INTERNAL REPORT

HoP User Guide

F. Buffa[1], G. Serra[1] and S. Poppi[1]
[1]INAF - Osservatorio Astronomico di Cagliari
Report N. xx, Released: 25/12/2017
Reviewer: C. Migoni

OAC Osservatorio Astronomico di Cagliari

Authors

author	email	contribution
Franco Buffa	franco.buffa@inaf.it	code development
Giampaolo Serra	giampaolo.serra@inaf.it	system development
Sergio Poppi	sergio.poppi@inaf.it	system development

Acronyms

AS	Active Surface
HoP	Holography Package
LUT	Lookup-Table
MHS	Microwave Holography System
PR	Primary Reflector (aka M1)
RA	Reference Antenna
RT	Radio Telescope
SRT	Sardinia Radio Telescope

HoP User Guide

Version 1.0

edited by
F. Buffa, G. Serra and S. Poppi

October xx, 20xx

INAF – Osservatorio Astronomico di Cagliari, Via della Scienza 5, 09047 Selargius (CA), Italy

Contents

1 Introduction

HoP stands for Holography Package of the Sardinia Radio Telescope (SRT) [1, 2]. It is a software implementing the well-established phase-coherent microwave holography method for the surface diagnostic of the high gain antennas [3]. This method allows to map the surface deformations of a large reflector antenna by exploiting the well-known[1] Fourier transformation relationship between the aperture field and the far-field pattern of this type of antennas. Therefore, given the complex far-field pattern of a high gain antenna, one can compute the map of the surface deformations (i.e. the holographic map) with respect to the ideal parabolic profile, by carrying out a pipeline of several functions among which the two dimensional inverse fast Fourier transformation is the core.
This is what HoP carries out by means of a pipeline composed of a set of GNU Octave[2] scripts. HoP can currently process the data set measured by the SRT microwave holographic system (MHS). MHS was installed firts at Medicina radio telescope [4] and then at SRT [5] to measure the telescope far-field pattern by an on-the-fly raster scan around a geosinchronous satellite. Minor changes are needed to make HoP able to process whatever holographic data set. Since the SRT is provided with an active surface (AS), surface deformations due to the telescope self-weight and panel misalignments may be corrected by operating the 1116 actuators[3] placed under the 1008 panels[4] of the primary reflector (PR). For this reason, the HoP pipeline ends providing a look-up table (LUT) of the corrections at the elevation at which the telescope far-field pattern is measured. The LUT can be used by the SRT antenna control software to command the AS actuators to a new position long the direction parallel to the antenna axis.
This document deals with describing the HoP structure and providing its user guide. After a short theoretical formulation of the microwave holography method, Section 2 explains how to plan a holographic measurement starting from the parameters of the antenna under test and the requirements on the resolution and accuracy of the final holographic map. The MHS output data, i.e. the holographic data set (the input for HoP) is then described in Section 3. The structure and the installation of the HoP pipeline is outlined in Section 4. Moreover, a detailed description of the parameter list of the configuration file and the function list in the HoP pipeline is reported in Section 5 and 6 respectively. Finally, a step-by-step istructions to process a holographic dataset from scratch is outlined in Section 7. **Finally.... future version of HoP package will be discussed....**.

2 Holography in a nutshell

Let $T(u,v)$ be the far-field radiation pattern of a large reflector antenna, i.e. a complex function related to the antenna aperture field by a two dimensional inverse Fourier transformation ($\mathcal{F}^{-1}$). By calculating the phase distribution of the aperture field, one can derive the deformations on the reflector surface $\varepsilon(x,y)$ by [6]:

$$\frac{\varepsilon(x,y)}{\lambda} = \frac{1}{4\pi}\sqrt{1+\frac{x^2+y^2}{4f^2}}\,Phase\left(e^{i2kf}\mathcal{F}^{-1}(T(u,v))\right) \tag{1}$$

where λ is the wavelength at which the far-field pattern is measured, f the focal length of the ideal parabolic surface and $k = 2\pi/\lambda$. The coordinates (x,y) locate a generic point of the parabolic surface projected on the aperture plane in a cartesian reference system having the origin on the focal point.The radiation pattern coordinates (u,v) are related to the antenna reference system (x_h,y_h,z_h) by (see figure 1) [6]:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_h \\ y_h \\ z_h \end{bmatrix} \tag{2}$$

[1] From the antenna theory.

[2] Octave (`https://www.gnu.org/software/octave/`) is a mathematics-oriented language highly compatible with Matlab.

[3] With a maximum stroke of ±15 mm.

[4] Panel size ranges from 2.4 to 5.3 m^2.

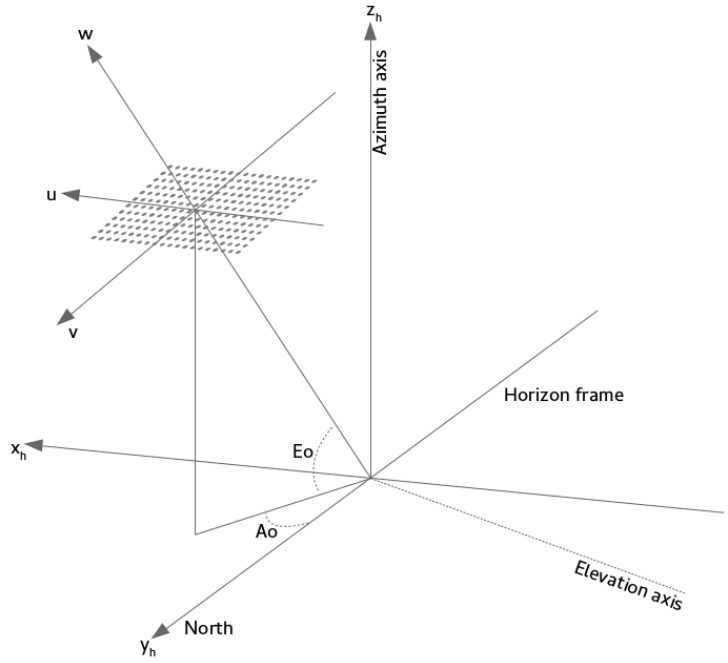

Figure 1: Horizon and (u,v,w) frames for an A/E mount.

where ϕ and θ are the Euler angles. For an azimuthal/elevation (hereafter A/E) telescope mount, we can write:

$$\begin{aligned} x_h &= \cos(E)\sin(A) \\ y_h &= \cos(E)\cos(A) \\ z_h &= \sin(E) \end{aligned} \tag{3}$$

In Figure 1, the horizontal reference frame (x_h,y_h,z_h) and the (u,v,w) one are defined together with A_o and E_o, i.e. the azimuth and elevation coordinates of the satellite position[5], which are related to the Euler angles by $A_o = -\phi$, $E_o = \theta + \pi/2$.
As a general rule, the holographic campaign must be carefully planned in order to achieve the expected performances in terms of spatial resolution and phase distribution accuracy on the antenna aperture plane. First of all, a suitable sampling interval (SI) must be chosen to measure the far-field pattern in order to avoid aliasing effects [6]. SI is:

$$SI[^\circ] = \alpha \frac{\lambda}{D} \frac{\pi}{180} \tag{4}$$

D is the antenna diameter and α is an over-sampling parameter, empirically chosen between 0.5 and 1, useful to avoid the replica overlapping on the aperture plane domain after the $\mathcal{F}^{-1}$ calculation. Other holographic experiments [7] show that a suitable α is ≤ 0.83. During the SRT holographic campaign ($\lambda = 0.026m$ and $D = 64m$) we got $\alpha = 0.77$ and then $SI \sim 0.018^\circ$.

Given α, the number of grid points in the antenna radiation pattern map $(N \times N)$ is related to the resulting spatial resolution and to the number of pixels in the aperture plane $(M \times M)$ by the following

[5]around which the telescope far-field pattern is measured.

relation:

$$N = 360M/(\alpha\pi) \tag{5}$$

Then, if we aim to achieve a spatial resolution of ∼1.4 m ($M = 64/1.4 = 46$) for the 64 m diameter SRT, we will have to set a 65×65 pixel radiation pattern map. The surface error accuracy ($\delta\varepsilon$) is expressed as:

$$\delta\varepsilon = \lambda N/(4\pi SNR) \tag{6}$$

A signal-to-noise ratio (SNR) better than 63 dB is required for the SRT holographic system in order to get $\delta\varepsilon \sim 100\mu m$.
With respect to the Fourier transform in equation 1, if the data are not uniformly distributed in the u-v space, it will be necessary to regularize them via an interpolation procedure. The interpolation sampling intervals are typically chosen to be:

$$\Delta u = \Delta v = \kappa_o \lambda/D \tag{7}$$

where κ_o is a sampling factor introduced with the aim to minimize the aliasing effects ($0.5 \leq \kappa_o \leq 1$).
Note that the grid must be zero-padded before to perform the Fourier transform. We typically use 512×512 grids ($M_p = 512$). Therefore after the transform we get a map with the following pixel resolution:

$$\Delta x = \Delta y = D/(\kappa_o(M_p - 1)) \tag{8}$$

We obtain $\Delta x = \Delta y = 0.157$ [m/px] by adopting $\kappa_o = 0.8$.

3 Raw-data file

The SRT holographic backend data are recorded into 12 column file having the following format:

Col 1: azimuth coordinate [°]

Col 2: elevation coordinate [°]

Col 3: RT total power

Col 4: RA total power

Col 5: RA quadrature total power

Col 6: RT-RA correlation (I-Cross-correlation)

Col 7: RT-QRA quadrature correlation (Q-Cross-correlation)

Col 8: correlation percent

Col 9, 10 and 11: not used

Col 12: time stamp

The measurements are typically performed as azimuth scans. The satellite position is estimated by means of precise ephemerals. At the ** beginning ** and the end of each azimuth line the schedule executes a calibration measurement on the satellite[6], ** before changing the antenna elevation position for a new line scan **. The SRT radiation pattern $T(Az, El)$ is obtained combining the columns 4, 5, 6, and 7: $Real(T(Az, El)) = Col6/\sqrt{Col4}$ and $Imag(T(Az, El)) = Col7/\sqrt{Col5}$.

[6]The on-source calibration measurements are performed in order to record the time-dependent amplitude and phase variations.

4 HoP structure

HoP is designed as a set of Octave functions to be called in sequence. Basically each function performs one, or more, simple operations: it loads an input file (created by the previous function call), it does something and it saves an output file (the input for the next call) ** prova ad esprimere meglio la frase precedente **. No supplementary information are needed, apart for the configuration file shared by all the scripts.
HoP performs basically three main tasks:

- pre-processing of the far-field pattern;
- aperture field calculation and aberration analysis;
- LUT estimation.

In the pre-processing, HoP extracts from the timestamped raw-data files ** (the MHS output data) ** the antenna azimuth and elevation position during the scan, the azimuth and elevation of the satellite (on-source calibration data) and the phase and amplitude computed by the correlator. These data are used to create a uniformly spaced grid, optionally corrected by the source phase and amplitude drifting effects.
In the aperture field calculation, HoP performs a 2D-Fourier Transformation of the far-field pattern and the phase ambiguity fixing ** adjustment **. Then, the antenna surface aberrations are (if needed) analyzed in terms of Zernike polynomials.
Finally, the aperture phase is modeled, panel by panel, by means of a bi-variate polynomial fitting considering all the data points falling within the range of each panel. The surface deflections for each actuator (i.e. the LUT) are obtained as the average of four/two polynomials calculated at the actuator position, as each actuator shares four/two panels ** prova a spiegarlo meglio **.

4.1 Installing HoP

If you are familiar with the GNU Octave environment you know that Octave has been built highly compatible with respect to Matlab. This means that, in principle, the HoP routines could be run within Matlab with a few minor changes. At the current development phase we still have not tried to run HoP in Matlab, but you are encouraged to try it.
Installing HoP is easy. You can download the package ** from `https://www.ict.inaf.it/gitlab/pippo/pluto`. Then, you just need to put the */hop/dev* folder, containing the HoP functions, somewhere in your disk and to make it "visible" to Octave:

```
octave:1> addpath('/home/franco/hop/dev')
octave:2> savepath
```

This creates (or updates) the $\sim$ */.octaverc* file which contains the Octave path customized by the user.

```
franco@tasinanta:~> cat .octaverc
## Begin savepath auto-created section, do not edit
  addpath ('/home/franco/hop/dev', '-begin');
## End savepath auto-created section
```

If you decide that your working folder is *hop* it could also contain your project folders. Each project folder contains all you need to elaborate a single holographic measurement: a configuration file (with .m extension) and a raw-data file.

5 Configuration file

The HoP functions share the same configuration file. The configuration file is an Octave file (.m) whose name is the only parameter that can be passed by argument to the functions. If no argument is passed, the functions will try to open *conf.m* by default. The configuration file contains all the parameters to be set in order to elaborate the holographic data (see table 1).

Parameter	Type	Description	Used by
filename	string	raw-data file name	preproc ql
datapath	string	SRT AS folder	lut zern
mp	int	map resolution [pixel]	all functions
scn	string	select scan type	preproc
source	boolean	on-source calibration [true/false]	preproc
nma	int	phase smoothing time-lag	preproc
azcut	float array	az. window parameters [°]	preproc uv
elcut	float array	el. window parameters [°]	preproc uv
dam	float array	azimuth/elevation velocity threshold	preproc
sgm	int	window function parameter	phase
trs	int	amplitude threshold	phase
panels	int	display actuators/panels	errmap zern
lambda	float	wavelength	errmap phase preproc uv
D	float	antenna diamenter	errmap lut phase uv
k0	float	sampling factor	errmap lut panelfit phase uv zern
F	float	focal length	errmap phase
tol2	float	mask threshold	phase
r0	int	sub-reflector radius	phase
r0_tol	float array	mask parameters	phase
r1	int	antenna radius	phase zern
pr	int	map center y [px]	phase
pc	int	map center x [px]	phase
i1	int	unwrapping starting point y [px]	unwr unwr1
j1	int	unwrapping starting point x [px]	unwr unwr1
n2	int	unwrapping iteration limit	unwr unwr1
tol	int	unwrapping tolerance	unwr unwr1
aber	int	aberration correction	errmap
cmap	string	colormap	errmap zern
tol3	int	errmap colormap cutoff	errmap
tol4	int	zern colormap cutoff	zern
sfactor	float	scale factor	errmap
zp	int array	Zernike indexes	zern
npan	int	number of panels	lut
nact	int	number of actuators	lut

Table 1: HoP setting variable list.

5.1 Parameter description

A short description of the HoP parameters is provided herein:

filename: Raw-data file name. It may include absolute or relative path.

datapath: Absolute path to the folder containing the panel/actuator topology files.

mp: Map pixel resolution.

scn: *scn*='az' for azimuth scan, *scn*='el' for elevation scan.

source: Apply the source calibration data correction [true/false].

nma: Moving average time-lag used to smooth the satellite data. The satellite phase and amplitude are used to correct the data if $source = true$.

azcut: Float array containing three parameters [°]: the azimuth coordinate of the satellite, the map azimuth half-width (all point outside this window will be rejected) and the satellite azimuth position uncertainty.

elcut: Float array containing three parameters [°]: the elevation coordinate of the satellite, the map elevation half-width (all point outside this window will be rejected) and the satellite elevation position uncertainty.

dam: Float array containing two parameters defining the azimuth scanning velocity upper and lower limits [°/time]. The grid points are selected in terms of their azimuth velocity, this criterion allows to label the point collected during the antenna slewing (outliers) or during the on-source pointing (calibration data).

sgm: σ parameter [pixel] of the Gaussian function used as window function for the Fourier transform.

trs: Amplitude threshold [%]. The amplitude map is centered by means of a fitting, only the grid points with amplitude $> trs$ are considered.

panels: If $panels$=1 the actuators are represented in the map, while if $panels$=2 the panels are shown. If you don't want any graphic symbol set $panels$=0.

lambda: Wavelength [m].

D: Antenna diameter [m].

k0: Sampling factor introduced in equation 7.

F: Focal length [m].

tol2: Antenna blocking parts (quadrupode) threshold. All the map parts with amplitude $\leq tol2$ will be masked (required by the unwrapping algorithm).

r0: Radius [pixel] of the sub-reflector. This parameter allows to mask the sub-reflector. It is required by the unwrapping algorithm ($r0 = R_{M2}/\Delta x = 4.5/0.157 = 29$ pixels).

r0_tol: Float array containing two parameters needed to properly set the sub-reflector mask. The first parameter, $r0\_tol(1)$ [pixel], defines a region ranging from $r0$ to $r0 + r0\_tol(1)$, the second one, $r0\_tol(2)$, is the amplitude threshold to be applied in this circular region. This parameter is required by the unwrapping algorithm to deal with the sub-reflector blockage area.

r1: Antenna radius [pixel]. This parameter allows to mask the map region outside the antenna. It is required by the unwrapping algorithm ($r1 = R_{M1}/\Delta x = 32/0.157 = 204$ pixels).

pr: Pixel y (row) coordinate of the map center[7]. If both *pr* and *pc* are set to zero, the map center is evaluated automatically.

pc: Pixel x (column)coordinate of the map center. If both *pr* and *pc* are set to zero, the map center is evaluated automatically.

i1: Row coordinate of the map point (i1,j1), which is the starting point for the recursive unwrapping algorithm.

j1: Column coordinate of the map point (i1,j1), which is the starting point for the recursive unwrapping algorithm.

[7]HoP adopts the upper-left coordinate system convention for the grids.

n2: Iteration limit for the recursive unwrapping algorithm.

tol: Threshold for the unwrapping algorithm. The phase ϕ is corrected if $\Delta\phi > tol \cdot \pi$.

cmap: Set the current colormap (the Octave function $colormap('list')$ returns a list with all of the available colormaps).

aber: Aberration parameter. If $aber = 0$ a plane is subtracted from the unwrapped phase map, if $aber = 1$ a plane + defocusing + feed displacement effects are subtracted. If $aber = 2$ also the astigmatism is considered.

tol3: Cutoff term [mm] for the color bar used by *errmap* for the map plot.

tol4: Cutoff term [mm] for the color bar used by *zern* for the map plot.

sfactor: Multiplicative factor for *errmap*. Such parameter is useful to calibrate the error map.

zp: Zernike polynomials indexes required by the *zern* function.

npan: Number of panels ($npan = 1008$).

nact: Number of actuators ($nact = 1116$).

5.2 Configuration file example

The configuration parameters are Octave variables: integer, float, array, etc. The parameters may be inserted in whatever order. If a parameter is changed during the processing, you should restart your analysis, as such variables are shared by different functions.
A typical HoP configuration file is shown below:

```
filename='./holo_20171005_174034';
datapath='/home/franco/hop/dev/data/';
%
scn='az';
mp=512;
nma=50;
source=true;
azcut=[183.38 2 0.1];
elcut=[44.245 2 0.1];
dam=[0.005 0.02];%
%
cmap='jet';
sgm=50; %34;
trs=15;
panels=0;
%
lambda=0.027;
D=64;
k0=0.8;
F=21.0236;
tol2=0.0003;%0.0004;
r0=29; %29;
r0_tol=[1 0.002];%[8 0.004];
r1=205;
%
```

```
pr=25;
pc=-20;
%
i1=350;
j1=250;
n2=150000;
tol=0.8;
tol3=3;
tol4=2.5;
aber=0;
sfactor=-1;
%
zp=[1 2 3 4 6 7 8 9 10];
%zp=1:10;
%zp=1:28;
%
npan=1008;
nact=1116;
%
```

Note that each variable declaration ends with ";" as usual in the Octave environment. Comments ("%") may be inserted freely in the code in order to increase the file readability.

6 Function list

Apart for the configuration file (parameter file) the only required input is the raw-data file. The functions are sequentially called until the LUT is created. Each function needs one (or more) input file created by the previous one. The names of intermediate files are hard-coded so you can't modify them. Each function creates one output files required by the next elaboration step.
The lut function needs further input files describing the primary mirror panel geometry (see datapath parameter).

Function name	Purpose	Input	Output
ql	quick-look	raw-data file	
preproc	data striping	raw-data file	formatted far-field data
uv	uv space projection	formatted far-field data	gridded data
phase	inverse Fourier transform	gridded data	wrapped phase
unwr	quick phase unwrapping	wrapped phase	unwrapped phase
unwr1	precise phase unwrapping	wrapped phase	unwrapped phase
errmap	error map estimation	unwrapped phase	error map
zern	aberration analysis	error map	corrected error map
lut	look-up table estimation	corrected error map	LUT

Table 2: Function list.

6.1 ql

Purpose: data quick-look

- *input files*: raw-data
- *output files*:
- *plots*: azimuth and elevation speed plots, Az/El map

ql is used to check the raw-data file. The raw-data file contains three data types: the map points, the source calibration data and (perhaps) outliers. All those data must be correctly filtered and interpreted by the next call: *preproc*. The *ql* plots allow you to estimate some useful parameter required by *preproc* as *azcut*, *elcut* and *dam*.

6.2 preproc

Purpose: data striping

- *input files*: raw-data
- *output files*: azel.txt
- *plots*: azimuth and elevation speed plots, Az/El map, amplitude and phase source drift

preproc creates the azel.txt file required by *uv* to compute the uv-map. It contains four data columns: azimuth, elevation and real and imaginary parts of the map points. The source signal is used to correct the map for source drift effects (see *source* and *nma*). As first step, *preproc* considers the antenna azimuth (or elevation) angular velocity (see *dam* and *scn*) in order to extract the map data. Once the "true" map data are labeled, the code extracts the source data. The calibration source data are selected within the window defined by *azcut* and *elcut*. All the points outside this window are considered as outliers and expunged.

6.3 uv

Purpose: data gridding

- *input files*: azel.txt
- *output files*: uuvv.dat
- *plots*: antenna radiation pattern

uv creates the uuvv.dat file which contains the gridded antenna radiation pattern in the uv space.

6.4 phase

Purpose: inverse Fourier transform

- *input files*: uuvv.dat
- *output files*: phase.dat
- *plots*: Gaussian fit results, amplitude and wrapped phase maps

phase creates the phase.dat file containing the wrapped phase data. The *trs* parameter is used to select the amplitude threshold (in %) for the Gaussian fitting applied to center the map in uv space. Alternatively the map center coordinates (in px) may be specified by *pr* and *pc*. Only if $pr = pc = 0$ the map center is determined by the Gaussian fitting. *phase* needs some parameter to define the blocking antenna parts (sub-reflector and struts). Such parameters define the amplitude thresholds below which the map is masked (see figure 2).

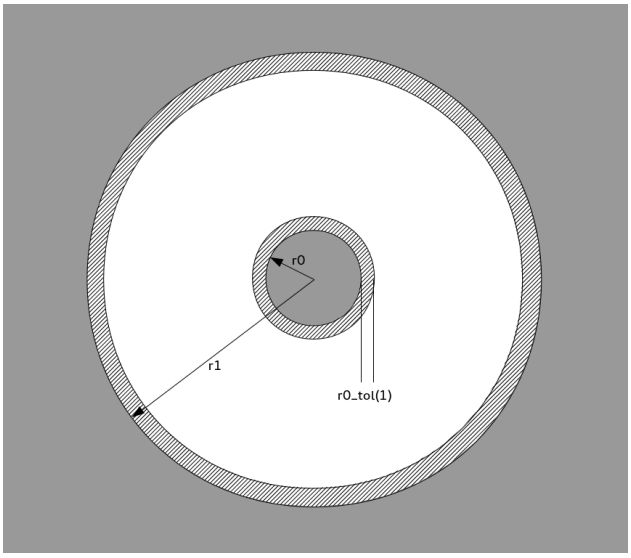

Figure 2: Only the circular region $r0 \leq r \leq r1$ is considered by HoP for calculations. The unwrapping algorithm requires the masking of the blocking parts within this area by means of $tol2$. Near the sub-reflector you can use $r0\_tol$ to refine the mask near the sub-reflector, $r0\_tol(1)$ defines the masking area, $r0\_tol(2)$ the amplitude threshold.

Furthermore the *sgm* parameter (px) defines the window function radius used as Fourier transform tapering function.
The *phase* printout shows few information:

```
dx=0.156556 (m/px)
r0=4.540117 (m)
r1=32.093933 (m)
dfx=-2.818004, -18 (m, px)
dfy=2.974560, 19 (m, px)
df=4.097457 (m)
```

Where dx is the map pixel size, r0 is the sub-reflector radius (see $r0$), r1 is the primary mirror radius (see $r1$), dfx and dfy are the map center offset calculated by the Gaussian fitting (or set by means of *pr* and *pc*).

6.5 unwr

Purpose: phase unwrapping

- *input files*: phase.dat
- *output files*: unwr.dat
- *plots*: unwrapped phase map

unwr creates the unwr.dat file containing the unwrapped phase data. The $n2$ parameter limits the number of iterations of the recursive algorithm. The i1 and j1 are the pixel coordinates of the starting point. The starting point must be carefully chosen near the map center (where the amplitude is high) and far from the blocking parts (sub-reflector, struts, etc.).

6.6 unwr1

Purpose: phase unwrapping
$unwr1$ is slower than *unwr*, but more reliable. Use $unwr1$ with the same strategy adopted for *unwr*.

6.7 errmap

Purpose: error map estimation

- *input files*: unwr.dat
- *output files*: errmap.dat
- *plots*: error map

errmap creates the errmap.dat file containing the error map grid from the phase map. Because the sign convention adopted in deriving equation 1, a negative phase error means the surface is locally higher, while a positive phase error means the surface is locally lower. The $sfactor$ parameter allows you to set the error map sign. If $sfactor = -1$ you will obtain a phase error map, while if $sfactor = 1$ you will obtain a correction map (the LUT). Note that such parameter may be used as scale factor (if known) to weight the entire map. A common procedure consists in moving one or more panels of a known amount in order determine the scale factor.
The *aber* parameter allows you to subtract first- and second-order aberration terms due to optics' misalignments. The term P to be subtracted from the phase distribution is obtained as a polynomial fitting [8, 9]:

$$P = a_1 + a_2 x + a_3 y + a_4 \rho^2 + \rho^2(a_5 x + a_6 y) + a_7 \sigma^2 \tag{9}$$

Where a_k are the fitting parameters, $\rho = \sqrt{x^2 + y^2}$ and $\sigma = \sqrt{x^2 - y^2}$.
The feed displacement coordinates may be derived from the fitting results as:

$$\begin{aligned} x' &= 2a_5 f^3 \lambda/\pi \\ y' &= 2a_6 f^3 \lambda/\pi \\ z' &= 2a_4 f^2 \lambda/\pi \end{aligned} \tag{10}$$

Where f and λ are the focal length and the wavelength respectively. If $aber = 0$ only a plane is subtracted from the map ($P = P(a_1, a_2, a_3)$). If $aber = 1$, also defocusing and feed displacement effects are considered ($P = P(a_1, a_2, ..., a_6)$), while if $aber = 2$ all first- and second-order aberrations are considered: beam steering + defocusing + feed displacement + astigmatism ($P = P(a_1, a_2, ..., a_7)$). Note that this analysis should be considered as alternative to the (lower-order) Zernike polynomial analysis.

6.8 zern

Purpose: Zernike polynomial analysis

- *input files*: errmap.dat
- *output files*: zern.dat
- *plots*: error map corrected by aberrations, Zernike polynomial map (aberration map), Zernike polynomial fitting weights

zern creates the zern.dat file containing the error map grid corrected by the fitted polynomials selected by the *zp* parameter. *zp* is an integer array containing the Zernike polynomials indexes (table 3). The fitting result (aberration map) is subtracted by the error map and saved in the zern.dat file. Optics and receiver misalignments introduce spurious aberrations on the holographic map. If you know your system and such effects are well characterized, you may decide to expunge such contributions.
More in general, *zern* allows you to characterize the antenna surface in terms of small scale and large scale deformations. The fitting weights give you information about each contribution in a quantitative way.
You have to run *zern* even if the Zernike analysis is not required, as it creates the *lut* input file. In such a case set $zp = 0$.

zp	Aberration	Z(n,m)	
1	piston	Z(0,0)	
2	horizontal tilt	Z(1,-1)	
3	vertical tilt	Z(1,1)	
4	oblique primary astigmatism	Z(2,-2)	
5	defocus	Z(2,0)	
6	vertical primary astigmatism	Z(2,2)	
7	oblique trefoil	Z(3,-3)	
8	horizontal coma	Z(3,-1)	
9	vertical coma	Z(3,1)	
10	vertical trefoil	Z(3,3)	
11	oblique tetrafoil	Z(4,-4)	
12	oblique secondary astigmatism	Z(4,-2)	
13	primary spherical	Z(4,0)	
14	vertical secondary astigmatism	Z(4,2)	
15	vertical tetrafoil	Z(4,4)	

Table 3: The first 15 Zernike polynomials.

6.9 lut

Purpose: Primary reflector look-up table

- *input files*: zern.dat
- *output files*: lut.txt
- *plots*:

The *lut* output is lut.txt, the look-up table file. The LUT is obtained by fitting each panel (figure 3) by means of the bivariate form $P = a_1 i^2 + a_2 i + a_3 ij + a_4 j^2 + a_5 j$. Where (i,j) are the map pixel coordinates and a_k are the fitting parameters. The error map pixels belonging to a specific panel are obtained by masking the error map grid with a discretized panel representation (figure 4). As each actuator shares four (in some case two) panels, the LUT is obtained by averaging the four (two) polynomials evaluated at the actuator position.

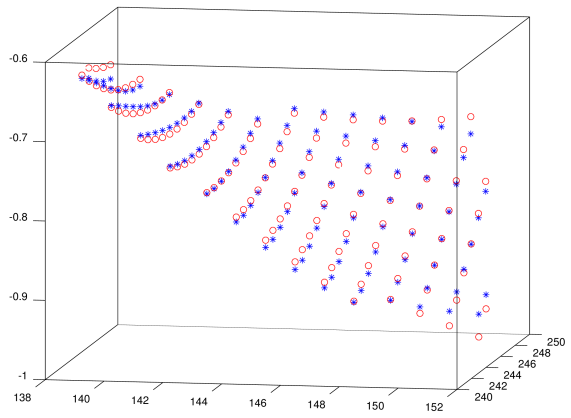

Figure 3: Panel deflections as measured by holography (red circles) and bivariate fitting (blue stars). Units are mm for the vertical axis, pixels for the horizontal ones.

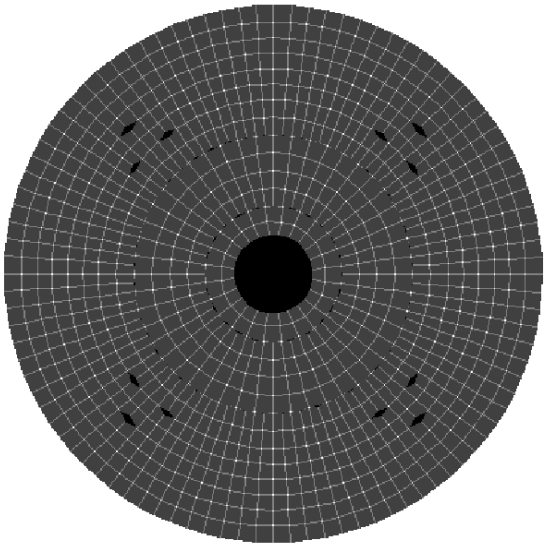

Figure 4: SRT panel grid mask.

Each panel is represented (on average) by about 150 pixels for $mp = 512$. The lut.txt file is a 1116 record file. Each record contains the panel code, the polynomial values at the actuator position, the average of the polynomial values, the greatest deviation from the mean value and, finally, the error map value at the actuator position.

6.10 HoP data files

The table below gives information about the HoP data files.

File name	Created by	Used by	Content
raw-data file	MHS	*ql*, *preproc*	see chapter 4
azel.txt	*preproc*	*uv*	*Re* and *Im* parts of holographic signal and on-source corrections (array)
uuvv.dat	*uv*	*phase*	*Re* and *Im* parts of the radiation diagram in *uv* co-ordinates (grid)
phase.dat	*phase*	*unwr*, *unwr*1	mask (array), amplitude (grid) and phase (grid)
unwr.dat	*unwr*, *unwr*1	*errmap*	mask (array), unwrapped phase (grid)
errmap.dat	*errmap*	*zern*	mask (array), error map (grid), clipped error map (see *tol*3)
zern.dat	*zern*	*lut*	Zernike coeff. (array), error map (grid), clipped error map (see *tol*4), Zernike surface (grid)
lut.txt	*lut*		look-up table file

Table 4: Output file list.

7 HoP by example

In this section we will process a holographic data-set from scratch. Data has been acquired at the SRT, in February 2017. The tracked satellite was Eutelsat 7A (11.54 GHz, elevation $\sim 44°$).
As first step we create a new folder (*hop_example*) into our projects folder. Then we copy the raw-data file (srt_holo.txt) into the folder. Now we can create an empty file that will be our parameter file. We may decide to call it conf.m which is the default parameter file name or we can choose a different name, let's say param.m. In such a case the parameter file name must be passed to the functions as argument.

```
octave:18> pwd
ans = /home/franco/hop/projects/hop_example
octave:19> ls
param.m srt_holo.txt
```

With our favorite text editor we open the param.m file and we insert the following line.

```
filename='./srt_holo.txt';
```

Note that we didn't write the full path, this means that we assume that *hop_example* is the current working directory for Octave.
What we did is all we need to run *ql*.

```
octave:20> ql('param')
```

ql doesn't create any file. We run it to take a preliminary look to the Az/El map. Figure 5 shows the *ql* plots, while figure 6 displays zoomed parts of these plots.

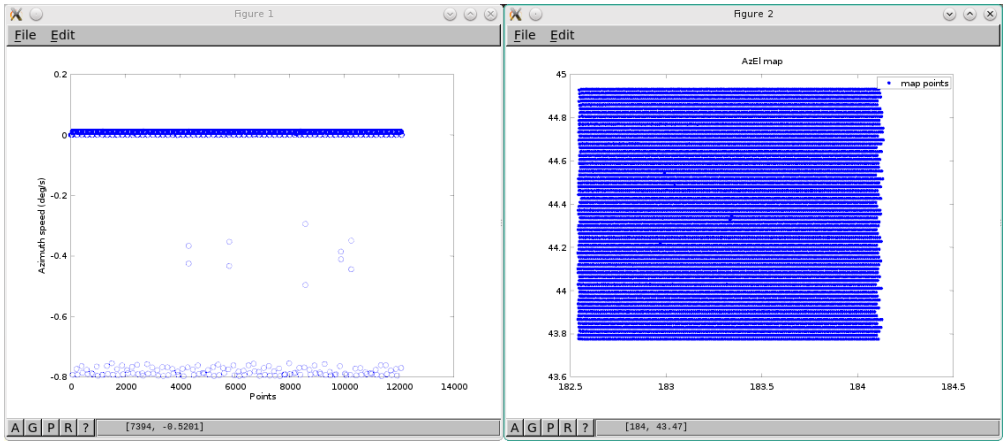

Figure 5: *ql* plots: azimuth speed (left) and Az/El map (right).

The next call (*preproc*) will extract on-source and map data from the raw-data file. Considering the left panel of figure 6, the azimuth speed range $(0.005 \div 0.015)$ corresponds to the scan points. The small arch of the right panel of figure 6 corresponds to the on-source data that will be used to correct the satellite drifting effects. Note that *ql* also displays the elevation speed, we don't consider this plot as we are dealing with an azimuth scan.

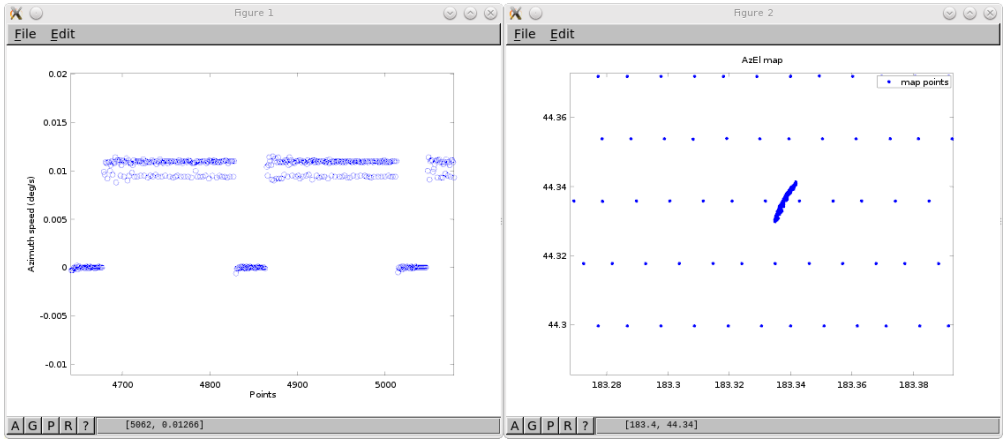

Figure 6: *ql* plots: zoomed azimuth speed (left) and Az/El map (right).

We can now add a few lines to param.m needed by *preproc*.

```
azcut=[183.3 2 0.05];
elcut=[44.34 2 0.05];
dam=[0.005 0.015];
```

```
scn='az';
source=true;
nma=50;
lambda=0.027;
```

dam is used to define the azimuth velocity range, while *azcut* and *elcut* define the source central coordinates, the map window and the source window. For the map window we chosen $\pm 2^\circ$ including all data points, but you can decide to vary such parameter, for example if you want to crop the map in a different way. The source window just defines the on-source point region; all data outside this window ($\pm 0.05^\circ$) are considered outliers and expunged from the map. *nma* is the number of samples (smoothing factor) considered for the moving average of source data. $source = true$ means that the phase will be corrected by the moving averaged satellite drifting effects. Finally we can run *preproc*.

```
octave:21> preproc('param')
*******************************
Scan points=9679
On source points=2354
az0=183.300000, el0=44.340000 (deg)
RMS_phase=0.892 (deg)
phase_err=22.3 (um)
RMS_ampl=0.0015
SNR=658.3
*******************************
```

Figure 7 shows the four plots we obtained by calling *preproc*. Note that the map points are now labeled (map points, satellite points and outliers). You should check those points to be sure that you parameter choice was proper. *preproc* prints the phase RMS that gives you an idea about the noise injected by the source, while the phase error is an estimation of the resulting uncertainty contribution in the error map. The bottom panels of figure 7 show the source amplitude and phase variations. As told before, the uv map will be corrected by such effects as we set $source = true$ in param.n.

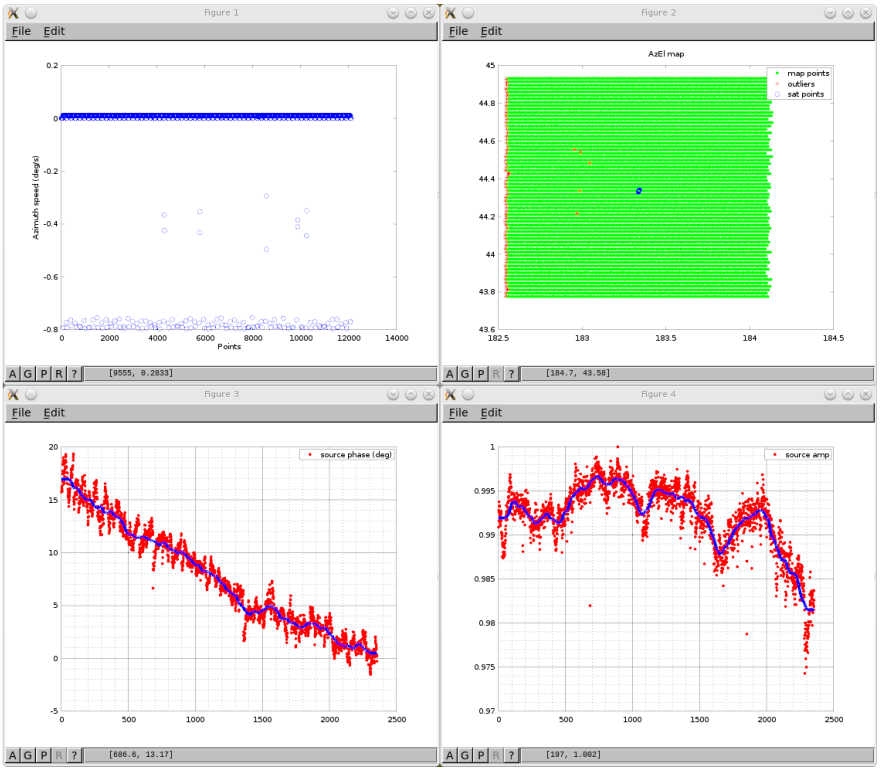

Figure 7: *preproc* plots, azimuth speed (top-left), Az/El map (top-right), phase and amplitude corrections (bottom). The blue lines in bottom panels are the satellite phase and amplitude smoothed by the moving average.

The next call, *uv*, needs a few more lines in param.m.

```
D=64;
k0=0.8;
mp=512;
```

$k0$ is the sampling factor introduced in equation 7. It should be chosen as a trade-off between (u, v) and (x, y) spaces optimal resolutions. Now, we can run *uv*.

```
octave:23> uv('param')
```

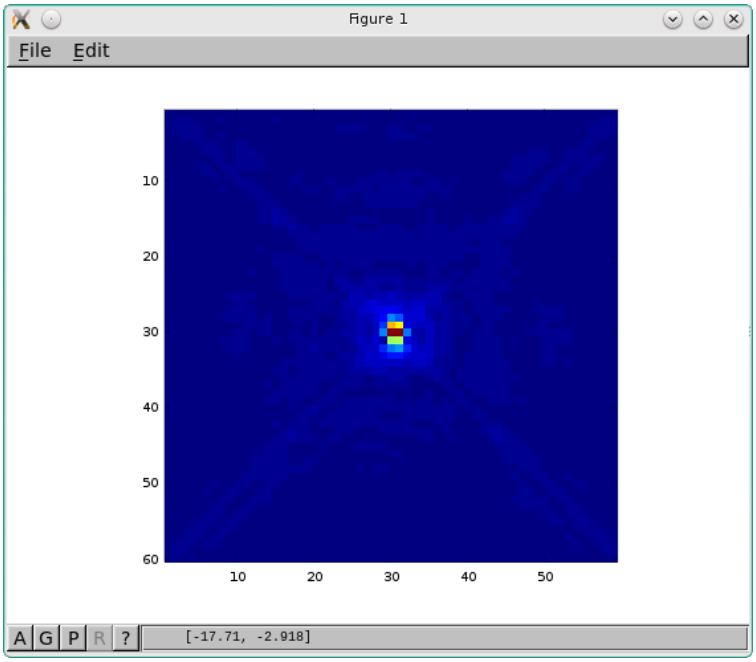

Figure 8: *uv* plot: radiation pattern (pixels).

Figure 8 shows the output plot of *uv*: the ** uniformly spaced radiation pattern in the (u,v) plane. If we are interested on a "pictorial" image of the radiation pattern, we can modify $k0$ in such a way:

```
k0=0.2;
```

Then we run *uv* again.

```
octave:24> uv('param')
```

The radiation pattern is saved in uuvv.dat file, **??? lets' load and manipulate it from the Octave command line**:

```
octave:30> close all
octave:31> ls
azel.txt param.m srt_holo.txt uuvv.dat
octave:32> load uuvv.dat
octave:33> whos
Variables in the current scope:

   Attr Name        Size                     Bytes  Class
   ==== ====        ====                     =====  =====
        ei       512x512                2097152  double
        er       512x512                2097152  double
octave:34> a=abs(er+i*ei);
octave:35> a=a/max(max(a));
octave:36> a=20*log10(a);
octave:37> surf(a)
octave:38> view (80,50)
```

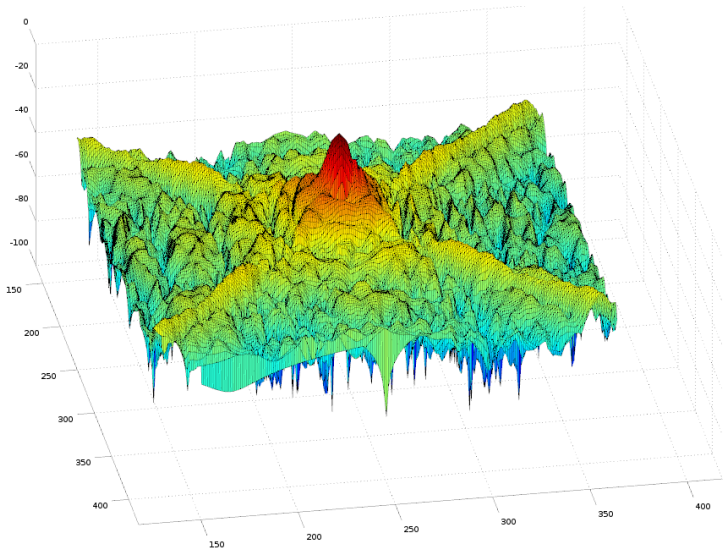

Figure 9: Radiation pattern obtained from uuvv.dat.

Figure 9 shows what should appear on the screen. If you are interested, it might be a good idea to slice the far-field radiation pattern and plot it in order to study the SNR.
Before proceeding, we need to refresh our parameter file:

```
k0=0.8;
```

Then we have to run "uv" again:

```
octave:39> clear all
octave:40> uv('param')
```

phase needs many parameters. We will choose carefully the best value for those parameter related to the masking of the antenna blocking parts. A good masking is mandatory to properly fix the phase ambiguities (*unwr*). $tol2$ is the amplitude threshold. All map points whose amplitude is lower than $tol2$ are masked (i.e. not considered in the calculations). Furthermore, you could decide to mask some feature near the sub-reflector; $r0\_tol(1)$ defines a circular area (in pixels), over $r0$, where a specific threshold mask ($r0\_tol(2)$) is applied.
As thresholds affect the phase unwrapping, it could be necessary to try many different thresholds (re-editing param.m) to achieve an optimal parametrization.

```
F=21.0236;
sgm=34;
pr=0;
pc=0;
trs=15;
r0=29;
r1=205;
r0_tol=[8 0.004];
tol2=0.0004;
```

We are ready to run *phase*:

```
octave:46> phase('param')
*******************************
dx=0.156556 (m/px)
r0=4.540117 (m)
```

```
r1=32.093933 (m)
dfx=-2.818004, -18 (m, px)
dfy=2.974560, 19 (m, px)
df=4.097457 (m)
*******************************
```

The top panels of figure 10 show the results of the Gaussian fitting that determines the map center. Only the amplitude values above *trs* [%] are considered for the fitting.
The bottom panels display amplitude and wrapped phase maps.

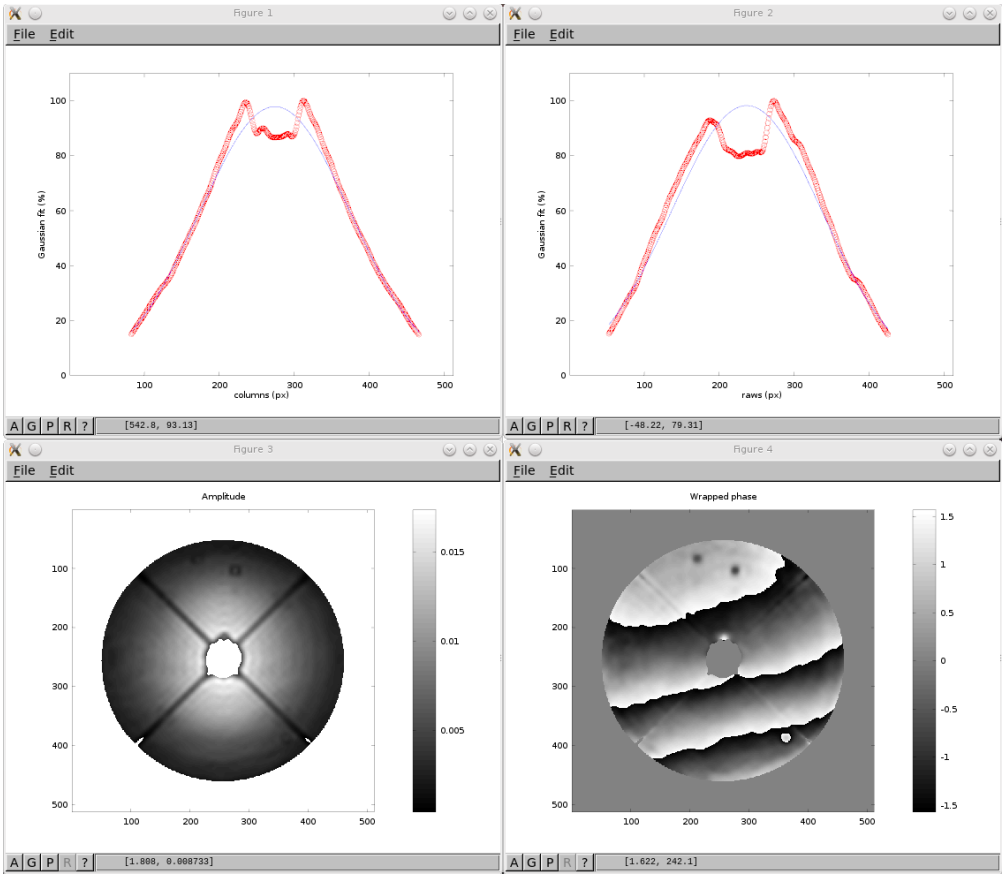

Figure 10: *phase* plots: the map center is determined by a fitting procedure (top), amplitude map (left-bottom) and wrapped phase map (right-bottom).

The next step concerns the phase unwrapping. We insert four more parameters and run *unwr*.

```
i1=400;
j1=250;
n2=150000;
tol=0.8;
```

```
octave:73> tic;unwr('param');toc;
128620/150000 iterations
Elapsed time is 22.6094 seconds.
```

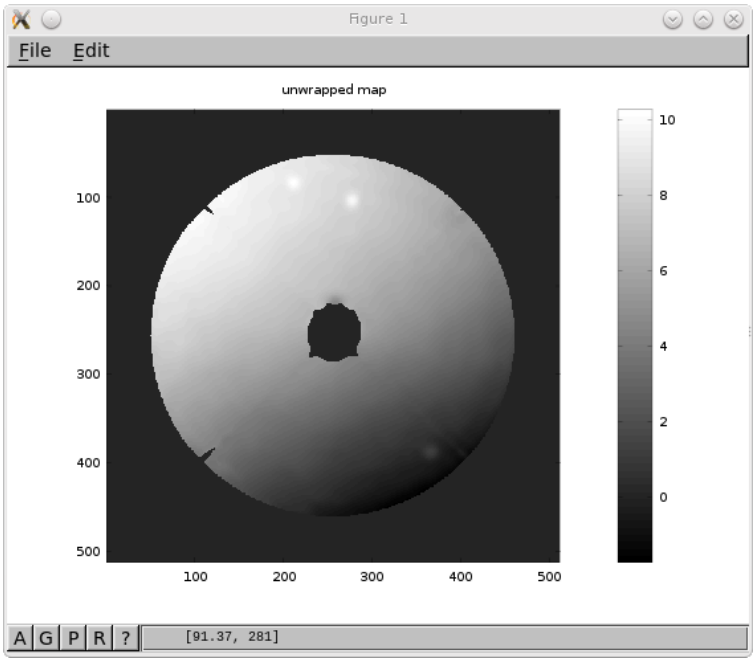

Figure 11: *unwr* plot: unwrapped phase map.

As the unwrapping algorithm is relatively slow, we use tic/toc to evaluate the execution time. Figure 11 shows the unwrapping result. ** If a smooth and regular map appears**, you should be confident that the algorithm worked well. But if phase discontinuities are still evident you should consider to modify the masking and/or to choose a new entry for $i1$ and $j1$ (algorithm starting point).
The next parameters are needed by *errmap*: $tol3$ defines the color map limits, $sfactor$ is the scale factor, *cmap* selects the color map to be used and *aber* sets the aberrations to be subtracted.

```
tol3=3;
sfactor=-1;
cmap='jet';
panels=0;
aber=0;
```

```
octave:75> errmap('param')
```

*** dopo aver lanciato errormap il codice restituisce il seguente errore: 'errormap' undefined near line 1 column 1 **** *** il comando è errmap e non errormap' . By running *errmap* we finally obtained the antenna surface profile (figure 12).

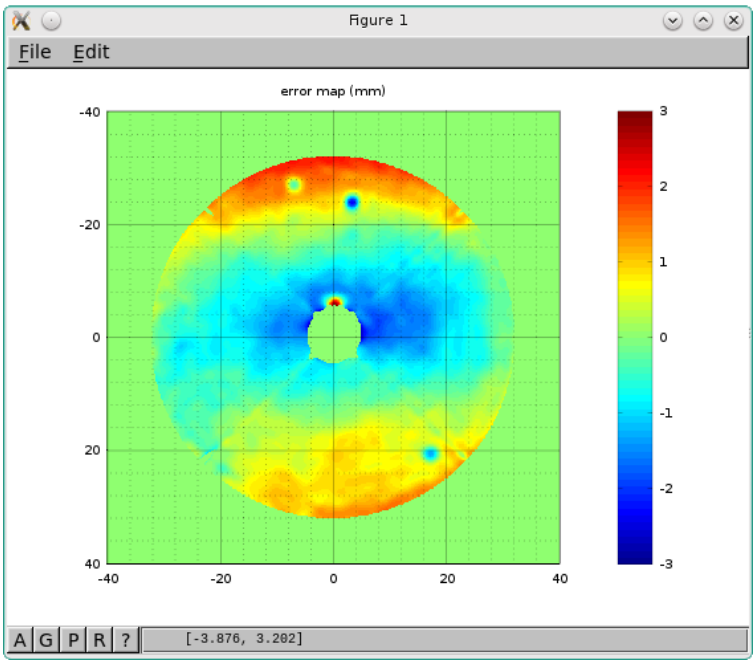

Figure 12: *errmap* plot: error map.

If you are familiar with deformations of large antennas probably you recognized the astigmatism (Z(2,2)) in the map of figure 12. By adding a few parameters we can characterize the surface aberrations in terms of Zernike polynomials. We may consider the first six terms including astigmatism in our analysis.
$panels = 0$ means that the actuator geometry will not be displayed in the Zernike map, $tol4$ is the color map limit and *datapath* is the path to the panels' geometry folder.

```
tol4=3;
zp=[1 2 3 4 5 6];
datapath='/home/franco/hop/dev/data/';
```

```
octave:77> zern('param')
```

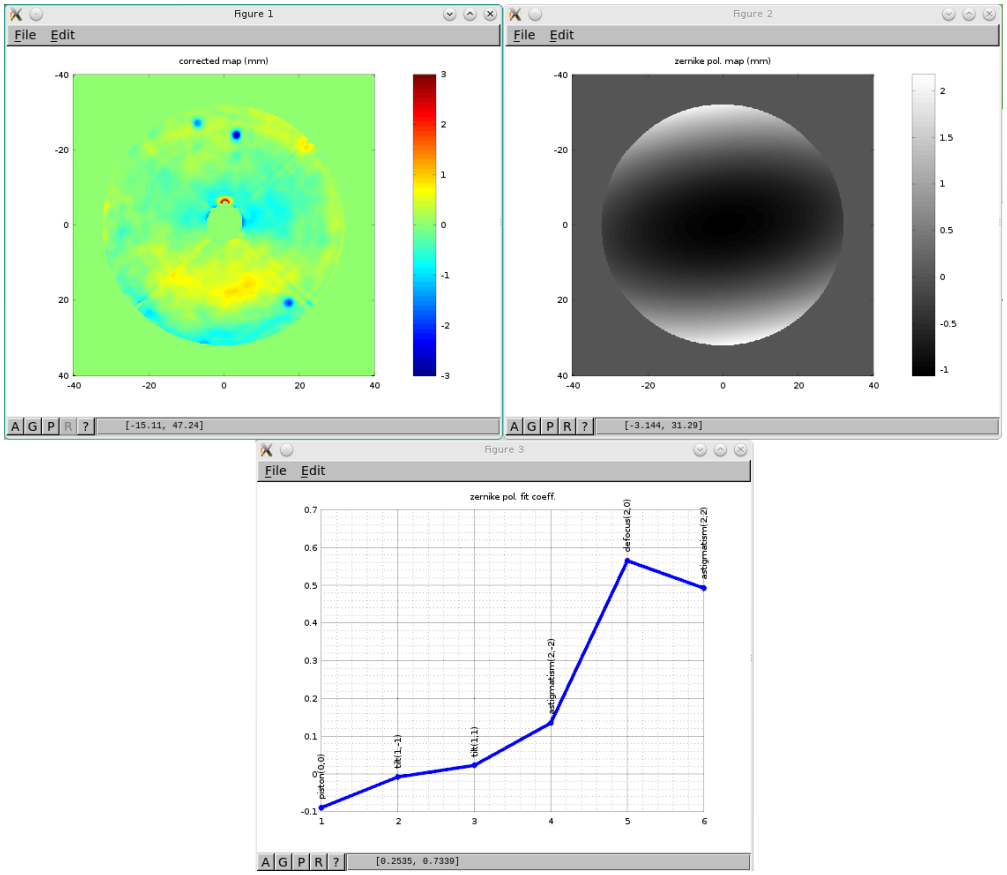

Figure 13: *zern* plots: error map deprived of large-scale deformations (top-left), Zernike surface (top-right), fitting parameters (bottom).

Figure 13 shows the *zern* outputs. The top-left is the error map of figure 12 without the large-scale deformation contribution displayed in the top-right panel. Note that coma is still present, maybe you would extend the analysis by considering more polynomials (e.g. Z(3,1), see table 3). The third panel represents the fitting parameters of each polynomial contribution. Defocus and astigmatism seem to be the most important aberrations affecting our antenna. As SRT is equipped with an AS system we want to correct all the surface error contributions. Thus we run again *zern* removing only tilt and piston from the error map.

```
zp=[1 2 3];
```

```
octave:78> zern('param')
```

Once we added the last parameters we can run *lut*.

```
npan=1008;
nact=1116;
```

```
octave:79> lut('param')
```

Finally *lut* creates the lut.txt file, winch contains deflections or corrections depending on the sign of $sfactor$:

```
SRT-03-09 T09H03 -1.782 0.000  0.000 -1.759 -1.771  0.012 -1.764
SRT-03-10 T07H03 -1.838 -1.828 -1.810 -1.867 -1.836  0.031 -1.826
SRT-03-11 T05H03 -1.819 0.000  0.000 -1.826 -1.822  0.003 -1.819
```

```
SRT-03-12 T03H03 -1.804 -1.773 -1.797 -1.807 -1.795  0.022 -1.787
SRT-03-13 T01H03 -1.668 0.000  0.000 -1.646 -1.657  0.011 -1.661
SRT-03-14 T95H03 -1.392 -1.396 -1.349 -1.369 -1.377  0.027 -1.393
SRT-03-15 T93H03 -1.151 0.000  0.000 -1.181 -1.166  0.015 -1.179
SRT-03-16 T91H03 -1.071 -1.095 -1.044 -1.030 -1.060  0.035 -1.066
SRT-03-17 T89H03 -0.899 0.000  0.000 -0.865 -0.882  0.017 -0.879
SRT-03-18 T87H03 -0.442 -0.406 -0.350 -0.456 -0.414  0.064 -0.447
SRT-03-19 T85H03 -0.462 0.000  0.000 -0.489 -0.475  0.013 -0.464
```

The first two columns are the actuator codes, the columns from 3th to 6th are the fitted panel surfaces evaluated at the position of each actuator. The 7th column is the average of the fitted values, the 8th column is the greatest deviation from the mean value. The last column is the error map evaluated at the actuator position (to be compared with the 7th column). All quantities are expressed in mm.
It is worth noting that astigmatism (but also defocus and coma) may also be due to holographic receiver feed displacements and/or feed phase center problems [9]. In the holography maps such effects may be superimposed to the "real" large-scale reflector deflections due to gravitational sag. All those effects related to the feed displacements must be minimized, before to perform the holographic run, by reaching the optimal feed focusing.

References

[1] Bolli P., Orlati A., Stringhetti L., et al., *Sardinia Radio Telescope: General Description, Technical Commissioning and First Light*, Journal of Astronomical Instrumentation, Vol. 4, Nos.3, 4, 2015.

[2] Prandoni I., Murgia M., Tarchi A., et al., *The Sardinia Radio Telescope: From a Technological Project to a Radio Observatory*, Astronomy and Astrophysics, accepted in March 2017, an internal report is in preparation.

[3] Rahmat-Samii Y., *Surface diagnosis of large reflector antennas using microwave holographic metrology: An iterative approach*, Radio Science, **19**, 1205–1217, 1984.

[4] Serra G., Bolli P., Busonera G., Pisanu T., Poppi S., Gaudiomonte F., Zacchiroli G., Roda J., Morsiani M. and Lpez-Prez J. A., *The microwave holography system for the Sardinia Radio Telescope*, Proc. SPIE 8444, Ground-based and Airborne Telescopes IV, 2012.

[5] G. Serra, S. Poppi, P. Bolli, F.Gaudiomonte, F. Buffa, *SRT Holographic System: installation and validation of the upgraded version*, internal report in preparation.

[6] Rahmat-Samii Y., *Microwave holography of large reflector antennas –simulation algorithms*, IEEE Trans. Antennas Propagat., **33**, 1194–1203, 1985.

[7] Lopez-Perez J. A., et al. *Surface Accuracy Improvement of the Yebes 40 Meter Radiotelescope Using Microwave Holography*, IEEE Trans. Antennas Propagat., **62**, 2624–2633, 2014.

[8] Tarchi D. and Comoretto G., *Holographic measurement on Medicina radio telescope using artificial satellites at 11 GHz*, Astronom. Astrophys., **275**, 679–685, 1993.

[9] Cogdell J. R. and Davis J. H., *Astigmatism in reflector antennas*, IEEE Trans. Antennas Propagat., **21**, 565–567, 1973.