls (filter zeroes) can be placed into directions of interfering plane waves, quite equivalent to notch filters. Stop band ripple caused by a steep band pass and the finite number of array elements (spatial samples) is equivalent to electrical beam sidelobes that appear into often unwanted directions. This allows signals other than the targeted celestial source to leak in.

Naturally, one can form any number of beams $y_b(t) = {w_b}^H \cdot x(t)\ ; b = 0 \ldots (B-1)$ from the same array signal vector $x(t)$, allowing the array to look into many directions simply by varying ${w_b}^H$.

Non-adaptive beamforming shifts the phases in array signal vector $x(t)$ such that for some desired beam *b* towards direction (θ, φ), the shift will cancel the phase delays r_b that a plane wave from this direction would experience at each array element location. For a planar 2D array this corresponds to electrically tilting the array to align it with the incoming plane wave. The beam steering vector r_b is given by:

$$r_b = \left[e^{-jk_b \cdot r_0}, e^{-jk_b \cdot r_1}, \ldots, e^{-jk_b \cdot r_{N_{ant}-1}}\right]\ ;\ k_b = \frac{2\pi}{\lambda}[\sin(\theta)\cos(\varphi),\ \sin(\theta)\sin(\varphi),\ \cos(\theta)] \quad (12.1)$$

Array element positions are $r_i \in \mathbb{R}^3$. Wave vector $k_b \in \mathbb{R}^3$ matches the direction (θ, φ) of the desired beam *b*. The non-adaptive beamformer weights for a classic beamformer (CBF) are simply the complex conjugate of the above steering vector r_b. They can be considered fixed FIR filter weights.

$$w_{cbf,b} = {r_b}^* \quad (12.2)$$

For adaptive beamforming, weights may be updated periodically using array covariance $\hat{C}_{xx}$. Maximum power response or minimum mean square error (MMSE) weights are:

$$w_{MMSE,b} = {\hat{C}_{xx}}^{-1} r_b \quad (12.2)$$

Maximum variance minimum distortion beamformer (MVDR) weights, also called Capon Beamformer weights, are derived from the generic linearly-constrained minimum variance (LCMV) conditions by additionally requiring that $w^H \cdot r_b \equiv 1$. Optimal weights are

$$w_{MVDR,b} = \frac{\hat{C}_{xx}^{-1} r_b}{{r_b}^* \hat{C}_{xx}^{-1} r_b} \quad (12.3)$$

MVDR is sensitive to main dish deformations and pointing errors in array elements. Uncertainties in element look directions degrade the interferer suppression, and if steering angles don't match the celestial source direction precisely, the source tends to get suppressed in the beam output signal. Widening the electrical beam angle (widening the spatial band pass) is the underlying idea of Robust MVDR (RB-MVDR). One option is to compute optimal MVDR weights using the original $\hat{C}_{xx}$ but include additive white noise $\tilde{C}_{xx} = \hat{C}_{xx} + \varepsilon^2 I$ (white noise gain constraint, WNGC). This also makes $\tilde{C}_{xx}$ less likely to be non-invertible, considering the inversion required for (12.3).

In the C++ beamformer library, Robust MVDR (RB-MVDR) is implemented as Cox Projection WNGC. It first computes $w_{MVDR,b}$, then extends it in a direction orthogonal to the steering r_b by a factor α.

$$w_{Cox,b} = \left(\frac{{w_{MVDR,b}}^H r_b}{|r_b|}\right)\frac{r_b}{|r_b|} + \alpha \cdot \left(w_{MVDR,b} - \left(\frac{{w_{MVDR,b}}^H r_b}{|r_b|}\right)\frac{r_b}{|r_b|}\right) \tag{12.4}$$

The choice of α depends on the array. $|r_b|$ is the L2 norm of the vector (Euclidean distance) to normalize r_b into a unit vector. For $\alpha = 1$ the $w_{Cox,b}$ weights are identical to MVDR. With $\alpha > 1$ a spherical error constraint increases around the exact steering, giving a wider beam i.e. wider band pass.

All above beamformer weight updates, when refreshed at long time intervals, create an undesired “pattern rumble”. Rumble happens for fast non-stationary interference. It is also caused by weight jitter associated with $\hat{C}_{xx}$ estimation errors. Rumble leads to gain fluctuations and reduces stable system integration time and thus decreases sensitivity towards celestial sources. In very small compact arrays (7-beam, 11-beam) this is problematic. Larger arrays are affected less.

Recursive update approaches are possible, they reduce rumble by adapting beamformer weights more smoothly. However, they require either slowly changing interferers or shorter (but then noisier) covariance matrix estimates. None have been implemented in the C++ library at the moment.

Due to pattern rumble, adaptive beamforming should be used only in compact arrays that have a large number of elements (for example >20).

13 Details on Reference Signal Subtraction

Interference can be subtracted from array covariance data using data from low-gain RFI reference antennas that are not sensitive to the astronomical source. Unlike subspace methods it does not require estimates on the number of interferers *q* and unlike real-time adaptive filtering it does not cause pattern rumble. Array signals are left intact as corrections are applied in post processing. However, subtraction requires covariances between reference and array antennas. These are formed and time integrated on FPGAs. Power of two matrix sizes convenient for FPGA processing may not be possible.

Van der Veen et al. describe RFI subtraction from covariance data [VE04]. They provide a generic solution. A special case for two reference antennas and at most one interferer per channel is covered in Briggs et al. [BRI00]. The C++ library implements both. Derivations and an improvement on VE04 are given below.

We start with N_{ant} reference antenna and array signals $x_i(t)$ concatenated into vector $x(t)$, similar to (10.1), but making a distinction between the *n* references antennas ($1 \le n < N_{ant}$) and the main array:

$$x(t) = \left[x_0(t), x_1(t), x_{n-1}(t), \dots, x_{N_{ant}-1}(t)\right] \tag{13.1}$$

Next we assume that reference antennas (subscript *ref*) are not sensitive to the astronomical source, but that they see the same short-term stationary interferers that also affect the main array (subscript *arr*).

$$x_i(t) = \begin{cases} n_{ref,i}(t) + \sum_q i_{ref,q}(t) & for\; 0 \le i < n \\ a_{arr,i}(t) + n_{arr,i}(t) + \sum_q i_{arr,q}(t) & for\; n \le i < N_{ant} \end{cases} \tag{13.2}$$

Noise terms $n(t)$ are assumed to be i.i.d. for all reference and array antennas. The astronomical signal $a(t)$ is present in main array data only. Signals $i_{ref,p}(t)$ and $i_{arr,p}(t)$ originate from *q* uncorrelated short-term stationary interferers $i_q(t)$. Each of the terms in (13.2) quietly incorporates the antenna gain and geometric signal delays and phase shifts. Rewriting (13.2) in vector form with all N_{ant} terms $x_i(t)$:

$$\begin{aligned} x(t) &= [\bar{0},\, a_{arr}(t)^T]^T + \left[n_{ref}(t)^T, n_{arr}(t)^T\right]^T + \sum_q \left[i_{ref,q}(t)^T, i_{arr,q}(t)^T\right]^T \\ &= a(t) + n(t) + \left[A_{ref}, A_{arr}\right](t) \end{aligned} \tag{13.3}$$

Signal sources are assumed to be mutually uncorrelated. Each adds its own independent covariance. We insert the above $x(t)$ into (10.2) to yield an estimated $\hat{\mathrm{C}}_{\mathrm{xx}}(t)$ and write this out in partitioned form:

$$\hat{\mathrm{C}}_{\mathrm{xx}} = \begin{bmatrix} \mathrm{R_{rr}} & \mathrm{R_{ra}} \\ \mathrm{R_{ar}} & \mathrm{R_{aa}} \end{bmatrix} = \hat{\mathrm{C}}_{\mathrm{a}} + \hat{\mathrm{C}}_{\mathrm{n}} + \hat{\mathrm{C}}_{\mathrm{A}} = \begin{bmatrix} \mathrm{A_{ref}A_{ref}}^{\mathrm{H}} + \sigma_{\mathrm{N,ref}}{}^2\mathrm{I} & \mathrm{A_{ref}A_{arr}}^{\mathrm{H}} \\ \mathrm{A_{arr}A_{ref}}^{\mathrm{H}} & \mathrm{C_{a,a}} + \mathrm{A_{arr}A_{arr}}^{\mathrm{H}} + \sigma_{\mathrm{N,arr}}{}^2\mathrm{I} \end{bmatrix} \tag{13.4}$$

The $\hat{\mathrm{C}}_{\mathrm{xx}}$ partitioning uses four submatrices: reference-reference ($\mathrm{R_{rr}}$: $n \times n$), reference-array ($\mathrm{R_{ra}}$: $n \times (N_{ant} - n)$, $\mathrm{R_{ar}}$: $(N_{ant} - n) \times n$) and array-array ($\mathrm{R_{aa}}$: $(N_{ant} - n) \times (N_{ant} - n)$). The term $\mathrm{C_{a,a}}$ is the main array covariance for the astronomical signal. Internal uniform noise powers are $\sigma_{\mathrm{N,ref}}{}^2 \cong \sigma_{\mathrm{N,arr}}{}^2$.

The goal is to find and subtract the interferer-induced term $\mathrm{A_{arr}A_{arr}}^{\mathrm{H}}$ included in $\mathrm{R_{aa}}$ (13.4) to get

$$\mathrm{C_{clean}} = \mathrm{C_{a,a}} + \sigma_{\mathrm{N,arr}}{}^2\mathrm{I} = \mathrm{R_{aa}} - \mathrm{A_{arr}A_{arr}}^{\mathrm{H}} \tag{13.5}$$

The main array interference $\mathrm{A_{arr}A_{arr}}^{\mathrm{H}}$ can be derived via other $\hat{\mathrm{C}}_{\mathrm{A}}$ covariances. Writing out just $\hat{\mathrm{C}}_{\mathrm{A}}$:

$$\hat{\mathrm{C}}_{\mathrm{A}} = \begin{bmatrix} \mathrm{A_{ref}A_{ref}}^{\mathrm{H}} & \mathrm{A_{ref}A_{arr}}^{\mathrm{H}} \\ \mathrm{A_{arr}A_{ref}}^{\mathrm{H}} & \mathrm{A_{arr}A_{arr}}^{\mathrm{H}} \end{bmatrix} \quad ; \quad \mathrm{A_{ref}}: n \times q\,, \;\; \mathrm{A_{arr}}: (N_{ant} - n) \times q \tag{13.6}$$

Instead of a summation-collapsed $\mathrm{A_{ref}}$: $n \times 1$ as in (13.3), we use $\mathrm{A_{ref}}$: $n \times q$ and note this does not change $\hat{\mathrm{C}}_{\mathrm{A}}$. Matrix $\mathrm{A_{ref}}$ has full rank (*min*(*n,q*)) for *q* uncorrelated interferers and *n* independent reference antennas. The $\mathrm{A_{ref}}$ product $\mathrm{A_{ref}A_{ref}}^{\mathrm{H}}$: $n \times n$ is invertible for $q \ge n$ so we can expand

$$\begin{aligned} \mathrm{A_{arr}A_{arr}}^{\mathrm{H}} &= \mathrm{A_{arr}}\left(\mathrm{A_{ref}}^{\mathrm{H}}\mathrm{A_{ref}}^{\mathrm{H}^{-1}}\right)\left(\mathrm{A_{ref}}^{-1}\mathrm{A_{ref}}\right)\mathrm{A_{arr}}^{\mathrm{H}} = \mathrm{A_{arr}A_{ref}}^{\mathrm{H}}\left(\mathrm{A_{ref}A_{ref}}^{\mathrm{H}}\right)^{-1}\mathrm{A_{ref}A_{arr}}^{\mathrm{H}} \\ &= \mathrm{R_{ar}}\left(\mathrm{A_{ref}A_{ref}}^{\mathrm{H}}\right)^{-1}\mathrm{R_{ra}} \cong \mathrm{R_{ar}}\left(\mathrm{A_{ref}A_{ref}}^{\mathrm{H}} + \sigma_{\mathrm{ref}}{}^2\mathrm{I}\right)^{-1}\mathrm{R_{ra}} = \mathrm{R_{ar}R_{rr}}^{-1}\mathrm{R_{ra}} \end{aligned} \tag{13.7}$$

The approximation holds for $\sigma_{\mathrm{N,ref}}{}^2\mathrm{I} \ll \mathrm{A_{ref}A_{ref}}^{\mathrm{H}}$. The noise $\sigma_{\mathrm{N,ref}}{}^2\mathrm{I}$ in $\mathrm{R_{rr}}$ guarantees that $\mathrm{R_{rr}}$ is always invertible, even for $q < n$, including the interference-free case $q = 0$. Thus the pseudo-inverse $\mathrm{R_{rr}}^{\dagger}$ in van der Veen [VE04] can be replaced by $\mathrm{R_{rr}}^{-1}$. For two reference antennas the inverse is:

$$\mathrm{R_{rr}}^{-1} = \begin{bmatrix} \mathrm{a} & \mathrm{b}^* \\ \mathrm{b} & \mathrm{c} \end{bmatrix}^{-1} = \frac{1}{\mathrm{ac-bb^*}} \begin{bmatrix} \mathrm{c} & -\mathrm{b}^* \\ -\mathrm{b} & \mathrm{a} \end{bmatrix} \tag{13.8}$$

Combining (13.5) and (13.7) the final RFI-cleaned covariance estimate $\tilde{\mathrm{C}}_{\mathrm{xx}}$ becomes

$$\tilde{\mathrm{C}}_{\mathrm{xx}} = \begin{bmatrix} 0 & 0 \\ 0 & \mathrm{C_{clean}} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & \mathrm{R_{aa}} - \left(\mathrm{R_{ar}}\mathrm{R_{rr}}^{-1}\mathrm{R_{ra}}\right) \end{bmatrix} \tag{13.9}$$

The C++ library compares means of diagonal entries (variances) of the reference $\mathrm{R_{rr}}$ and array $\mathrm{R_{aa}}$ submatrices. Cleaning (13.9) is done only for reference means 10 times larger than the array mean. When noise is uniform ($\sigma_{\mathrm{ref}}{}^2 \approx \sigma_{\mathrm{arr}}{}^2$) this tests for a high interferer to noise ratio which indicates presence of at least one interferer ($q > 0$) and tells that subtraction is indeed warranted.

So far we have largely ignored noise powers $\sigma_{\mathrm{N,ref}}{}^2, \sigma_{\mathrm{N,arr}}{}^2$. Noise allows to compute $\mathrm{R_{rr}}^{-1}$ but decreases the accuracy of the interference estimate (13.7). Examples are given in section 14. When references are increasingly noisy the van der Veen performance degrades significantly. It fails completely for interference to noise ratios (INR) of INR < 8 in the reference antenna.

Another method presented here is to replace $\mathrm{R_{rr}}$ by an estimate $\widehat{\mathrm{R}}_{\mathrm{rr}}$ derived from the array and reference covariance $\mathrm{R_{ra}}$. The final cleaned covariance is not affected by array-unrelated noise powers:

$$\mathrm{C_{clean}} = \mathrm{R_{aa}} - \mathrm{R_{ar}}\widehat{\mathrm{R}}_{\mathrm{rr}}{}^{\dagger}\mathrm{R_{ra}} = \mathrm{R_{aa}} - \mathrm{R_{ar}}\left(\mathrm{R_{ra}}\mathrm{R_{aa}}^{-1}\mathrm{R_{ar}}\right)^{\dagger}\mathrm{R_{ar}} \tag{13.10}$$

$$= \mathrm{R_{aa}} - \mathrm{R_{ar}}\mathrm{R_{ar}}^{\dagger}\left(\mathrm{R_{aa}}\mathrm{R_{ra}}^{\dagger}\right)\mathrm{R_{ra}}$$

$$= \mathrm{R_{aa}} - \left(\mathrm{R_{ar}}\mathrm{R_{ar}}^{\dagger}\right)\mathrm{R_{aa}}\left(\mathrm{R_{ra}}^{\dagger}\mathrm{R_{ra}}\right) = \mathrm{R_{aa}} - \left(\mathrm{R_{ar}}\mathrm{R_{ar}}^{\dagger}\right)\mathrm{R_{aa}}\left(\mathrm{R_{ar}}\mathrm{R_{ar}}^{\dagger}\right)^{*} = \mathrm{R_{aa}} - \mathrm{A}\mathrm{R_{aa}}\mathrm{A}^{*} \tag{13.11}$$

The noise-free $\widehat{\mathrm{R}}_{\mathrm{rr}}$ can be rank deficient and not invertible, requiring the use of a pseudoinverse (†). The large inverse $\mathrm{R_{aa}}^{-1}$ is computationally expensive. This is avoided by the expanded form (13.11) with pseudoinverses for the non-square matrices $\mathrm{R_{ra}}, \mathrm{R_{ar}}$. Recall that pseudoinverses are one-sided inverses and typical arrays have Hermitian $\mathrm{R_{ra}}: n \times (N_{ant} - n)$ with $n < (N_{ant} - n)$. Thus $\mathrm{R_{ra}}^{\dagger}$ is a right and $\mathrm{R_{ar}}^{\dagger}$ conversely a left inverse, defined by $\mathrm{R_{ra}}\mathrm{R_{ra}}^{\dagger} = \mathrm{I}$ and $\mathrm{R_{ar}}^{\dagger}\mathrm{R_{ar}} = \mathrm{I}$ respectively, while the opposite hand products $\mathrm{R_{ra}}^{\dagger}\mathrm{R_{ra}} = \left(\mathrm{R_{ar}}\mathrm{R_{ar}}^{\dagger}\right)^{*} \neq \mathrm{I}$ are the coupling matrices used for RFI subtraction.

Unlike van der Veen (13.9), the above methods (13.10),(13.11) do not fail even for very low reference antenna INR ratios (INR < 10 and even INR < 1). Some examples are given in the next chapter.

In similar fashion, Briggs et al. [BRI00] ignore $\mathrm{R_{rr}}$ diagonal entries. They use only one $\mathrm{R_{rr}}$ off-diagonal entry in a triple product, or bispectrum, computed towards the RFI source. This achieves closure phase on the RFI and superior RFI cancellation performance. The method is however limited to $n = 2$ reference antennas and $q \leq 1$ interferers. The new covariances $\tilde{\mathrm{C}}_{\mathrm{aa}}(l,j)$ with RFI subtracted are:

$$\tilde{\mathrm{C}}_{\mathrm{aa}}(\mathrm{i,j}) \approx \hat{\mathrm{C}}_{\mathrm{aa}}(\mathrm{i,j}) - \frac{\mathrm{R_{ar}}(\mathrm{i,a_1})\mathrm{R_{ra}}(\mathrm{a_2,j})}{\mathrm{R_{rr}}(\mathrm{a_1,a_2})} \approx \hat{\mathrm{C}}_{\mathrm{aa}}(\mathrm{i,j}) - \frac{\mathrm{R_{ar}}(\mathrm{i,a_1})\mathrm{R_{ra}}(\mathrm{a_2,j})\mathrm{R_{rr}}^{*}(\mathrm{a_1,a_2})}{\alpha(\mathrm{f}) + \mathrm{R_{rr}}(\mathrm{a_1,a_2})\mathrm{R_{rr}}^{*}(\mathrm{a_1,a_2})} \tag{13.12}$$

The C++ library requires the input covariance matrix to have same layout as in (13.4) with $\mathrm{R_{rr}}: 2 \times 2$ in the top left corner. Parameter $0 < \alpha(\mathrm{f}) \ll 1$ helps numerical stability in RFI-free channels with zero cross-correlation.

14 Reference Signal Subtraction Examples

At the time of writing, no APERTIF data incorporating RFI reference antennas was available. Thus we use a synthetic covariance generator model similar to that in section 11. However, data for certain array element indices is computed differently. These elements correspond to RFI reference antennas. The astronomical signal is not included, the RFI signal is added at a gain higher than that of the rest of the array and noise power can be unequal to main array noise power. Reference antenna positions are not critical. Details are in the Matlab files *subspcrfi_modelgen2.m* and *subspcrfi_test_subtraction.m*.

The performance of the four different subtraction methods Kesteven-Briggs (KB), Generic I by van der Veen (GV), Generic II (G2) from (13.10) and Generic III (G3) from (13.11) is given Table 4. The synthetic data use one astronomical point source, various system noise levels, and varying reference antenna numbers and RF interferer counts and powers. Performance is stated in terms of the maximum absolute error given by ε = max(abs(delta)). Delta is the element by element difference between the subtraction cleaned covariance matrix and an originally RFI-free covariance matrix. For a relative performance figure, values ε may be divided by the ${\sigma_{\text{astro}}}^2$ signal power found in the first column of the table.

Comparing Table 4 results shows the KB method has the best performance for single RFI, while G3 outperforms it for dual RFI. G2 wins in the triple RFI situation. The coefficient of variation ($c_v = \sigma_\varepsilon/\mu_\varepsilon$; error standard deviation divided by mean error) is not shown in the table. The most stable are KB and G3 with c_v≈1e-7. GV is stable with c_v≈1e-3 but generally has high error. The triple-RFI best performer G2 has the highest variation with c_v≈1e-2. Thereby the best method to choose is either KB or G3.

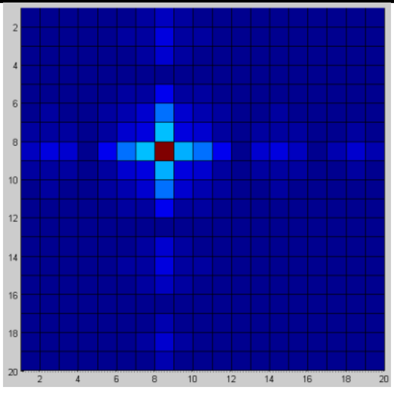
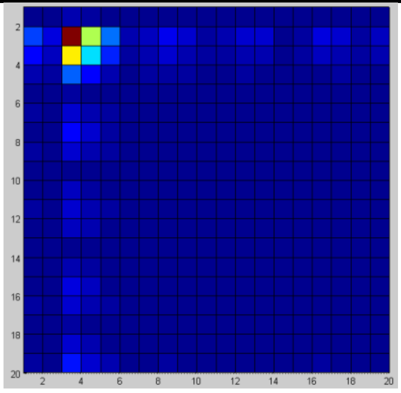
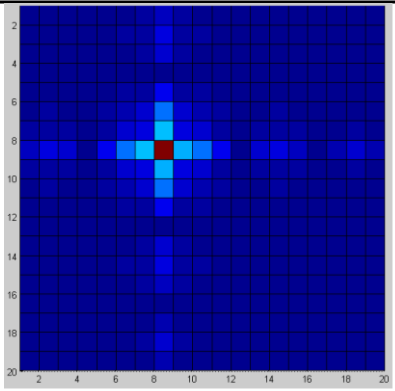
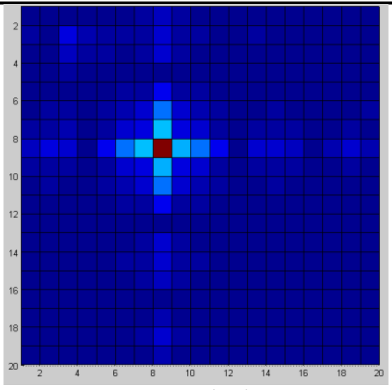
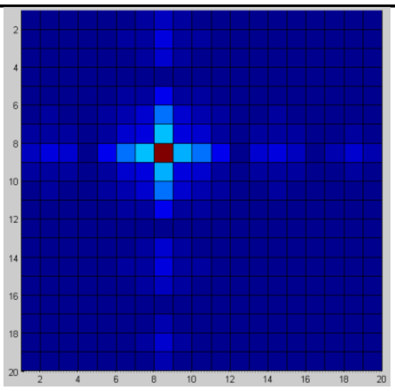
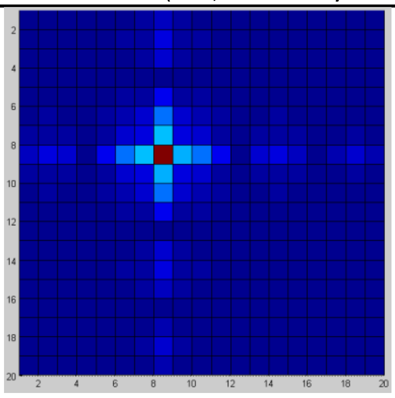
Table 4 – Performance comparison of all four subtraction methods for the same 64-element antenna array. Rows show signal power level inputs and the emulated covariance estimation error, number of antennas (reference + array), the derived smallest interference to noise ratio (INR) in the references and the resulting max. abs. errors ε of each method relative to the RFI-free model. Not all cases are physical. Changed input values and lowest errors are indicated by bold type.

	${\sigma_{\text{astro}}}^2$	${\sigma_{\text{rfi,arr}}}^2$	${\sigma_{\text{rfi,ref}}}^2$	${\sigma_{\text{N,arr}}}^2$	${\sigma_{\text{N,ref}}}^2$	N_{ant}	|ε(C)|	INR	ε(KB)	ε(GV)	ε(G2)	ε(G3)
		single RFI				**2 refs**						
1)	1e-3	2	x 15	1e-6	1e-6	2+64	1e-9	30e6	**2.2e-11**	3.8e-8	1.5e-8	5.5e-8
2)	1e-3	2	x 15	1e-6	**3e-3**	2+64	1e-9	10e3	**2.2e-11**	1.0e-4	1.8e-8	5.5e-8
3)	1e-3	2	x 15	1e-6	**3**	2+64	1e-9	10	**2.2e-11**	9.5e-2	2.0e-8	5.5e-8
4)	1e-3	2	x 15	1e-6	3	2+64	**1e-6**	10	**4.1e-11**	9.5e-2	1.5e-1	2.1e-5
		dual RFI				**≥2 refs**						
5)	1e-3	[2 ; 5]	x 15	1e-6	1e-6	2+64	1e-9	30e6	7.0	2.8e-7	1.5e-7	**7.6e-8**
6)	1e-3	[2 ; 5]	x 15	1e-6	**3**	2+64	1e-9	10	7.0	0.72	1.5e-7	**7.6e-8**
7)	1e-3	[2 ; 5]	x 15	1e-6	3	**3**+64	1e-9	10	---	0.41	1.5e-7	**7.6e-8**
8)	1e-3	[2 ; 5]	x 15	1e-6	3	**4**+64	1e-9	10	---	0.19	1.5e-7	**7.6e-8**
9)	1e-3	[2 ; 5]	x 15	1e-6	**30**	**2**+64	1e-9	1	7.0	3.44	1.5e-7	**7.6e-8**
10)	1e-3	[2 ; 5]	x **3e-3**	1e-6	30	2+64	1e-9	1e-4	7.0	6.99	1.5e-7	**7.6e-8**
		triple RFI				**≥3 refs**						
11)	1e-3	[2 ; 3 ; 5]	x 15	1e-6	1e-6	3+64	1e-9	30e6	---	2.8e-5	**2.2e-6**	4.1e-6
12)	1e-3	[2 ; 3 ; 5]	x 15	1e-6	**3e-3**	3+64	1e-9	10e3	---	8.4e-2	**2.2e-6**	4.1e-6
13)	1e-3	[2 ; 3 ; 5]	x 15	1e-6	**3**	3+64	1e-9	10	---	6.02	**2.2e-6**	4.1e-6
14)	1e-3	[2 ; 3 ; 5]	x 15	1e-6	**1e-6**	**4**+64	1e-9	30e3	---	4.6e-6	**8.0e-7**	4.1e-6
15)	1e-3	[2 ; 3 ; 5]	x 15	1e-6	1e-6	**5**+64	1e-9	30e3	---	1.3e-6	**5.5e-7**	4.1e-6

To provide a visual feeling for the performance of each subtraction method we will list separate UV image sets for three cases: single RFI, dual RFI and triple RFI interferers per channel. Synthetic covariance data sets generated with corresponding parameters were cleaned by the different methods and transformed into UV images using UV gridding and 2D Fourier as described near the end of section 11.

Note that the UV images include of course only the array-array covariance data. Covariances against reference antennas are not used because they do not contain the astronomical source of interest.

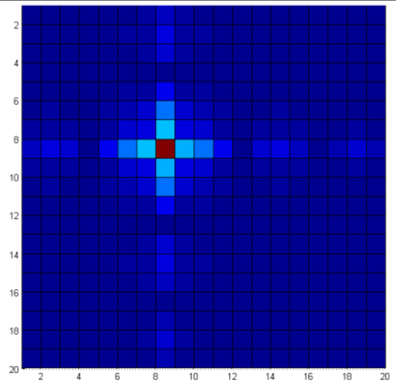
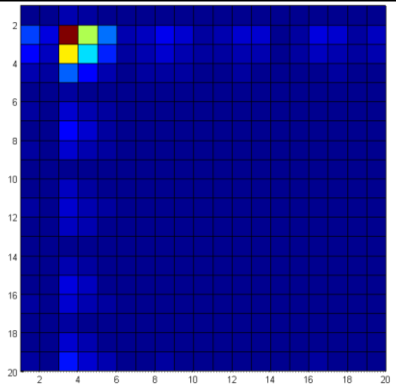
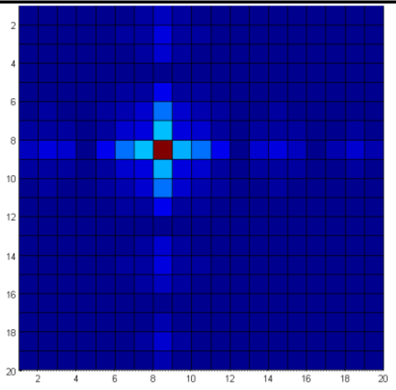
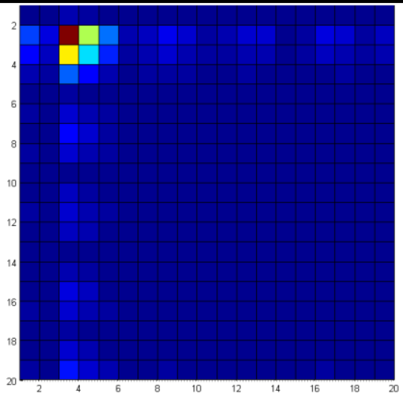
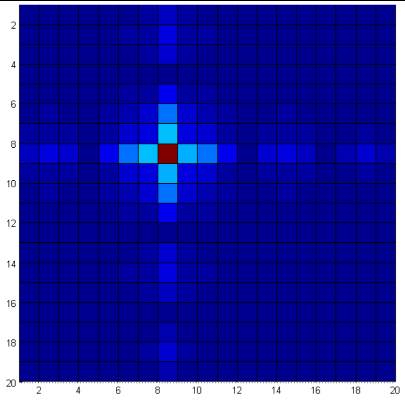
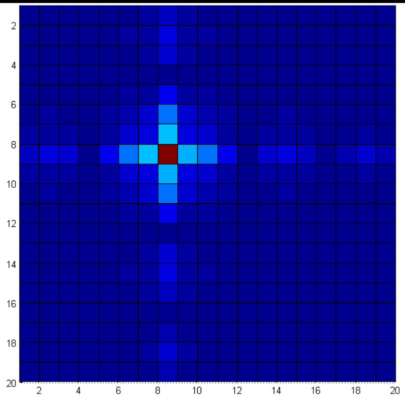
Table 5 – UV images of dirty and cleaned covariances, generated with Table 4 parameters of result 2.

Model with 1 RFI and 2 references, INR=($\boldsymbol{\sigma_{rfi,ref}/\sigma_{N,ref}^2}$)=10e3		
Parameters of Table 4 result 2: $\boldsymbol{\sigma_{astro}}^2$=1e-3, $\boldsymbol{\sigma_{rfi,arr}}$=2, $\boldsymbol{\sigma_{rfi,ref}}$=2*15, $\boldsymbol{\sigma_{N,arr}}^2$=1e-6, $\boldsymbol{\sigma_{N,ref}}^2$=3e-3, N_{ant}=2+64.		
 RFI-free	 Contaminated	 KB method (best, see Table 4**)**
 GV method	 G2 method	 G3 method

Results for the example scenario with one RF interferer and two reference antennas located at some rather arbitrary point are shown in the UV images of Table 5. The reference antennas may simply be additional elements (with omni directivity) located within a larger phased array. They can also be external antennas situated close to a focal plane array. In terms of simulation results, UV images and optimality, the exact location makes no difference.

The Table 5 UV images show that all four RFI subtraction methods are performing very well in a subjective visual comparison. The exact subtraction errors are stated in Table 4. From that table the Kesteven-Briggs method stands out as providing the most exact subtraction with least residual RFI.

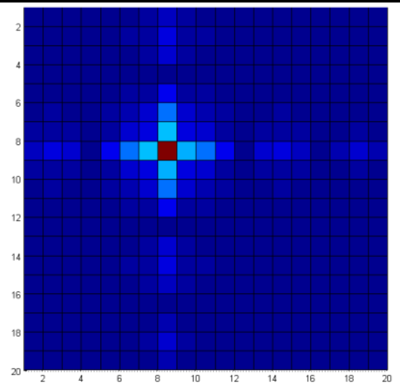
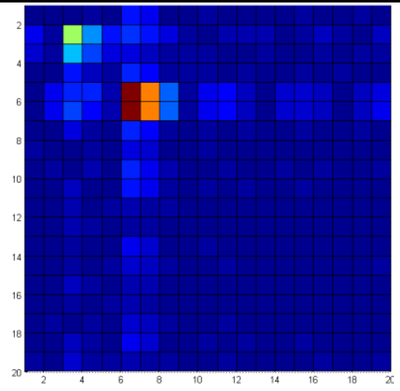
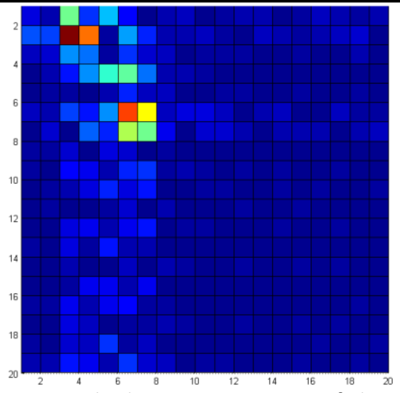
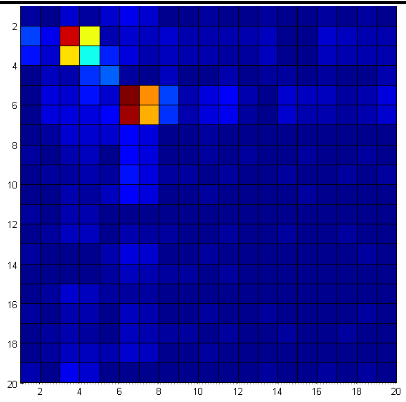
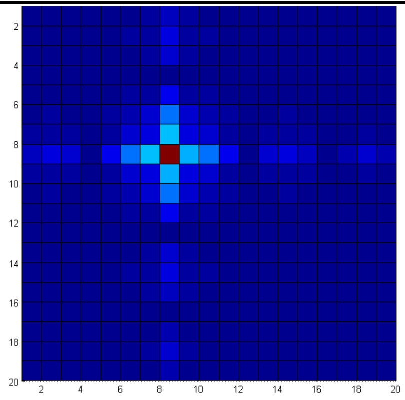
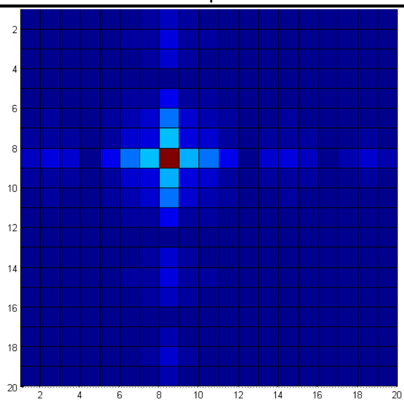
Table 6 - UV images of dirty and cleaned covariances, generated with Table 4 parameters of result 3.

Model with 1 RFI and 2 references, INR=($\sigma_{\mathbf{rfi,ref}}/\sigma_{\mathbf{N,ref}}{}^2$)=10		
Parameters of Table 4 result 3: $\boldsymbol{\sigma}_{\mathbf{astro}}{}^2$=1e-3, $\boldsymbol{\sigma}_{\mathbf{rfi,arr}}$=2, $\boldsymbol{\sigma}_{\mathbf{rfi,ref}}$=2*15, $\boldsymbol{\sigma}_{\mathbf{N,arr}}{}^2$=1e-6, $\boldsymbol{\sigma}_{\mathbf{N,ref}}{}^2$=3, N_{ant}=2+64.		
 RFI-free	 Contaminated	 KB method (best, see Table 4**)**
 GV method	 G2 method	 G3 method

In comparison to scenario depicted in Table 5 on the previous page, for Table 6 we have lowered the reference antenna INR from a modestly high 10e3 to a rather low INR of 10. Such a low INR corresponds to weakly detected RFI or low power RFI in the reference antennas. The van der Veen generic approach leaves considerable residual RFI, while the Kesteven-Briggs method continues to perform best, followed by the G2 and then G3 methods.

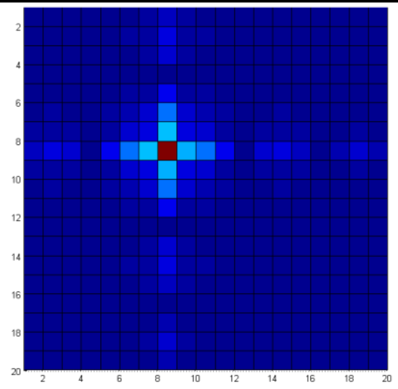
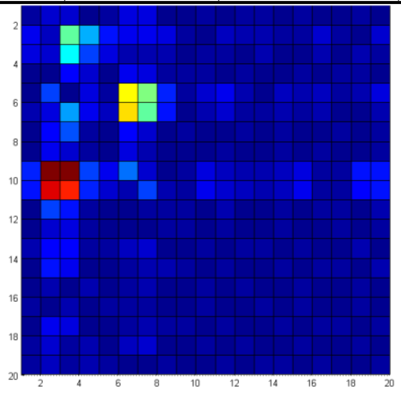
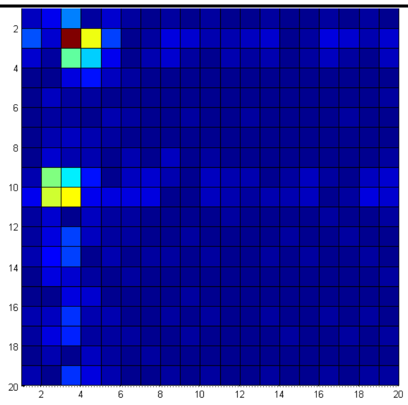
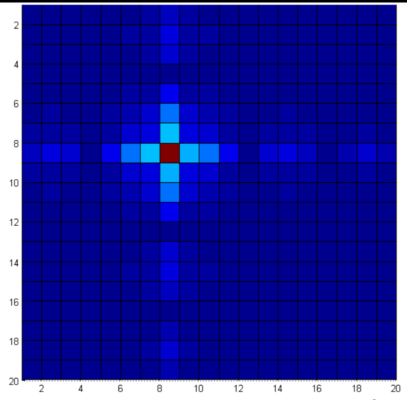
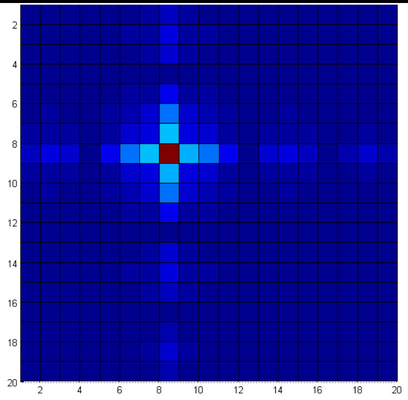
It is now left to show the performance with two RF interferers, a scenario not covered by the KB method (an attempt to extend the closure phase relation in the KB method to cover several RF interferers via a larger number of interconnected closure phase triangles has been unsuccessful so far). The two tables on the next pages show situations with two and three RF interferers.

Table 7 - UV images of dirty and cleaned covariances, generated with Table 4 parameters of result 9.

Model with 2 RFI and 2 references, INR=min($\boldsymbol{\sigma_{rfi,ref}}/\boldsymbol{\sigma_{N,ref}}^2$)=1		
Parameters of Table 4 result 9: $\boldsymbol{\sigma_{astro}}^2$=1e-3, $\boldsymbol{\sigma_{rfi,arr}}$=[2;5], $\boldsymbol{\sigma_{rfi,ref}}$=[2;5]*15, $\boldsymbol{\sigma_{N,arr}}^2$=1e-6, $\boldsymbol{\sigma_{N,ref}}^2$=30, N_{ant}=2+64.		
 RFI-free	 Contaminated	 KB method: assumption #RFI≤1 fails
 GV method	 G2 method	 G3 method (best, see Table 4**)**

The Table 7 figures show a two interferer, two reference antenna scenario. The KB method fails as the assumption of at most 1 interferer no longer holds. Additionally, the low INR of 1 causes the GV method to fail. The G2 and G3 methods are still able to leave no visible trace of RFI in the cleaned covariance.

Table 8 - UV images of dirty and cleaned covariances, generated with Table 4 parameters of result 12.

Model with 3 RFI and 3 references, INR=min($\boldsymbol{\sigma_{rfi,ref}/\sigma_{N,ref}^2}$)=10e3		
Parameters of Table 4 result 12: $\boldsymbol{\sigma_{astro}}^2$=1e-3, $\boldsymbol{\sigma_{rfi,arr}}$=[2;3;5], $\boldsymbol{\sigma_{rfi,ref}}$=[2;3;5]*15, $\boldsymbol{\sigma_{N,arr}}^2$=1e-6, $\boldsymbol{\sigma_{N,ref}}^2$=3e-3, N_{ant}=3+64.		
 RFI-free	 Contaminated	KB method: not applicable, not shown. Assumption #RFI≤1 does not hold.
 GV method	 G2 method (least error, see Table 4)	 G3 method (most stable, see text)

Lastly, Table 8 figures show a three interferer, three reference antenna setting at a modestly high INR of 10e3. The KB method fails as the assumption of at most 1 interferer no longer holds. The INR is still relatively too low for the GV method to work successfully. The G2 and G3 methods are again able to leave no visible trace of RFI in the cleaned covariance.

15 Credits and Future work

Wim van Capellen from ASTRON kindly provided the APERTIF 1.49 GHz on- and off-source covariance data snapshots on Virgo A with prominent AfriStar satellite RFI contaminating the covariance matrices. This covariance data was used to demonstrate Nulling methods.

Regarding future work, Jeffs and Warnick [JW09] findings on spectral bias (“spectral scooping”) for narrowband interference are interesting. Bias is caused by beamformer weights calculated from a covariance matrix that is only an estimate and not exact. Spectral bias may be of interest mainly for PSD estimation and correction. Spectral line observations might benefit from such corrections.

An application for RFI eigenvector template matching (“spatial footprint” matching) would also be interesting. This refers to the simple transform $y(t) = Wx(t)$ mentioned in section 10. Matrix W is the

RFI eigenvector matrix and $x(t)$ is a real-time data vector from an antenna array. The RFI score vector $y(t)$ is thresholded to detect a match to the RFI. Array RFI flagging in real-time on FPGA, or off-line in software on slightly time averaged $x(t)$ are possible. This method would be new in radio astronomy.

A programming related task could be the integration of the library functions into AIPS or CASA. For CASA this is reasonably straight forward. Compiling the huge CASA package by one self from public sources is the only major issue; that path is unsupported and effectively undocumented. Unsurprisingly, CASA compilation attempts ran into several major issues, but solvable given time. Integrating the library into AIPS is harder. AIPS uses Fortran and C while the library is object oriented C++.
Unfortunately as well CASA and AIPS general directions are currently unclear in regard of whether or not they might support small local arrays or focal plane arrays. Most arrays (ASKAP, LOFAR, …) use their own toolset and pipeline.

16 References

[BK05] F. H. Briggs, J. Kocz; *Overview of Technical Approaches to RFI Mitigation*, e-print arXiv:astro-ph/0502310, 2005, http://arxiv.org/pdf/astro-ph/0502310
[BRI00] F. H. Briggs et al.; *Removing radio interference from contaminated astronomical spectra using an independent reference signal and closure relations*, The Astronomical Journal Vol 120 No 6, 2000, http://arxiv.org/abs/astro-ph/0006222v2
[CZO87] H. Cox, R. Zeskind, M. Owen; *Robust adaptive beamforming*, IEEE Trans ASSP, Vol ASSP-35, 1365-1377, 1987, http://ieeexplore.ieee.org/iel6/29/26204/01165054.pdf
[IPP05] Ippoliti, Romagnoli, Fontanella; *A noise estimation method for corrupted data*, Statistical Methods and Applications, Vol 14, 343-356, 2005, http://www.springerlink.com/content/74x042n77330115t/
[JW09] Jeffs, Warnick; *Spectral Bias in Adaptive Beamforming With Narrowband Interference*, IEEE Trans SP, Vol 57, No 4, April 2009, http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=04731746
[VE04] Van der Veen et al.; *Spatial filtering of RF interference in Radio Astronomy using a reference antenna*, Proc. IEEE ICASSP, pp. II. 189-193, 2004, *http://ieeexplore.ieee.org/iel5/9248/29344/01326226.pdf*
[WE02] S. Werner; *Reduced Complexity Adaptive Filtering Algorithms with Applications To Communication Systems*, 2002, ISBN 951-22-6087-5, http://lib.tkk.fi/Diss/2002/isbn9512260875/
[WK85] M. Wax, T. Kailath; *Detection of Signals by Information Theoretic Criteria*, IEEE Trans ASSP, Vol ASSP-33, No 2, April, 1985, *http://ieeexplore.ieee.org/iel6/29/26190/01164557.pdf*