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

Max-Planck Institute for Radio Astronomy
Jan Wagner, 29. August 2011

Developed for FP7 ALBiUS WP6 D6.3.1, RFI Mitigation
This work received funding under EC Contract #

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.

Contents

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.

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
```

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 two types of RFI Template subtraction. See van der Veen [VE04] and Briggs [BRI00]. Requires that: 1) covariance data was observed with $N_{ref} \geq 1$ reference antennas 2) must have $N_{ref} \geq N_{RFI/channel}$, otherwise system underdetermined 3) 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).

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.

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. The double-precision performance is in Table 2 below.

Table 2 – Throughput of full processing using double precision complex arithmetic, 64 phased array elements, synthetic and APERTIF 71-channel 64x64-size covariance data in memory, running on 1 core of a dual-processor Intel Xeon E5430 system (12MB L2, 2.66 GHz, quad core).

# 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

Computation can be spread across available CPU cores using a ParallelFor loop. Performance of the processing steps will scale linearly. It is also possible to recompile the C++ library to use 32-bit single precision floating point in all vector and matrix arithmetic. The single precision performance is shown in Table 3 below.

Table 3 - Throughput of full processing using single precision complex arithmetic, 64 phased array elements, synthetic and APERTIF 71-channel 64x64-size covariance data in memory, running on 1 core of a dual-processor Intel Xeon E5430 system (12MB L2, 2.66 GHz, quad core).

# 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

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, noise powers, list of signals (their powers and angles) and 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 -90..90deg phi/theta angles. The array radiation pattern is plotted in 3D.
subspcrfi_plotEVspec	Plot multi-channel eigenvalue spectrum. Overlays N most dominant eigenvalues into the same plot.
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.

	Uses methods from van der Veen [VE04] and Briggs [BRI00].
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.

Requirements on the Antenna Array data

Below are the (reasonable) requirements the input data must be 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 earlier, while planning the technical aspects of an astronomic observation:

1. "Covariances" between array elements should be formed and time-integrated over short time intervals (STI: 1ms<T_{int}<10ms). The choice of T_{int} is a balance between increased noise at very short T_{int} on one hand, and less effective RFI mitigation at the other due to multipathing and RFI variability at very long T_{int}.
2. Time-integrated covariances should best be formed on FPGA (fixed-point) or GPU (single precision floating point) for CPU speed and I/O limit reasons.
3. If "Covariances" are cross-correlations, the matrices must contain frequency domain data.
4. If "Covariances" are time-domain covariances, the matrices must contain time domain. Signals of all contributing antennas need to have zero mean (E[X] = 0).
5. RFI reference antennas may be used. In this case, full covariances between reference antennas and array antennas need to be formed. While giving indices out to all the antennas, the reference antennas should have the lowest indices. That is, reference antenna data should be in the top left corner of the covariance matrices.
6. 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.

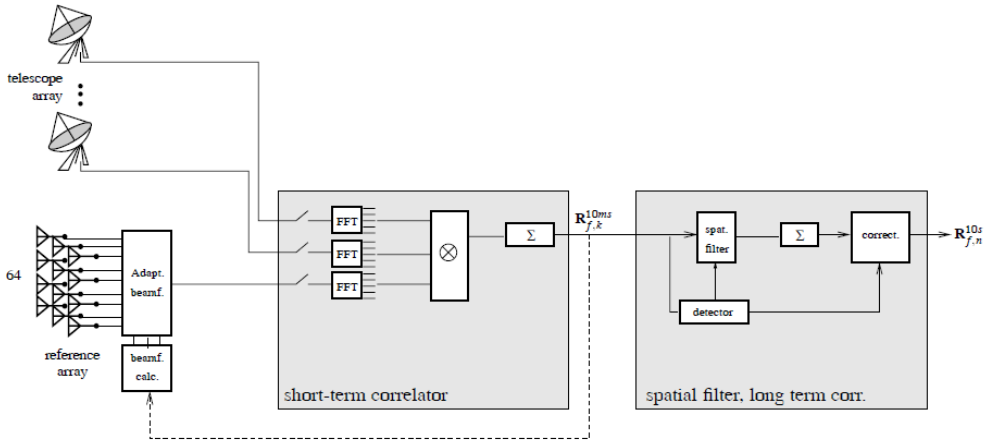

Figure 1 – Suggested processing configuration. Figure is re-used from van der Veen et al. [VE04].
Mid block: All signal sources are channelized (FFT), channel covariances (X) are short-term integrated (Σ) into matrices $R_{f,k}$.
Right block: matrices may be processed in the C++ library (RFI reference subtraction, spatial filtering e.g. Nulling) before long term integration, or C++ library can update RFI-nulling beam weights (beamf. calc). Left: signal sources and RFI reference signals.

An example telescope configuration can be seen in Figure 1 by van der Veen et al. They used the phased array as an RFI reference "antenna", with beams steered towards RFI signals. The telescope array

Data processing in Figure 1 consists of cross-correlation matrices ("covariances") from several FFT channels (subbands) being formed in FX correlator style. Cross-correlations are time integrated over some short time interval (STI). These STI covariance estimates can be further processed with for example this C++ Beamformer library, outputting new beamformer weights or modified covariance data that went through RFI mitigation steps. The cleaned STI covariance estimates are then further time-integrated according to observer wishes.

Details on Subspace Methods

The C++ library and Matlab code provide decompositions of the Hermitian array covariance matrix estimate ($\hat{C}_{xx}$) and has methods for interferer nulling. Nulling is based on partitioning of the eigenspaces of the matrix decomposition into interferer and noise subspaces. Below is a very condensed summary of the method. We start with a receiver array that has N_{ant} elements and a single narrow-band frequency channel. The time snapshot of all N_{ant} real-valued or complex-valued signals from the array is combined into a signal vector $x(t)$,

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

The true array covariance or cross-correlation matrix C_{xx} is estimated by $\hat{C}_{xx}$ which forms the average of signal cross-correlations over a short time range that consists of M signal vector snapshots,

$$\hat{C}_{xx}(t) = \frac{1}{M}\sum_{i=0}^{M-1} x(t+iT)\cdot x^*(t+iT) \quad (2)$$

where x* denotes the complex conjugate transpose. $\hat{C}_{xx}$ is a Hermitian conjugate symmetric (N_{ant} x N_{ant}) matrix and is non-singular when enough snapshots are integrated ($M > N_{ant}$). Covariance data passed to the C++ library must be data integrated by $M > N_{ant}$. This may be problematic in pulsar observations at very short timescales with a large number of antennas.

For short integration times, with no RFI present, and with a celestial source signal power less than the array element noise power $\sigma_N{}^2$, the $\hat{C}_{xx}(t)$ estimate is close to the cross-correlation of an independent identically distributed (i.i.d.) random variables process that has N_{ant} variables. In this case $\hat{C}_{xx}$ is the cross-correlation of white noise and $\hat{C}_{xx}$ has full rank and is well-conditioned.

Now we introduce RFI interferers. First we make the reasonable assumption that the astronomic, interferer and array noise signals are mutually orthogonal, that is, they are uncorrelated during the M-snapshot averaging time (the time is assumed to be shorter than any significant multipathing effects of the interferer). Now $\hat{C}_{xx}$ can be expressed as the linear sum of individual contributions,

$$\hat{C}_{xx}(t) = \hat{C}_A(t) + \hat{C}_I(t) + \hat{C}_{N(t)} \approx \hat{C}_A(t) + \hat{C}_I(t) + \sigma_N{}^2 I \quad (3)$$

where $\sigma_N{}^2 I : N_{ant} \times N_{ant}$ is the diagonal matrix of true auto-correlated noise power estimated by $\hat{C}_{N(t)}$. Note that covariance $\hat{C}_I(t)$ cannot be reliably estimated a priori. We thus can't simply subtract $\hat{C}_I(t)$ to yield an RFI-free version of $\hat{C}_{xx}$. However, when the number of RFI interferers *q* is less than the total number of antennas ($q < N_{ant}$), the interferer covariance estimate $\hat{C}_I(t)$ becomes rank deficient (rank *q*) and ill-conditioned. The eigendecomposition of $\hat{C}_I(t)$ has only *q* non-zero eigenvalues, remaining $N_{ant} - q$ eigenvalues are zero. This is the starting point for RFI mitigation.

To start nulling, the RFI-contaminated Hermitian covariance matrix estimate $\hat{C}_{xx}$ is first converted into its singular value decomposition (SVD) or its eigenvalue decomposition (EVD), respectively

$$\hat{C}_{xx} = USV^* = W\Lambda W^{-1} \text{ ; with properties } V^* = V^{-1} \text{ and S, } \Lambda \text{ diagonal} \quad (4)$$

Square matrices S and Λ contain the eigenvalues of $\hat{C}_{xx}$ on their diagonal. Diagonal values are usually sorted in non-increasing order with the largest value in the top left (0,0) of the matrix. Matrices U and V contain the left-hand and right-hand SVD eigenvectors. Eigenvectors of the EVD are in matrix W.

SVD and EVD decompositions are closely related. For simplicity we treat only nulling using the EVD decomposition. As a practical note, EVD has numerical problems for ill-conditioned $\hat{C}_{xx}$. SVD may be preferable as it tends to be more stable and accurate when $\hat{C}_{xx}$ is ill-conditioned.

The eigenvalue decomposition of $\hat{C}_{xx}$ in (3) is, by linearity,

$$\hat{C}_{xx} = \hat{C}_A(t) + \hat{C}_I(t) + \sigma_N{}^2 I = W_{xx}(\Lambda_A + \Lambda_I + \Lambda_N){W_{xx}}^{-1} = W_{xx}\Lambda_{xx}{W_{xx}}^{-1} = W\begin{bmatrix}\Lambda_{00} & 0\\ 0 & \Lambda_{11}\end{bmatrix}W^{-1} \quad (5)$$

with Λ_{xx} eigenvalue submatrix Λ_{00}: $q \times q$ with *q* interferer powers and Λ_{11}: $(N_{ant} - q) \times (N_{ant} - q)$ with noise powers. Note all eigenvalues in Λ_{xx} are non-negative and real-valued for Hermitian $\hat{C}_{xx}$.

Applying the earlier assumption that astronomic source noise power is lower than the antenna noise (${\sigma_{\mathrm{A}}}^2 \ll {\sigma_{\mathrm{N}}}^2$) and that each antenna sees an interferer ${\sigma_{\mathrm{I,k}}}^2$ $(\mathrm{k} \in [0, N_{ant}])$ then the diagonal eigenvalue submatrices Λ_{00} (from interferer space) and Λ_{11} (from noise space) are

$$\Lambda_{00} \approx \begin{bmatrix} {\sigma_{\mathrm{I},0}}^2 + {\sigma_{\mathrm{N},0}}^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & {\sigma_{\mathrm{I,q-1}}}^2 + {\sigma_{\mathrm{N,q-1}}}^2 \end{bmatrix} \text{ containing interferer noise powers } {\sigma_{\mathrm{I,k}}}^2$$

$$\Lambda_{11} \approx \begin{bmatrix} {\sigma_{\mathrm{N,q}}}^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & {\sigma_{\mathrm{N,N_{ant-1}}}}^2 \end{bmatrix} \text{ containing only antenna noise powers} \quad (6)$$

To get an RFI-free estimate $\tilde{\mathrm{C}}_{\mathrm{xx}} = W_{xx}\tilde{\Lambda}_{xx}{W_{xx}}^T$ using a modified eigenvalue matrix $\tilde{\Lambda}_{xx}$, the entire interferer submatrix $\tilde{\Lambda}_{00}$ could be set to $\tilde{\Lambda}_{00} = \overline{0}$ ("Nulling"). However, this ignores the ${\sigma_{\mathrm{N,k}}}^2$ contribution in (6) ${\sigma_{\mathrm{I,k}}}^2 + {\sigma_{\mathrm{N,k}}}^2$ and biases $\tilde{\mathrm{C}}_{\mathrm{xx}}$. In practice a better approach is to "null" with an estimate of noise space eigenvalues found in Λ_{11}, and fill the diagonal of $\tilde{\Lambda}_{00}$ using either

$$\tilde{\Lambda}_{00} = diag([mean(trace(\Lambda_{11}))]) \quad \text{or} \quad \tilde{\Lambda}_{00} = diag([median(trace(\Lambda_{11}))]) \quad (7)$$

Median is more robust. The same eigenvalue replacement can be applied to SVD decompositions, too, to first create a "nulled" singular value matrix $\tilde{S}$ and then form the RFI-free estimate $\tilde{\mathrm{C}}_{\mathrm{xx}} = U_{xx}\tilde{\mathrm{S}}_{\mathrm{xx}}\mathrm{V}^*_{\mathrm{xx}}$.

The number of largest eigenvalues or singular values to null, i.e. the number of interferers $\tilde{q}$, may be estimated using Minimum Descriptor Length (Rissanen 1978) or other information criteria applied to a list of eigenvalues. For the log likelihood of MDL(k) we use a ratio of geometric and arithmetic means.

$$\tilde{q} = \arg \min_{k \in [0, \mathrm{N_{ant}-1}]} MDL(k) \quad (8)$$

$$MDL(k) = -M(\mathrm{N_{ant}} - \mathrm{k}) \cdot \ln \frac{\left(\prod_{i=k}^{\mathrm{N_{ant}-1}} \lambda_i\right)^{\frac{1}{\mathrm{N_{ant}-k}}}}{\frac{1}{\mathrm{N_{ant}-k}}\left(\sum_{i=k}^{\mathrm{N_{ant}-1}} \lambda_i\right)} + \frac{1}{2}k(2\mathrm{N_{ant}} - \mathrm{k} + 1) \cdot \ln M \quad (9)$$

The $\tilde{q}$ estimated by MDL is reliable when the interferer to noise ratio (INR) ${\sigma_{\mathrm{I}}}^2/{\sigma_{\mathrm{N}}}^2 \gg 1$ is large. The list of eigenvalues then has clear outliers. The log likelihood detects the dissimilarity of eigenvalues – geometric and arithmetic means are identical only when all eigenvalues take the same value.

MDL is only one possibility to estimate $\tilde{q}$. Other options to count eigenvalues that exceed a noise eigenvalue threshold and should thus be classified into RFI interferer space are: 1) find eigenvalues that exceed three standard deviations from the mean or median, 2) traverse a sorted eigenvalue list in non-decreasing order and find a point of sufficiently large deviation in say a moving average or simply local slope, 3) use some knee point detection method such as "Kneedle".

The typical shape of eigenvalues free of RFI and eigenvalues with two RFI interferers are shown in Figure 2 and Figure 3. The figures also show eigenvalues detected with MDL and 3-sigma.

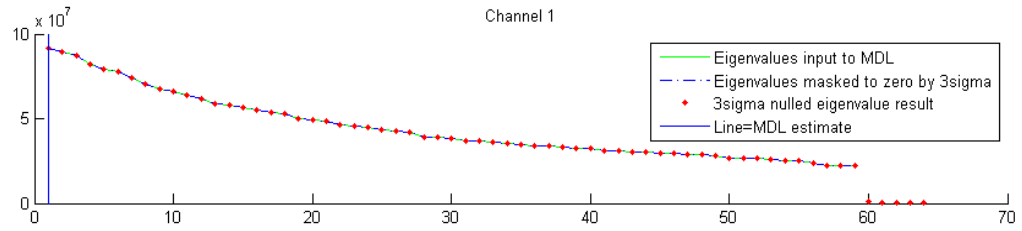

Figure 2 - Eigenvalues of APERTIF covariance data for Virgo A, channel 1 of 71, no RFI present. Array elements 1-59 connected, 60-64 disconnected, resulting in 5 near zero values. These are removed from the sorted list passed to MDL detection (solid green). Points left of vertical line are MDL-detected RFI. Three-sigma thresholding (dashed blue) finds no outlier eigenvalues. The final output after median replacement is identical to the input and has replaced no eigenvalues (red dots).

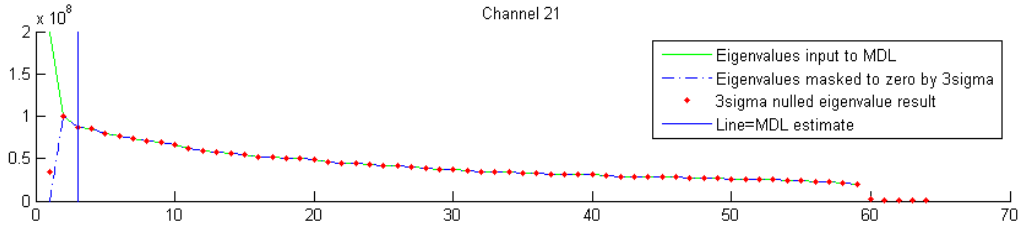

Figure 3 - Eigenvalues of APERTIF covariance data for Virgo A, channel 21 of 71, two RFI sources. Input to MDL detection (solid green) shows strong knee point at 2nd eigenvalue. The two points on left of vertical line are MDL-detected RFI. Three-sigma thresholding (dashed blue) detects only the first eigenvalue. Median replacement assigns median of non-flagged eigenvalues to all flagged eigenvalues. Here only first was replaced. Final result of processed eigenvalues (red dots) is later used to reconstruct a clean covariance matrix for the frequency channel.

After estimating q, nulling and reconstructing a clean covariance matrix, the time series of nulled covariance matrices $\tilde{C}_{xx}(t)$ can be fed into long-term time integration for imaging and spectral line detection or into pulsar data de-dispersion processing.

Strongly detected pulsars may need special care in the processing. Pulsars can contribute dominant eigenvalues to the EVD or SVD decomposition and they may be identified as RFI and get erased.

To demonstrate the algorithm, a result of nulling applied with Matlab and also the C++ Beamformer library to data from APERTIF is shown in Figure 4.

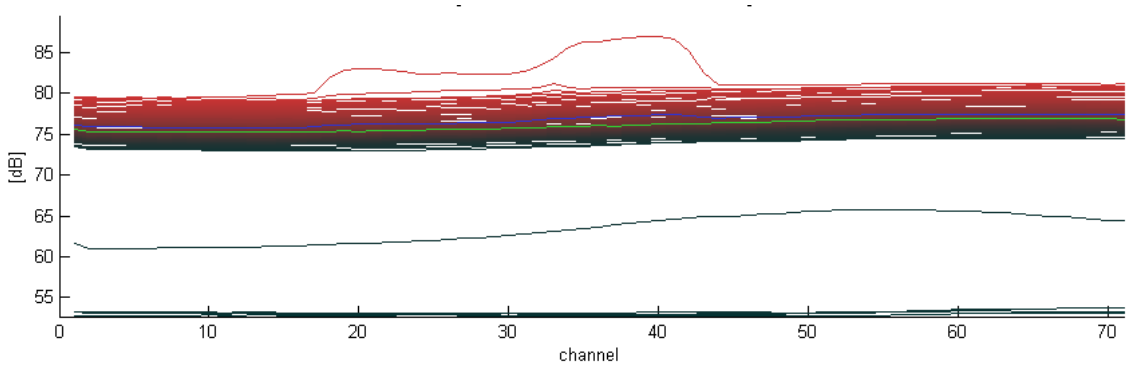

Figure 4 – Eigenspectrum of Virgo A covariances. Derived from 64-element 71-channel raw APERTIF covariance data (Wim van Cappellen). AfriStar digital satellite radio RFI around center channels. This EuroStar-family satellite signal adheres to ETSI EN 302 550-1-3. Horizontal axis: frequency channels 1 to 71 (1.4830-1.4967 GHz). Vertical axis: overlay of all 64 eigenvalue powers, colored red-to-black from largest to smallest eigenvalue. Green curve: median. Blue curve: mean.

The number of interferers q that should be nulled via median eigenvalue replacement is estimated with MDL and with a 3-sigma threshold. The figure shows one interferer detected in the central channels of the 1.4830 - 1.4967 GHz band. It is completely nulled in the processing output show in Figure 5.

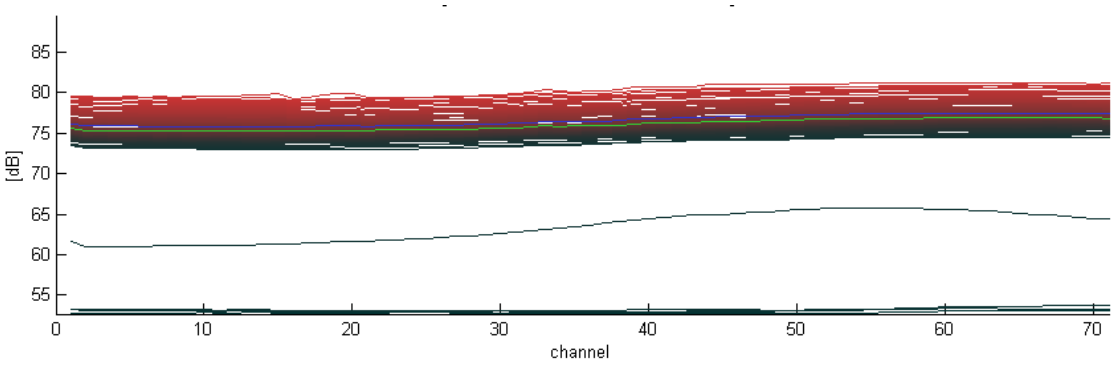

Figure 5 - Eigenspectrum of nulled Virgo A covariances. Original 71x64x64 matrices first SVD or EVD decomposed. Interferer number per channel auto-estimated with MDL and limited by 3-sigma thresholding. Dominant eigenvalues replaced by median of noise space eigenvalues ("nulled"). Nulled matrix decompositions are reconstructed, resulting in clean covariance matrices. Cleaned covariance matrices EVD-decomposed a second time to get eigenspectrum for this figure. Vertical axis: overlay of all 64 eigenvalue powers, colored red-to-black from largest to smallest eigenvalue. Green curve: median. Blue curve: mean.

The effect of MDL and 3-sigma RFI nulling on the autocorrelations is shown in Figure 6. ON-source and OFF-source data were available. Their difference was used to remove standing wave effects of the WSRT dish. Autocorrelations of all 64 elements are plotted along 71 channels.

Nulling is apparently highly effective in reducing the level of RFI in all elements. Nulling also prominently brings out the Virgo A continuum seen in array element 31 in Figure 5. There is a clear processing artifact, however. In element 11 near low channels, nulled OFF-source data contains near zeroes. This causes two spikes after subtraction from ON-source data. Clearly, nulling is not always 100% effective and a last step of data flagging will sometimes be necessary.

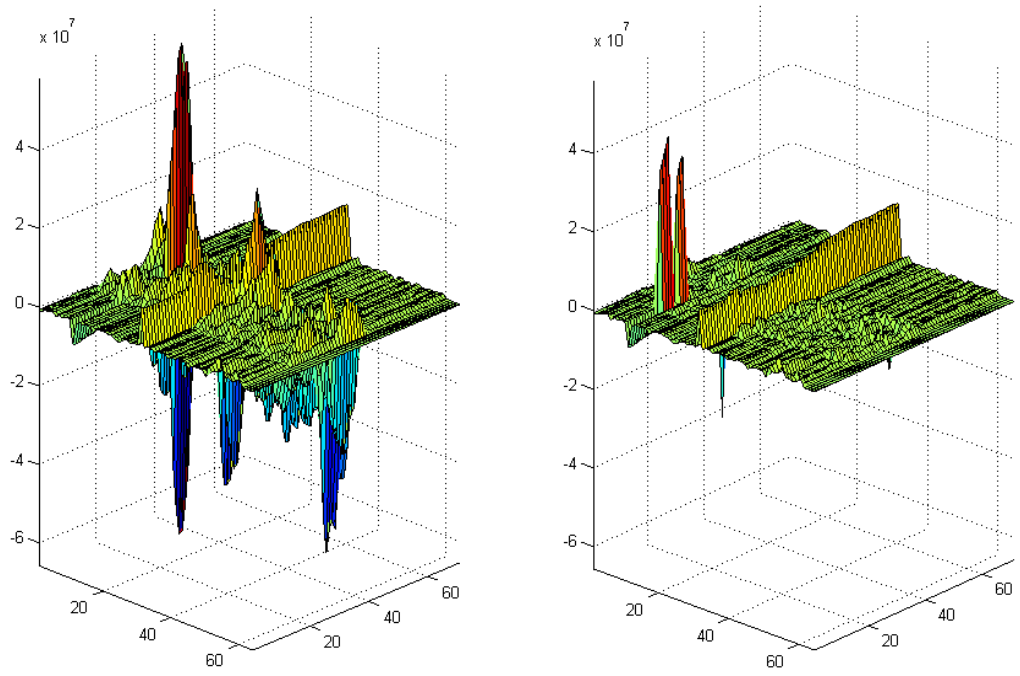

Figure 6 – Autocorrelations from APERTIF Virgo A covariances. Autocorrelations of 64 elements plotted over 71 channels. Left: ON-source minus OFF-source of original data. Right: same for data nulled using MDL and 3sigma thresholding.

For Figure 7 the Matlab UV gridder (part of Matlab portions of the C++ Beamformer library) was run on the APERTIF data, gridding the difference of nulled ON-source and nulled OFF-source covariance matrices, excluding autocorrelations. A final crude UV image was constructed by summing together 2D Fourier -transformed UV images of all 71 APERTIF data channels. The similarly processed and stacked but not RFI-mitigated original data is shown on the left of Figure 7.

In autocorrelation data, Virgo A is seen by off-center element 31. All APERTIF physical beams are non-overlapping. As only one element sees Virgo A it can not be imaged. However, nulled data in Figure 7 reveals a weak point source. It is also present in the original data, but gets masked by RFI in channels 20-45 if these channels are included in the stacking. The point source resides somewhere beyond the main dish and enters APERTIF physical beams through sidelobes. The signal may be caused by the Sun or by wide-band interference from the WSRT control building.

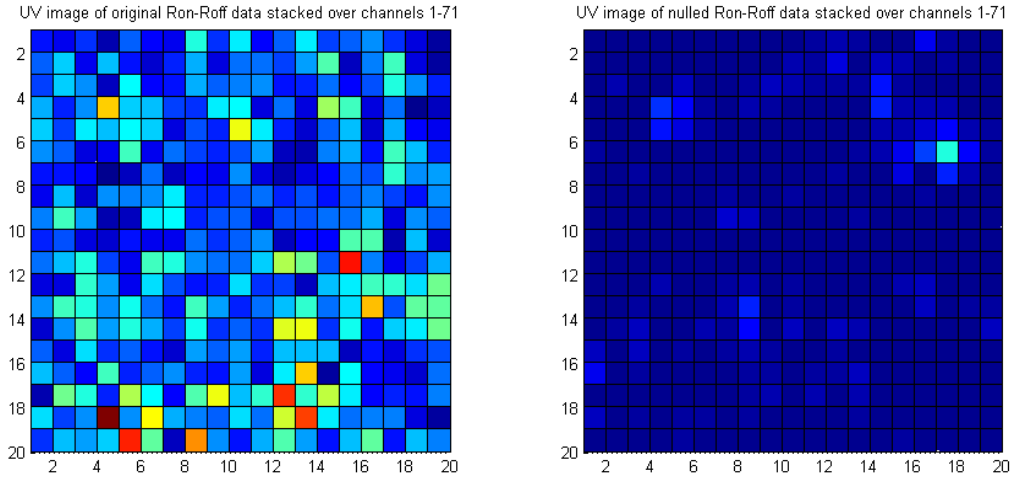

Figure 7 – Imaged covariances after UV gridding, 2D FFT transform and stacking channels 1 to 71. Image FOV approximately ±1 rad. Left: ON-source minus OFF-source difference imaged from original data. Right: same for nulled data. Nulling reveals point source at right image edge. This is likely wide-band RFI entering APERTIF from around the WSRT dish edge. Main dish subtends ±0.96 rad, WSRT has f/D=0.35, D=25m. The point source is seen only in the difference, not in separately imaged ON-source or OFF-source data.

The next three figures show the results of nulling synthetic data. Figure 10 plots a raw RFI-contaminated covariance of a single-channel 64-element array. The array sees two point source interferers and one faint astronomic point source. The RFI-contaminated data of Figure 10 is UV imaged in Figure 8. EVD-based nulling results in cleaned covariance $\tilde{C}_{xx}$ which is UV imaged in Figure 9. Both interferers have been removed while the astronomic point source (and array interference pattern) has been retained. The astronomic point source has identical magnitude as in a completely RFI-free model.

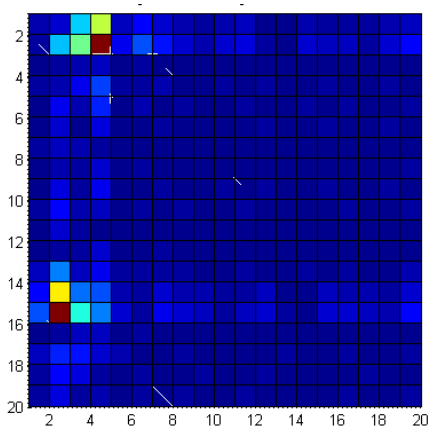

Figure 8 – Image of UV plane of synthetic data. Array is a 64-antenna uniform grid array. Element covariances contain two strong RFI point sources and one weak point source (INR=10^3).

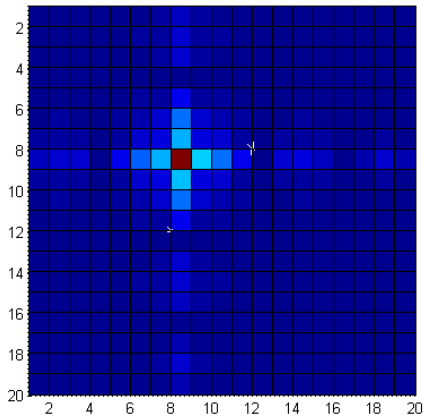

Figure 9 – Image of UV plane, after nulling has been applied to the underlying covariance data of Figure 8. Both RFI sources have been fully mitigated.

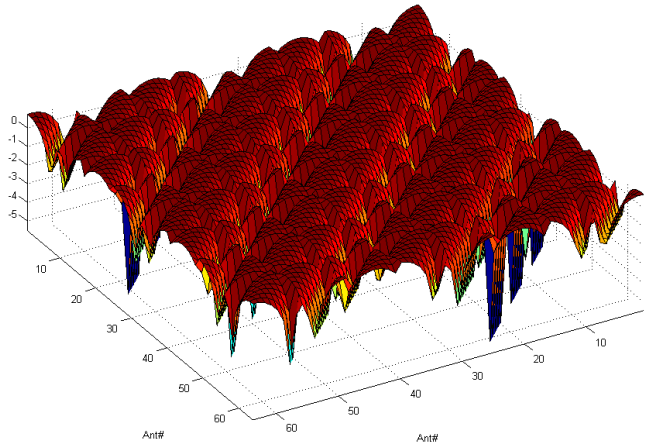

Figure 10 – Magnitude plot of synthetic data from 64x64–element covariance matrix. Array is a 64-element uniform grid array organized as 8*8. Element covariances contain two strong RFI point sources and one weak point source (INR=10^3). Periodicity stems from the square grid (8*8) spatial element layout that is plotted linearly.

Details on Adaptive Beamforming

Beamforming is essentially an adaptive filter process. There is a wealth of algorithms with varying computational complexity. Each algorithm optimal for some specific set of constraints. An extremely comprehensive introduction is S. Werners PhD thesis [WE02].

Beamforming produces a set of complex weights $w = [w_0, w_1, \dots, w_{N_{ant}-1}]$; $\|w\| \equiv 1$ that phase-shifts and combines array element signals $x(t) = [x_0(t), x_1(t), \dots, x_{N_{ant}-1}(t)]$ such that the weighted sum $y(t) = x(t) \cdot w$ 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 $y_b(t) = {w_b}^H \cdot x(t)\ ; b = 0 \dots (B-1)$ from the same array signal vector $x(t)$, allowing the array to look into many directions.

Non-adaptive beamforming simply shifts the phases of the $x(t)$ array signals vector to counteract the wave vector phase shifts of the desired electrical beam steering direction at all element positions. For a 2D array this corresponds to electrically tilting the array plane to align with the incoming plane wave. The beam steering r_b for beam *b* is

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

with array element positions $r_i \in \mathbb{R}^3$ and wave vector k_b of the desired electrical beam direction θ, φ.

From the above, non-adaptive weights are $w_b = {r_b}^*$.

For adaptive beamforming, the weights can be computed from array covariance data estimates $\hat{C}_{xx}$ The weights for maximum power response (minimum mean square error between and actual signal) can be computed as

$$w_{opt,b} = \hat{C}_{xx} r_b \quad (2)$$

Maximum variance minimum distortion beamformer (MVDR, 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

$$w_{MVDR,b} = \frac{\hat{C}_{xx}^{-1} r_b}{{r_b}^* \hat{C}_{xx}^{-1} r_b} \quad (3)$$

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

$$w_{Cox,b} = \left(\frac{{w_{MVDR,b}}^H r_b}{|r_b|}\right)\frac{r_b}{|r_b|} + \alpha \cdot \left(w_{MVDR,b} - \left(\frac{{w_{MVDR,b}}^H r_b}{|r_b|}\right)\frac{r_b}{|r_b|}\right) \quad (4)$$

The choice of α depends on the array. In (4) the $|r_b|$ 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 bandpass.

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.

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

Details on Reference Signal Subtraction

When RFI is captured with low-gain reference antennas that are insensitive to the celestial source, interference can be subtracted nontoxicly from array STI covariance data in a post-correlation step. Compared to its time domain counterpart, adaptive filtering, this method leaves original array signals untouched. Neither does it suffer from pattern rumble. The memory requirements are smaller and computations are also less involved, though complexity is still similar and lower bounded by $\Omega(N_{ant}^2)$.

RFI reference signals have to be cross-correlated against all array elements. This increases the covariance matrix size. It may not always be possible to keep matrix dimensions a convenient power of two.

RFI subtraction from array data is described in van der Veen et al. [VE04] and they present the solution for the generic case. A special case with two reference antennas and at most one interferer per channel is covered in Briggs et al. [BRI00].

We denote signal vector $x(t)$ with N_{ant} signals in total, the first *n* from RFI reference antennas, by

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

Reference antennas see interferers $a(t)$ and noise $n(t)$. The celestial source $v(t)$ is seen only by array elements. Noise is assumed to be i.i.d. for reference antennas. Likewise for array elements. We assume *p* stationary interferers that result in signals a_{ref} and a_{arr} for the two antenna types. Antenna signals

$$x_i(t) = \begin{cases} n_{ref,i}(t) + \sum_p a_{ref,p}(t) & for\ 0 \le i < n \\ v_i(t) + n_{arr,i}(t) + \sum_p a_{arr,p}(t) & for\ n \le i < N_{ant} \end{cases} \quad (2)$$

in vector form are

$$\begin{aligned} x(t) &= [0, \dots, 0,\ v_{arr}(t)^T]^T + \left[n_{ref}(t)^T, n_{arr}(t)^T\right]^T + \sum_p \left[a_{ref,p}(t)^T, a_{arr,p}(t)^T\right]^T \\ &= v(t) + n(t) + A(t) \end{aligned} \quad (3)$$

The short-term integrated covariance estimate

$$\hat{C}_{xx}(t) = \frac{1}{M}\sum_{i=0}^{M-1} x(t+iT) \cdot x^*(t+iT) \quad (4)$$

with uncorrelated noise and variances $\sigma_{\mathrm{ref}}{}^2$ and $\sigma_{\mathrm{arr}}{}^2$, after omitting the time dependence, becomes

$$\hat{\mathrm{C}}_{\mathrm{xx}} = \hat{\mathrm{C}}_{\mathrm{v}} + \hat{\mathrm{C}}_{\mathrm{A}} + \hat{\mathrm{C}}_{\mathrm{N}} = \begin{bmatrix} \mathrm{R}_{\mathrm{rr}} & \mathrm{R}_{\mathrm{ra}} \\ \mathrm{R}_{\mathrm{ar}} & \mathrm{R}_{\mathrm{aa}} \end{bmatrix} = \begin{bmatrix} \mathrm{A}_{\mathrm{ref}}\mathrm{A}_{\mathrm{ref}}{}^{\mathrm{H}} + \sigma_{\mathrm{ref}}{}^2\mathrm{I} & \mathrm{A}_{\mathrm{ref}}\mathrm{A}_{\mathrm{arr}}{}^{\mathrm{H}} \\ \mathrm{A}_{\mathrm{arr}}\mathrm{A}_{\mathrm{ref}}{}^{\mathrm{H}} & \mathrm{C}_{\mathrm{v,v}} + \mathrm{A}_{\mathrm{arr}}\mathrm{A}_{\mathrm{arr}}{}^{\mathrm{H}} + \sigma_{\mathrm{arr}}{}^2\mathrm{I} \end{bmatrix} \tag{5}$$

Here $\mathrm{A}_{\mathrm{ref}} : n \times (N_{ant} - n)$, $\mathrm{A}_{\mathrm{arr}} : (N_{ant} - n) \times (N_{ant} - n)$ and $\mathrm{C}_{\mathrm{v,v}} : (N_{ant} - n) \times (N_{ant} - n)$. A clean estimate of celestial source and noise covariance $\mathrm{C}_{\mathrm{v,v}} + \sigma_{\mathrm{arr}}{}^2\mathrm{I}$ can be formed by subtracting an estimate of $\mathrm{A}_{\mathrm{arr}}\mathrm{A}_{\mathrm{arr}}{}^{\mathrm{H}}$ from R_{aa}.

The interference term of the covariance matrix (5) is

$$\hat{\mathrm{C}}_{\mathrm{A}} = \begin{bmatrix} \mathrm{A}_{\mathrm{ref}}\mathrm{A}_{\mathrm{ref}}{}^{\mathrm{H}} : n \times n & \mathrm{A}_{\mathrm{ref}}\mathrm{A}_{\mathrm{arr}}{}^{\mathrm{H}} : n \times (N_{ant} - n) \\ \mathrm{A}_{\mathrm{arr}}\mathrm{A}_{\mathrm{ref}}{}^{\mathrm{H}} : (N_{ant} - n) \times n & \mathrm{A}_{\mathrm{arr}}\mathrm{A}_{\mathrm{arr}}{}^{\mathrm{H}} : (N_{ant} - n) \times (N_{ant} - n) \end{bmatrix} \tag{6}$$

The submatrix dimensions are noted for convenience. Now when the number of interferers $0 < q \leq n$ and when $\mathrm{A}_{\mathrm{ref}} : n \times q$ has full rank (i.e. interferers are uncorrelated), then $\mathrm{A}_{\mathrm{arr}}\mathrm{A}_{\mathrm{arr}}{}^{\mathrm{H}}$ can be expanded

$$\mathrm{A}_{\mathrm{arr}}\mathrm{A}_{\mathrm{arr}}{}^{\mathrm{H}} = \mathrm{A}_{\mathrm{arr}}\mathrm{A}_{\mathrm{ref}}{}^{\mathrm{H}}\left(\mathrm{A}_{\mathrm{ref}}\mathrm{A}_{\mathrm{ref}}{}^{\mathrm{H}}\right)^{-1}\mathrm{A}_{\mathrm{ref}}\mathrm{A}_{\mathrm{arr}}{}^{\mathrm{H}} \cong \mathrm{R}_{\mathrm{ar}}(\mathrm{R}_{\mathrm{rr}})^{-1}\mathrm{R}_{\mathrm{ra}} \tag{7}$$

This last term can be subtracted to yield a cleaned covariance estimate $\tilde{\mathrm{C}}_{\mathrm{xx}}$:

$$\tilde{\mathrm{C}}_{\mathrm{xx}} = \begin{bmatrix} \mathrm{R}_{\mathrm{rr}} & \mathrm{R}_{\mathrm{ra}} \\ \mathrm{R}_{\mathrm{ar}} & \mathrm{R}_{\mathrm{aa}} - (\mathrm{R}_{\mathrm{ar}}(\mathrm{R}_{\mathrm{rr}})^{-1}\mathrm{R}_{\mathrm{ra}}) \end{bmatrix}, \text{ alternatively } \tilde{\mathrm{C}}_{\mathrm{xx}} = \begin{bmatrix} 0 & 0 \\ 0 & \mathrm{R}_{\mathrm{aa}} - (\mathrm{R}_{\mathrm{ar}}(\mathrm{R}_{\mathrm{rr}})^{-1}\mathrm{R}_{\mathrm{ra}}) \end{bmatrix} \tag{8}$$

The three submatrices that contain reference antenna contributions may be set to zero.

Note the last term in (7) is only approximately equal to the first because of the autocorrelated noise term in $\mathrm{R}_{\mathrm{rr}} = \mathrm{A}_{\mathrm{ref}}\mathrm{A}_{\mathrm{ref}}{}^{\mathrm{H}} + \sigma_{\mathrm{ref}}{}^2\mathrm{I}$; $\sigma_{\mathrm{ref}}{}^2\mathrm{I} > 0$. Further, due to the noise, R_{rr} does not become singular in the interference-free scenario $q = 0$. The noise then adds a small bias to the (non-zero) correction.

The C++ Beamformer library compares means of the reference antenna autocorrelations and the array autocorrelations. It does not do the $(\mathrm{R}_{\mathrm{ar}}(\mathrm{R}_{\mathrm{rr}})^{-1}\mathrm{R}_{\mathrm{ra}})$ subtraction unless the reference mean autocorrelation is 10 times higher than the array mean, which, assuming $\sigma_{\mathrm{ref}}{}^2 \approx \sigma_{\mathrm{arr}}{}^2$, indicates that a higher-INR interference is present ($q > 0$) and subtraction is warranted.

There is another but specialized method for RFI template subtraction by Briggs et al. [BRI00] that does not suffer from the above bias. It requires there are $n = 2$ reference antennas and at most $q \leq 1$ interferers. The method avoids the $\sigma_{\mathrm{ref}}{}^2\mathrm{I}$ noise bias. The corrected covariance value for each $\tilde{\mathrm{C}}_{\mathrm{xx}}(\mathrm{i},\mathrm{j})$ is

$$\tilde{\mathrm{C}}_{\mathrm{xx}}(\mathrm{i},\mathrm{j}) \approx \frac{\hat{\mathrm{C}}(\mathrm{i},\mathrm{a}_2)\hat{\mathrm{C}}^*(\mathrm{j},\mathrm{a}_1)}{\hat{\mathrm{C}}(\mathrm{a}_1,\mathrm{a}_2)} \approx \frac{\hat{\mathrm{C}}(\mathrm{i},\mathrm{a}_2)\hat{\mathrm{C}}^*(\mathrm{j},\mathrm{a}_1)\hat{\mathrm{C}}^*(\mathrm{a}_1,\mathrm{a}_2)}{\alpha(\mathrm{f})+\hat{\mathrm{C}}(\mathrm{a}_1,\mathrm{a}_2)\hat{\mathrm{C}}^*(\mathrm{a}_1,\mathrm{a}_2)} \tag{9}$$

Here $\mathrm{a}_1 = 0$, $\mathrm{a}_2 = 1$ are the indices of the two reference antennas. The parameter $\alpha(\mathrm{f})$ helps numerical stability in those frequency channels where there is no RFI. In those channels, reference antenna cross-

correlation $\hat{C}(a_1, a_2)$ would be close to zero. Choices for $\alpha(f)$ can be a small constant or the square of noise in $\hat{C}(a_1, a_2)$ after calibration.

Example output of the above two subtraction methods (generic and special case) is demonstrated in the figures below. Overall subtraction seems very effective, at least in the synthetic environment setup. An example of subtraction effectiveness with APERTIF data and a reference antenna setup is planned later.

The special case for a single RFI signal and two reference antennas is in Figure 10 to Figure 14. The general case with two and three RFI signals (combined with two and three reference antennas) is demonstrated in Figure 15 to Figure 18.

To see how both methods affect the covariance data if no interference is present, the error between subtraction-cleaned covariances and the original (RFI-free) covariance is plotted Figure 19. The signal power threshold check (comparison of reference and array autocorrelation means) was disabled, forcing the subtraction algorithms to run. The resulting difference in cleaned and true clean covariance data is zero. Varying the levels of element noise powers and spurious cross-correlated noise in relation to the astronomic signal results in similar zero or practically zero error.

With RFI present, the noise-induced bias of the generic algorithm becomes clear. Figure 20 compares the case with two reference antennas and one RFI (Nref=2, Nrfi=2). The peak error is about a factor 10^{-5} smaller in the special method. The error in the generic method improves when there are less interferers than reference antennas (Figure 21).

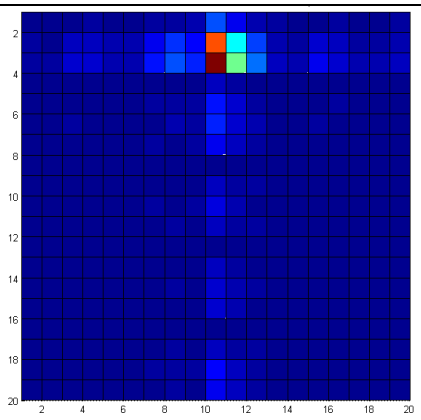

Figure 11 – UV image of synthetic data. Phased array 8*8, N_{ref}=2 antennas at diagonal corners, relative gain towards RFI g=10^3. Signals: 1 strong RFI, 1 weak sky point source.

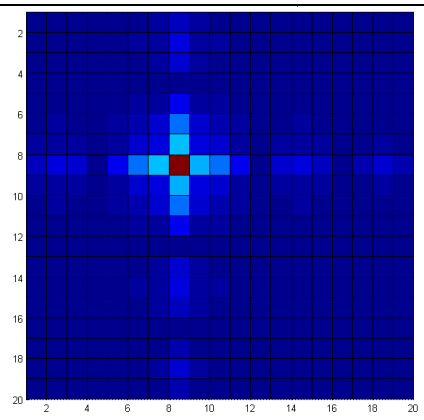

Figure 12 – UV image after RFI Nulling. Covariance underlying Figure 11 was nulled, excluding RFI reference antenna data.

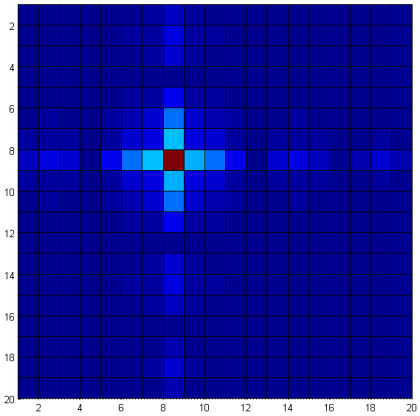

Figure 13 – UV image after specialized RFI Subtraction (1 RFI/channel, N_{ref}=2). Practically identical results with g=10^1 … 10^3 relative gains towards RFI. Result comparable to Nulling.

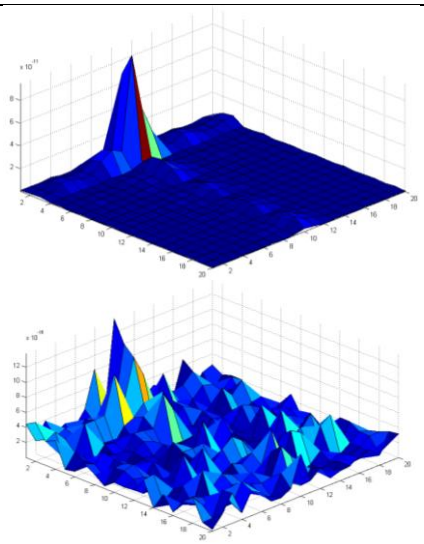

Figure 14 – Delta UV image at two RFI antenna gains. Difference between RFI-Subtracted covariance UV image and a reference RFI-free model UV image. Top: RFI antenna gain 10^1, vertical scale 0 to $9 \cdot 10^{-11}$. Bottom: RFI gain 10^3, vertical scale 0 to $13 \cdot 10^{-15}$.

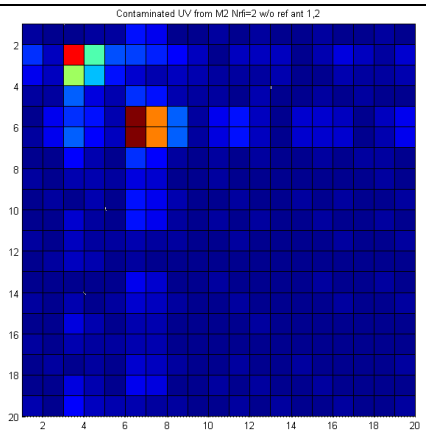

Figure 15 - UV image of synthetic data. Phased array 8*8, N_{ref}=2 antennas at diagonal corners, relative gain towards RFI g=10. Signals: 2 strong RFI, 1 weak sky point source.

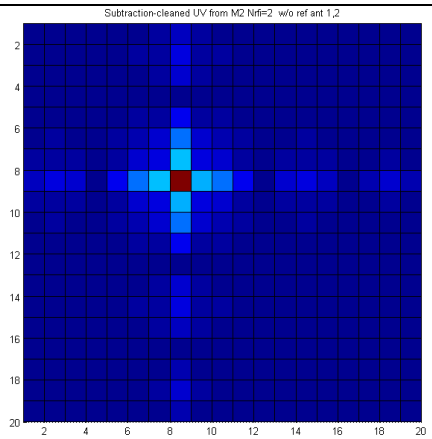

Figure 16 – UV image after RFI Subtraction (generic 2 RFI/channel, N_{ref}=2). Ref antenna gain towards RFI g=10.

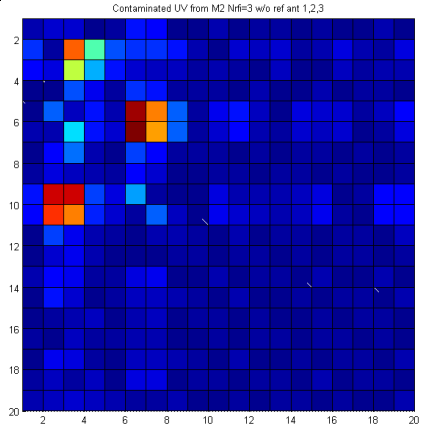

Figure 17 - UV image of synthetic data. Phased array 8*8, N_{ref}=3 antennas near array corners, relative gain towards RFI g=10. Signals: 3 strong RFI, 1 weak sky point source.

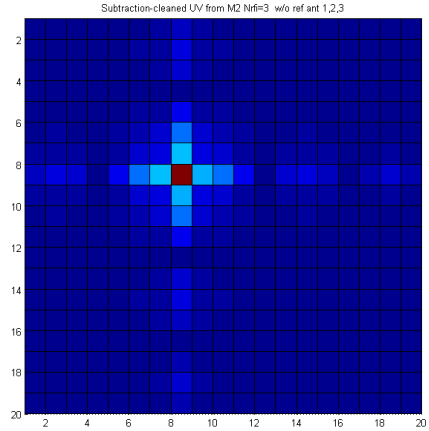

Figure 18 - UV image after RFI Subtraction (generic 3 RFI/channel, N_{ref}=3). Ref antenna gain towards RFI g=10.

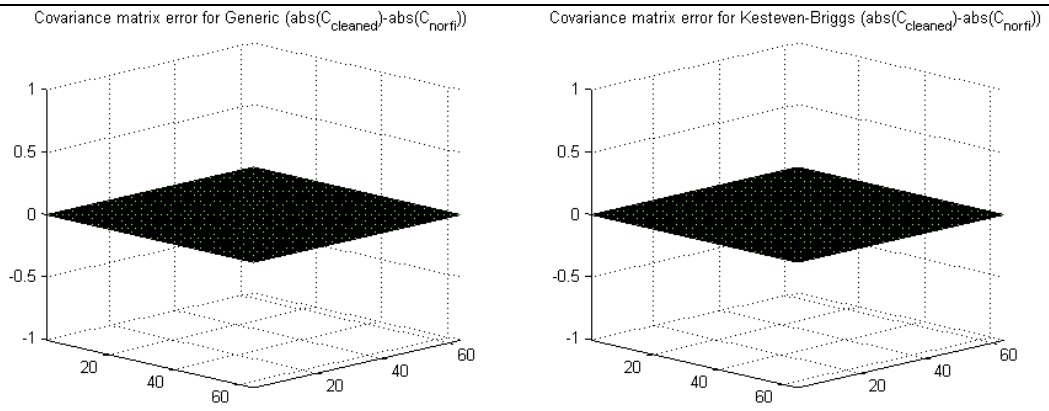

Figure 19 – Absolute difference between Subtraction cleaned covariance and an RFI-free model covariance. Two reference antennas (N_{ref}=2), no RF interference (N_{rfi}=0). Left: generic Subtraction method. Right: specialized Subtraction method. Data is zero to working precision even when pre-subtraction interferer level checking is bypassed.

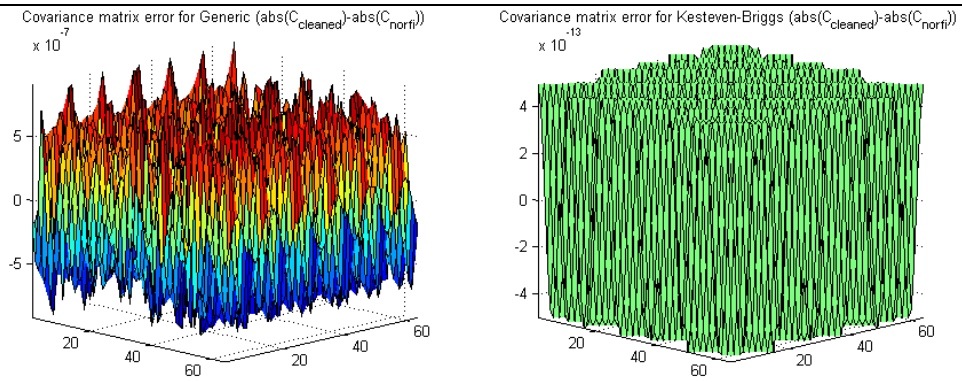

Figure 20 - Absolute difference between Subtraction cleaned covariance and an RFI-free model covariance. Two reference antennas and one RFI (N_{ref}=2, N_{rfi}=2). Left: generic Subtraction, scale 10^{-7}. Right: special Subtraction, scale 10^{-13}.

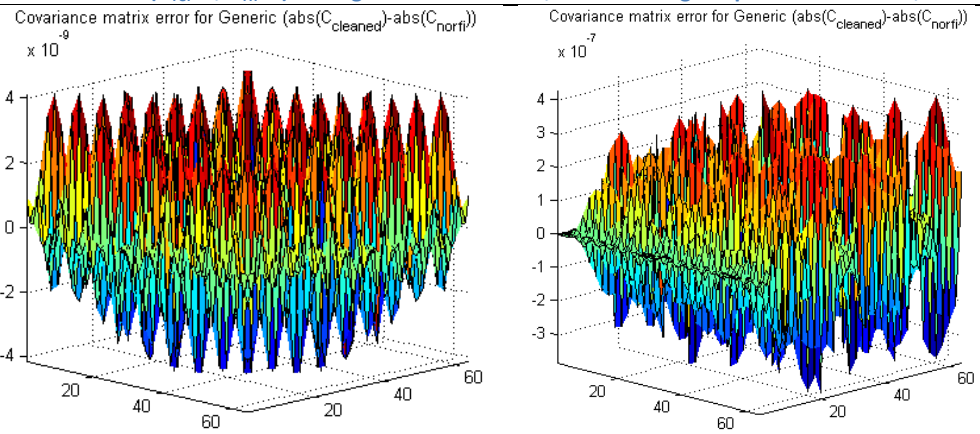

Figure 21 - Absolute difference between Subtraction cleaned covariance and an RFI-free model covariance, N_{ref}=3. Left: generic Subtraction and two RFI (N_{rfi}=2), scale 10^{-9}. Right: generic Subtraction and three RFI (N_{rfi}=3), scale 10^{-7}.

Credits and Future work

Wim van Cappellen from ASTRON kindly provided the APERTIF 1.49 GHz covariance snapshot (Virgo A with prominent AfriStar satellite RFI). This covariance data was used to demonstrate Nulling methods.

Regarding future work, it may be of interest to consider the Jeffs and Warnick [JW09] findings on spectral bias ("spectral scooping") for narrowband interference. Spectral bias is caused by beamformer weights that are calculated from a covariance matrix that is not the exact covariance matrix but an estimate. Spectral bias may be of interest mainly for PSD estimation and correction.

A further TODO might be to integrate functions of this library into AIPS or CASA. For CASA this is reasonably straight forward. Non-pretty details of Fortran access to C++ classes would need to be handled for AIPS, or AIPS visibility data modified through ParselTongue and the use of Numeric Python as well as rewriting the computations into Python form. However, at the moment it is still unclear how CASA or AIPS will support small local arrays or focal plane arrays. Most arrays have their own toolset and pipeline.

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