INTERNAL REPORT

HoP User Guide

F. Buffa[1], G. Serra[1] and S. Poppi[1]
[1]INAF - Osservatorio Astronomico di Cagliari

Report N. 71, Released: 10/01/2018
Reviewer: C. Migoni

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
FPGA	Field-Programmable Gate Array
HoP	Holography Package
LUT	Lookup-Table
M2	Sub-Reflector
MHS	Microwave Holography System
PR	Primary Reflector (aka M1)
RA	Reference Antenna
RMS	Root Mean Square
RT	Radio Telescope
SI	Sampling Interval
SNR	Signal to Noise Ratio
SR	Spatial Resolution
SRT	Sardinia Radio Telescope

HoP

Holography Package

HoP User Guide

Version 1.1

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

October 16, 2018

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

Contents

1 Introduction

HoP stands for Holography Package for the Sardinia Radio Telescope (SRT) [1, 2]. It is a software implementing the processing of a data set measured by 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 that type of antennas. Therefore, given the complex far-field pattern of a high gain antenna, one can compute the map of the surface deformations (hereafter called even holographic or error 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 made up 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 first at the Medicina radio telescope [4] and then at the SRT [5] to measure the telescope far-field pattern by an on-the-fly raster scan around a geosynchronous 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 the new positions along their stroke.
This document deals with describing the HoP framework 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 data acquisition criteria and the HoP input file is described in Section 3. The structure and the installation of the HoP pipeline is outlined in Section 4. Moreover, a detailed description of the configuration file parameters and the function list in the HoP pipeline is reported in Section 5 and 6 respectively. Finally, the step-by-step instructions to process a holographic data set from scratch are outlined in Section 7.

2 Holography in a nutshell

Let $T(u,v)$ be the far-field radiation pattern of a large reflector antenna, i.e. the 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]The panel size ranges from 2.4 to 5.3 m^2.

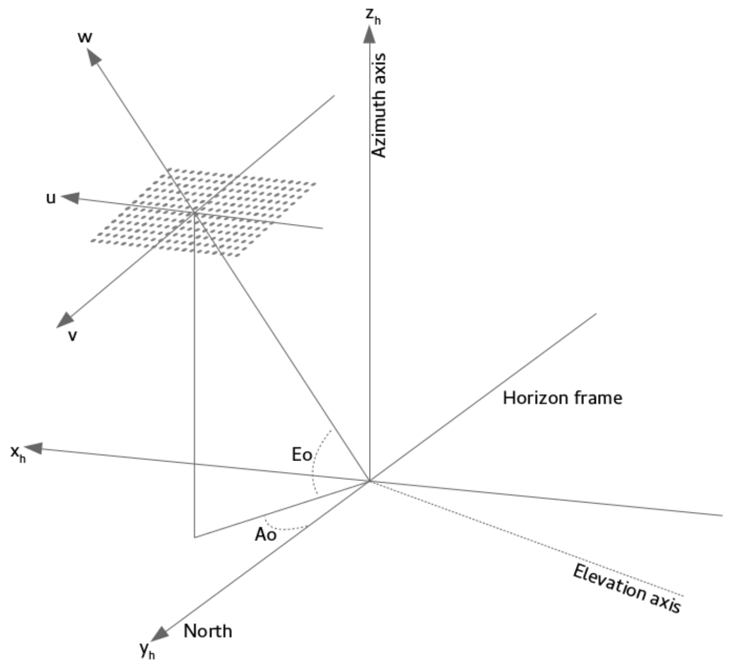


Figure 1: Horizon and (u,v,w) frames for an A/E mount.

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 horizon 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, 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 accuracy in recovering the antenna aperture plane. First of all, in order to avoid aliasing effects [6], a suitable sampling interval (SI) must be chosen to measure the far-field pattern. A general definition of SI is:

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

where D is the antenna diameter and α is an over-sampling parameter, which is empirically chosen between 0.5 and 1. Other holographic experiments [7] show that a suitable α to avoid the replica overlapping on the aperture plane domain (after the $\mathcal{F}^{-1}$ calculation) is ≤ 0.83. For the SRT holographic campaigns we have chosen $\alpha = 0.77$ for which we obtain $SI \sim 0.018^\circ$ with $\lambda = 0.026$ m and $D = 64$ m.
Then, given the desired spatial resolution (SR) on the aperture plane, the number N of sampling points of a generic radiation pattern map ($N \times N$ wide) is:

$$N = \frac{D}{\alpha SR} + 1 = \frac{M}{\alpha} + 1 \tag{5}$$

where M is the number of points in a generic aperture plane map ($M \times M$ large). Finally, the accuracy ($\delta\varepsilon$)

in recovering the pixel position is given by [8]:

$$\delta\varepsilon = \frac{\lambda M}{4\pi SNR} \tag{6}$$

where SNR is the signal-to-noise ratio of the measured far-field pattern. Therefore, if our goal is, for instance, $SR \leq 1.3\ m$, i.e. $M \geq 49$ for the SRT 64 m-diameter, we have to measure the far-field map with $N \geq 65$ sampling points (azimuthal or elevation scans) and a SNR ≥ 1000 (60 dB) is required in order to get a $\delta\varepsilon \leq 100\mu m$.
Once a (u, v) map is available, a pre-processing of the data set is needed before calculating the Fourier transform in equation 1. First of all, an interpolation procedure is carried out to regularize the map, if the sampling points are not uniformly distributed in the (u, v) space. Then, a zero-padding at the edges of the regularized map is added to improve the graphical quality of the aperture plane field map. A generic interpolation grid is typically defined by the grid size (the number of pixels) and by interval among two interpolation points:

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

where κ_o is a suitable factor introduced to regularize the (u, v) map and it is generally chosen in the range $0.5 \leq \kappa_o \leq 1$. Therefore, one can define the resulting pixel resolution as:

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

For an SRT holographic data set, the interpolation parameters are generally set to $M_p = 512$ and $\kappa_o = 0.8$, leading, after the Fourier transformation, to a pixel resolution equal to $\Delta x = \Delta y = 0.156$ [m/px] with $D = 64\ m$. Of course, this interpolation procedure doesn't improve the holographic map spatial resolution SR, depending on N, but it improves the graphical representation of the error map.

3 From MHS to HoP input data

Since 2016 the SRT is equipped with a microwave holography system (MHS) to monitor the surface deformations on its primary reflector. MHS is basically a two-element interferometer composed of two Ku-band receivers connected to a digital cross-correlator (backend) based on the FPGA technology [4]. The data acquisition and the file recorded by the MHS backend during a SRT holographic campaign are here described.
A typical SRT holographic campaign consists in measuring a geosynchronous satellite signal in the frequency interval 10.95-12.1 GHz, by means of two commercial feed horns located on the focus of the telescope main reflector and on the focus of a small reflector antenna[5]. From the feed horn to the backend, a radio frequency chain is optimized to amplify, frequency down-convert and filter a small portion (about 5 MHz) of the incoming signal, which can be chosen in the frequency interval 11.48-11.57 GHz by tuning a local oscillator.
Once the measurement starts, the backend continuously performs a real-time cross-correlation[6] of the two incoming 5 MHz-signals centered at 30 MHz, until the antenna control software finishes the schedule for mapping the telescope far-field pattern. Currently, the schedule consists of an on-the-fly raster scan[7] around an available satellite[8], whose position can be easily calculated from the ephemerides[9]. Moreover, a ten-seconds on-source pointing is generally added to the schedule at the beginning and at the end of each scan. The on-source measurements allow to record the time-dependent amplitude and phase variations during the map acquisition and to remove them in the data post-processing in order to calibrate the map.

[5]This antenna placed on the SRT apex floor as phase reference for the far-field pattern measurement.

[6]The cross-correlation has been implemented in the time-domain and the integration time was empirically chosen equal to 0.2 s in order to find a trade-off between the desired SNR and the overhead needed to the data recording.

[7]The raster scan can be performed in azimuth or elevation direction.

[8]At the geographic coordinates of the SRT site (39°29'39" N, 9°14'42" E) the geosynchronous satellites are visible over the horizon only for elevation angles ≤ 44 deg, see `https://www.lyngsat.com/` or `https://https://www.satbeams.com/`.

[9]The satellite position can be calculated by downloading the Two Line Elements file available at `https://www.space-track.org`.

Finally, for each data record[10], the backend control software manages the dataflow and writes a new line in a text file which is closed when the acquisition stops. Each file line consists of 12 columns described here following:

Col 1: azimuthal 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 columns 1 and 2 are the azimuth and elevation coordinates of the telescope pointing, which first the antenna control software sends to the backend manager and then this latter records them beside each data record. The columns 3 and 4 are the total powers of the radio telescope (RT) and reference antenna (RA) signals respectively. The column 5 is the total power of the 90°- shifted RA signal. The columns 6 and 7 are the correlation between the RT and RA signals and the correlation between the RT and the 90°- shifted RA signals. The columns 8 and 12 are the percentage of the normalized cross-correlation module and the date/time respectively. The columns 9 to 10 are the power in decibel of the RT and RA signals at the backend input. Column 11 is the cross-correlation phase expressed in degrees. The columns from 9 to 11 are mainly used to monitor the map acquisition, but they are not used in the HoP pipeline. At the end of the holographic map schedule, this text file (hereafter called MHS output or raw data file) can be processed by the HoP pipeline, which first of all combines the columns 4, 5, 6, and 7 to obtain:

$$\begin{aligned} Real(T(A,E)) &= \frac{Col6}{\sqrt{Col4}} \\ Imag(T(A,E)) &= \frac{Col7}{\sqrt{Col5}} \end{aligned} \tag{9}$$

where $Real(T(A,E))$ and $Imag(T(A,E))$ are the real part and the imaginary part of the far-field pattern $T(A,E)$.

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 processes something and it saves the results in an output file (the input for the next function). No supplementary information are needed, apart from the configuration file shared by all the scripts which is described in Section 5. HoP performs basically three main tasks:

- pre-processing of the far-field pattern and calibration data;

[10]Currently a single data record takes to be written about 0.4 s, 0.2 s for the integration and about 0.2 for the overhead.

- aperture field calculation and *optical* aberration analysis;
- LUT estimation.

In the pre-processing, HoP extracts from the raw data file the antenna azimuth and elevation during the acquisition, the azimuth and elevation during the on-source calibration and calculates the phase and the amplitude of the far-field map by using the correlation products and equations 9. These data are used to create a uniformly spaced (u, v) grid, optionally corrected by the on-source signal variations.
In the aperture field calculation, HoP performs a 2D-inverse fast Fourier transformation of the far-field pattern and corrects it for the phase ambiguity by a 2D-unwrapping procedure. Then, the antenna surface aberrations are (if needed) analyzed in terms of Zernike polynomials.
Finally, the aperture field phase is modeled, panel by panel, by means of a bivariate polynomial fitting considering all the data points falling within the area of each panel. For each actuator, a correction (i.e. the LUT element) is calculated by averaging four/two polynomials calculated at the actuator position, as each actuator shares four/two panels.

4.1 Installing HoP

If you are familiar with the GNU Octave environment you know that Octave has been built to be 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 do it.
Installing HoP is easy. You can download the package from `https://www.ict.inaf.it/gitlab/franco.buffa/hop`. 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 containing 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 data set composed of 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 like an argument to the functions. If no argument is given, 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 raw data file.

5.1 Parameter description

A detailed definition of the HoP parameters is provided herein and summarized in table 1:

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

datapath: Path to the folder containing the panel/actuator topology files. The topology files represent the reflector panels as a gridded map. Therefore, those files are specific for the selected number of pixel in the map (see mp parameter). In this HoP release two folders are provided: $data$ for $mp = 512$ and $data256$ for $mp = 256$.

mp: The number of pixels in the map is $mp \times mp$.

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

$source$: If true, the map calibration is applied by using the smoothed on-source data [true/false].

nma: Step of the temporal moving average used for the on-source data smoothing. The phase and amplitude of the smoothed on-source data are used to calibrate the map if $source = true$.

$azcut$: Three-elements float array used to discriminate between map data and on-source data along the azimuthal coordinate [°]: satellite coordinate, half-width of the map and the half-width of the window including just the on-source data.

$elcut$: Three-elements float array used to discriminate between map data and on-source data along the elevation coordinate [°]: satellite coordinate, half-width of the map and the half-width of the window including just the on-source data.

dam: Float array containing the upper and lower limits of scan velocity [°/time]. The grid points are selected in terms of their azimuthal (or elevation, depending on scn) scan speed. This criterion allows to label the points 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. Set $sgm = 0$, if you don't want to use the window function.

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

$panels$: If $panels$=1, a dark dot depicts each actuator position on the deformation map, while if $panels$=2, the edges of the each panel are shown. If you don't want any graphic symbol overlapping the error map, set $panels$=0.

$lambda$: Wavelength [m].

D: Antenna diameter [m].

$k0$: Sampling factor introduced in equation 7.

F: Focal length [m].

$tol2$: Threshold used in the aperture map to outline the blockage areas due to sub-reflector and quadripode shadow and the map edge. Basically, all the pixel having an amplitude $\leq tol2$ will be masked. This parameter, required by the unwrapping algorithm, is related to $r0\_tol$.

$r0$: Radius of the sub-reflector [pixel]. This parameter allows to mask the sub-reflector. It is required by the unwrapping algorithm. For instance, given the SRT sub-reflector radius $R_{M2} = 4.5m$ and $\Delta x = 0.156$ [m/px] (see Section 2), $r0 = R_{M2}/\Delta x = 4.5/0.156 = 29$ pixels.

$r0\_tol$: Float array containing two parameters needed to properly set the sub-reflector mask, see figure 2. 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 the inner 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 diameter. It is required by the unwrapping algorithm. For instance, given the SRT main reflector radius $R_{M1} = 32m$ and $\Delta x = 0.156$ [m/px] (see Section 2), $r1 = R_{M1}/\Delta x = 32/0.156 = 205$ pixels.

Parameter	Type	Description	Used by
filename	string	raw data file name	preproc ql
datapath	string	SRT AS folder	lut zern
mp	int	number of pixel in the map	all functions
scn	string	select scan type	preproc
source	boolean	on-source calibration [true/false]	preproc
nma	int	on-source data smoothing	preproc
azcut	float array	required to separate on-source data	preproc uv
elcut	float array	required to separate on-source data	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	primary reflector diameter	errmap lut phase uv
k0	float	sampling factor (see equation 7)	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	primary reflector 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: List of the HoP setting parameters and the functions using them.

pr: y-coordinate (row) of the map center[11] [pixel]. If both *pr* and *pc* are set to zero, the map center is evaluated automatically.

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

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

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.

n2: Iteration limit for the recursive unwrapping algorithm.

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

cmap: Set the current colormap (the Octave function $colormap('list')$ returns a list with all 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 and subtracted, finally if $aber = 3$ only plane + astigmatism is removed.

tol3: Range limits of the color bar used by *errmap* for the map plot [mm].

tol4: Range limits of the color bar used by *zern* for the map plot [mm].

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 ($npan = 1008$ for the SRT AS).

nact: Number of actuators ($nact = 1116$ for the SRT AS).

5.2 Configuration file example

The configuration parameters in the file $conf.m$ are Octave variables (integer, float, array, etc) and may be inserted in whatever order. When a parameter is changed during the processing, you should restart your analysis, as such variables are shared by different functions.
An example of 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'; % 'gray';
sgm=50; %34;
trs=15;
panels=0;
%
lambda=0.026;
D=64;
k0=0.8;
```

```
F=21.0236;
tol2=0.0003;%0.0004;
r0=29; %28;
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 legibility.

6 Function list

Together with the configuration file, only the holographic data-set file is required to run HoP. The HoP functions are sequentially called until the LUT is created. Each function needs one (or more) input file and creates an output file required by the next elaboration step. The names of the intermediate files are hard-coded, so you cannot modify them.
The *lut* function needs further input files describing the geometry of the SRT primary mirror panels (see *datapath* parameter). A detailed descriptions of the HoP function is available in the next sub-sections and then summarized in table 2. Finally, for the sake of clarity, the HoP output files are listed in table 4.

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	phase unwrapping	wrapped phase	unwrapped phase
unwr1	phase unwrapping	wrapped phase	unwrapped phase
unwr2	fast phase unwrapping	wrapped phase	unwrapped phase
unwr3	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, A/E map

ql is used to check the raw data file. The raw data file contains three data types: the map points, the on-source calibration data and (possible) 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, A/E map, amplitude and phase of the on-source data useful to measure the satellite drift during the map acquistion.

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 on-source data are used to correct the map for the satellite drift effects (see *source* and *nma* parameters in table 1). As first step, *preproc* considers the antenna azimuth (or elevation) angular velocity (see *dam* and *scn* parameters in table 1) in order to extract the map data. Once the map data are labeled, the code extracts the on-source data. These latter are selected within the window defined by *azcut* and *elcut* parameters. 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 (u, v) 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 (x, y) 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.
Note that *phase* needs some more parameters ($tol2$ and $r0\_tol$) required to define the blockage areas (see figure 2). Furthermore, the *sgm* parameter defines the window function radius used as Fourier transform tapering function.

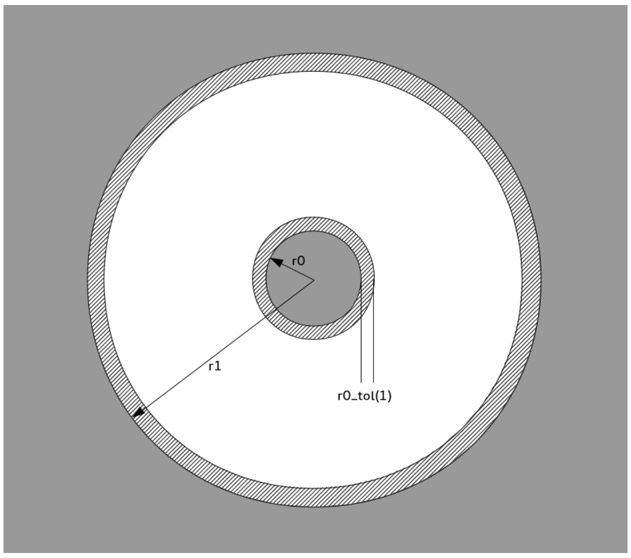


Figure 2: Only the circular region $r0 \leq r \leq r1$ is considered by HoP for calculations. The unwrapping algorithm requires the masking of the blockage areas by means of $tol2$. Close to the sub-reflector you can use $r0\_tol$ to refine the mask: $r0\_tol(1)$ defines the masking area, $r0\_tol(2)$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

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 parameters are the pixel coordinates of the starting point. The starting point must be carefully chosen near the map center (where the SNR is better than the outer parts of the map) and far from the blocking parts (sub-reflector, struts, etc.).

6.6 unwr1

Purpose: phase unwrapping

- *input files*: phase.dat
- *output files*: unwr.dat

- *plots*: unwrapped phase map

$unwr1$ implements an unwrapping algorithm slower than that implemented in $unwr$, but more reliable. This is an option one can choose when $unwr$ fails due, for instance, to a noisy data set. The $n2$ parameter limits the number of iterations of the recursive algorithm. The i1 and j1 parameters are the pixel coordinates of the starting point.

6.7 unwr2

Purpose: phase unwrapping

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

$unwr2$ is a driver for the well-established unwrapping algorithm proposed by Herraez et al. [9]. If you decide to use $unwr2$ you don't need to set $n2$, $i1$ and $j1$. We recommend you consider $unwr2$ as the first choice.

6.8 unwr3

Purpose: phase unwrapping

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

$unwr3$ implements the unwrapping algorithm proposed by Juarez-Salazar et al. [10]. If you decide to use $unwr3$ you don't need to set $n2$, $i1$ and $j1$.

6.9 errmap

Purpose: error map estimation

- *input files*: unwr.dat
- *output files*: errmap.dat
- *plots*: error map with or without systematic errors due to optics' aberrations

errmap creates the errmap.dat file containing the deformation map calculated by equation 1.
$sfactor$ allows you to set the map sign according to the deformation sign. Here we agree that $sfactor = -1$ produces a deformation map, while $sfactor = 1$ produces a correction map ready to fill the LUT. Note that such parameter may be also used as scale factor to weight the entire deformation map[12].
panel enables the actuator ($panel = 1$) or panel ($panel = 2$) representation on the deformations map. The *aber* parameter allows you to subtract first- and second-order aberration terms due to optics' misalignments (systematic errors). For this purpose, a polynomial fitting can be used to calculate the following P term [11, 12]:

$$P = a_1 + a_2x + a_3y + a_4\rho^2 + \rho^2(a_5x + a_6y) + a_7\sigma^2 \tag{10}$$

which can be subtracted to the phase distribution. Here a_k are the fitting parameters, $\rho = \sqrt{x^2 + y^2}$ and $\sigma = \sqrt{x^2 - y^2}$. In addition, the fitting results provide the feed displacement coordinates:

$$\begin{aligned} x' &= 2a_5f^3\lambda/\pi \\ y' &= 2a_6f^3\lambda/\pi \\ z' &= 2a_4f^2\lambda/\pi \end{aligned} \tag{11}$$

[12]Before measuring a beam pattern map, a panel can be intentionally moved of a known amount upward or downward to define the deformation sign and the scale factor.

If $aber = 0$, only a plane ($P = P(a_1, a_2, a_3)$) is subtracted from the map (beam steering removing). If $aber = 1$, defocusing and feed displacement effects ($P = P(a_1, a_2, ..., a_6)$) are removed as well. While if $aber = 2$, all the terms in equation 10 are removed ($P = P(a_1, a_2, ..., a_7)$) including the astigmatism effects. Finally, if $aber = 3$, only beam steering and astigmatism are removed ($P = P(a_1, a_2, a_3, a_7)$). Note that this analysis should be considered as alternative to the (lower-order) Zernike polynomial analysis.

6.10 zern

Purpose: Zernike polynomial analysis

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

zern plots the error map corrected for the large scale aberrations due to the optics' misalignment by a Zernike polynomial fitting. *zp* is an integer array containing the Zernike polynomials indexes, see table 3. The fitting result produces a map of the large scale aberrations (removed from the errmap.dat file) and saves it in the zern.dat file. 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 reflector surface in terms of small scale and large scale errors. The weights of the Zernike fitting provide you with information about the aberrations affecting the reflector surface.
You have to run *zern* even if the Zernike analysis is not required (i.e. $zp = 0$), as it creates the input file required from *lut*.

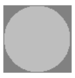
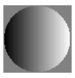
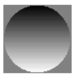
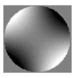
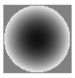
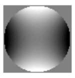
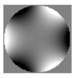
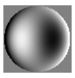
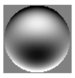
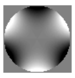
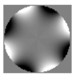
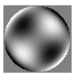
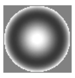
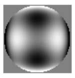
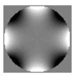
zp	Aberration	Z(n,m)	
1	piston	Z(0,0)	
2	horizontal tilt	Z(1,-1)	
3	vertical tilt	Z(1,1)	
4	oblique primary astigmatism	Z(2,-2)	
5	defocus	Z(2,0)	
6	vertical primary astigmatism	Z(2,2)	
7	oblique trefoil	Z(3,-3)	
8	horizontal coma	Z(3,-1)	
9	vertical coma	Z(3,1)	
10	vertical trefoil	Z(3,3)	
11	oblique 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.

6.11 lut

Purpose: Primary reflector look-up table

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

The *lut* output is a text file (lut.txt), i.e. the look-up table. Each LUT element is an offset to add to the actuator position in order to compensate for the measured deformation. This offset is the result of fitting the pixels of the panels closest to the corresponding actuator (figure 3) by means of a bivariate polynomial, having a form as $P = a_1 i^2 + a_2 i + a_3 ij + a_4 j^2 + a_5 j$. Where (i,j) are the pixel coordinates and a_k are the fitting parameters the function *lut* saves in the file pfit.dat. The 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 element is obtained by averaging the four (two) polynomials evaluated at the actuator position.

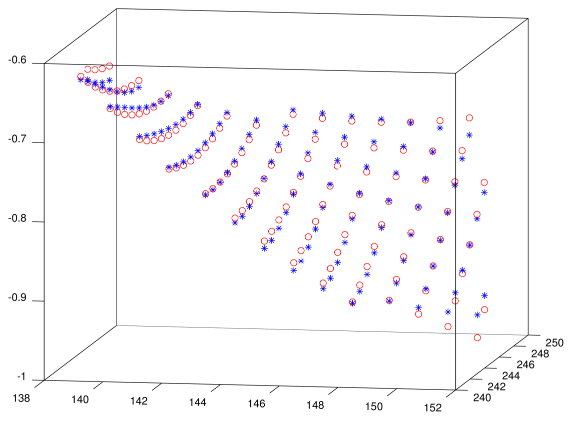


Figure 3: Panel deflection as measured by holography (red circles) and bivariate fitting (blue stars). Units are mm for the vertical axis, pixels for the horizontal ones.

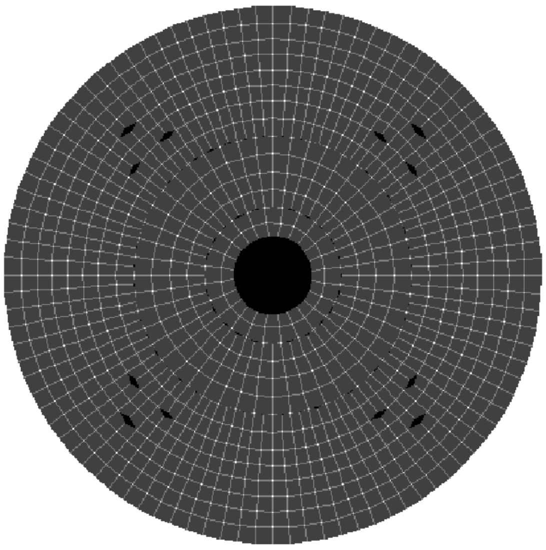


Figure 4: Representation of SRT panelled surface used to mask the holographic deformation map.

The number of pixels representing each panel varies with mp. For example, with $mp = 512$ the panels are represented (on average) by about 150 pixels each. The lut.txt file consists of 1116 records[13]. 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.

File name	Created by	Used by	Content
azel.txt	*preproc*	*uv*	Re and Im parts of the antenna radiation pattern and on-source data (array)
uuvv.dat	*uv*	*phase*	Re and Im parts of the antenna radiation pattern in uv coordinates (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)
pfit.dat	*lut*		bivariate polynomial fitting parameters
lut.txt	*lut*		look-up table file

Table 4: Output file list.

[13]One record for each SRT active surface actuator.

7 HoP by example

In this section we process a SRT holographic data-set from scratch. The data-set was measured in February 2017 by pointing Eutelsat 7A, a geosynchronous satellite at $\sim 44^\circ$ of elevation and receiving a portion of its signal at 11.54 GHz.
As first step, we create a new folder (*hop_example*) into */hop/projects* folder. Then, we copy the data-set file (srt_holo.txt) into the folder and create an empty file, the parameter file, that will be added to the same folder. This latter can be saved as *conf.m*, i.e. the default file name for the configuration parameter file or a different name can be used. We choose *param.m*, then 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, since we don't write the full path, we assume that *hop_example* is the current working directory for Octave. This is all we need to run *ql*.

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

ql does not create any file, but it allows one to have a look at A/E map and at the speed of the azimuth (or elevation) scan. Figure 5 shows the *ql* plots and a zoom in some parts of those plots is shown in figure 6.

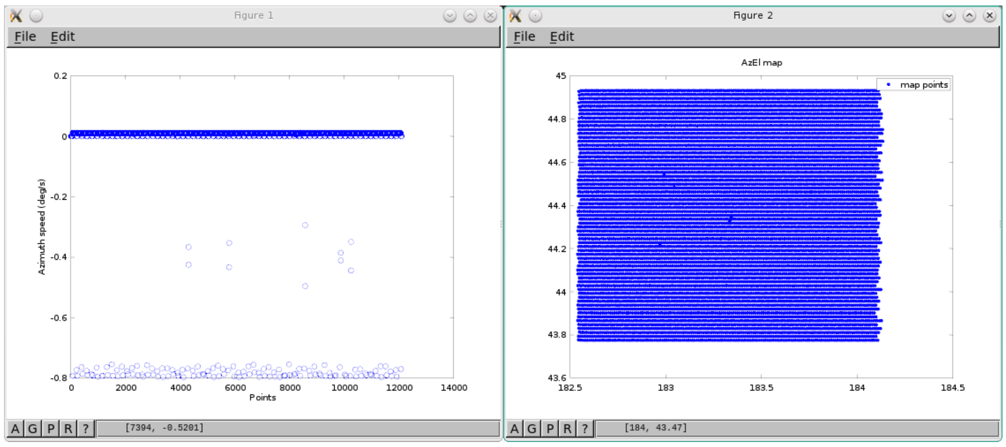


Figure 5: *ql* plots: azimuth scan speed (left) and A/E map (right).

The next call (*preproc*) extracts the on-source and map data from the data-set file. The azimuth scan points and the on-source data can be easily recognized respectively in in the left panel (speed range $0.005 \div 0.015$) and in the right panel of figure 6 (small arch in the center). Note that *ql* also displays the elevation speed, but we don't consider it as we are dealing with an azimuth scan.

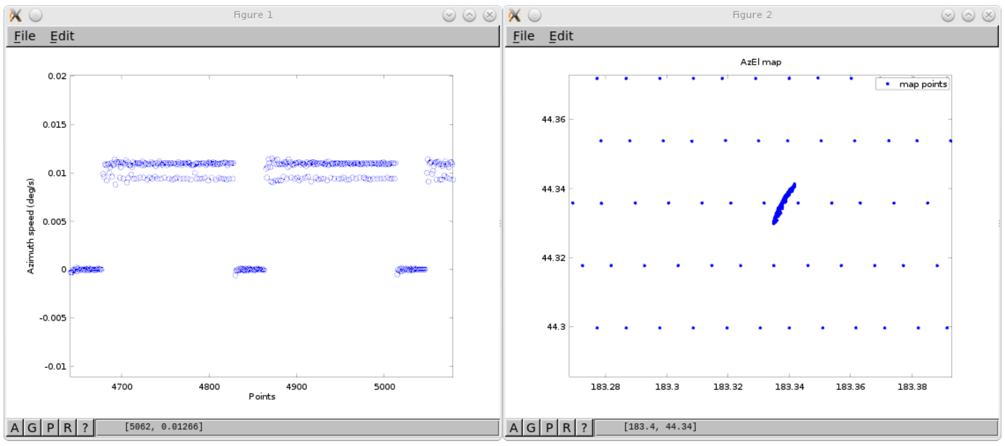


Figure 6: *ql* plots: zoom in azimuth scan speed (left) and A/E map (right).

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=0.026;
```

dam defines the azimuth speed range allowing to distinguish the on-source from the map data, the *azcut* and *elcut* ranges providing the source coordinates, the dimension of the windows needed to border the sky regions where the map and the on-source points are measured. We choose a map window dimension equal to $\pm 2^\circ$ including all data points, but one can decide to vary it in order to crop the map in a different way. All the data outside the on-source window (here $\pm 0.05^\circ$ wide) are considered outliers and expunged from the map. *nma* is the number of samples considered to calculate the moving average needed to smooth the on-source data. If $source = true$ the phase is corrected for the satellite drifting effects. Now, 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
********************************
```

After running *preproc* some useful information are written on the screen: the number of the map and on-source points inside the windows defined in *azcut* and *elcut*; the satellite coordinates; the RMS of the on-source data phase and amplitude which gives us an estimation of the phase measurement error and the resulting SNR. Then, *preproc* plots four panels all gathered in figure 7. The top panels show the azimuth scan speed (left) and the data points (right), where the map, the on-source (sat) points and outliers are labeled in different colored dots. One should check those points to be sure that the parameters (*dam*, *azcut*, *elcut* and *nma*) were properly chosen. The bottom panels show the on-source data phase (left) and amplitude (right) variations and the smoothed curve resulting from the moving average. The uv-map will be corrected by such effects if *source* = *true* in the parameter files.

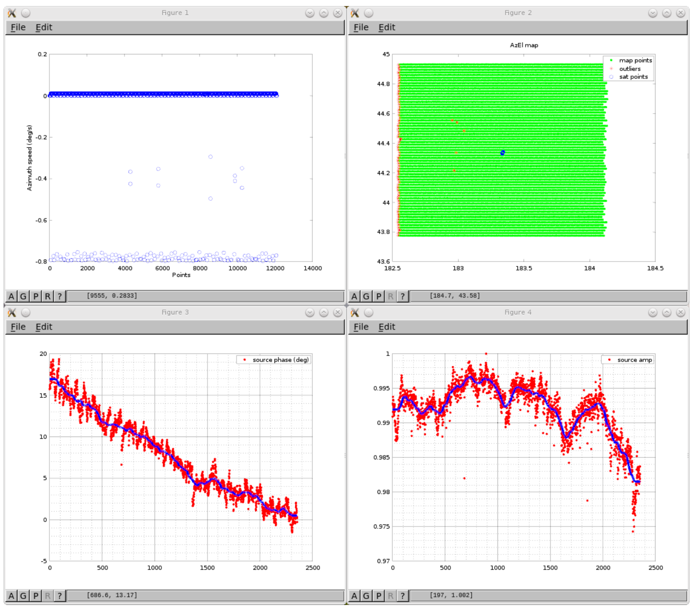


Figure 7: *preproc* plots: azimuth scan speed (top-left), A/E map (top-right), phase and amplitude on-source data fitting (bottom).

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 shows the *uv* output plot, i.e. the uniformly spaced radiation pattern in the (u,v) plane. If we are interested on an "artistic" 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')
```

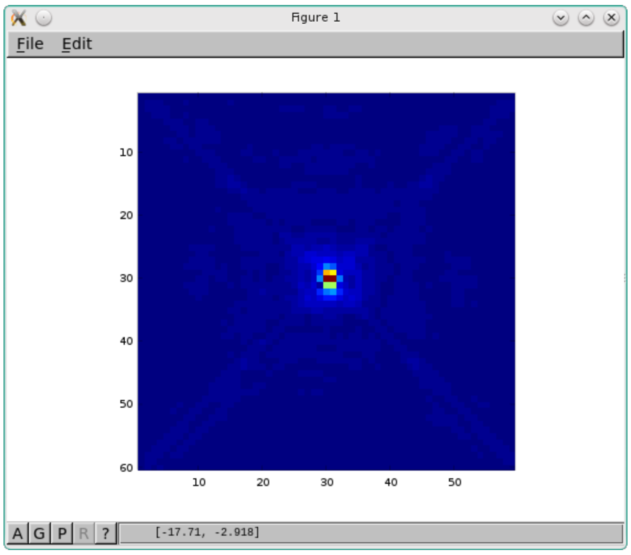


Figure 8: *uv* plot: radiation pattern (pixels).

Let's load and manipulate the radiation pattern in uuvv.dat file. 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)
```

A 3D-map of the far-field pattern is shown in figure 9 right after running the command surf.
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 whose values have to be carefully chosen to mask properly the blockage areas on the reflector surface. A good masking is mandatory to properly fix the phase ambiguities by the unwrapping algorithm (*unwr* or *unwr*1). *tol*2 is an amplitude threshold allowing to mask all the map points whose amplitude is lower than *tol*2 (i.e. not considered in the processing). Furthermore, you could decide to mask some unwanted features which could appear close to the sub-reflector blockage area by using $r0\_tol(1)$ (see figure 2).

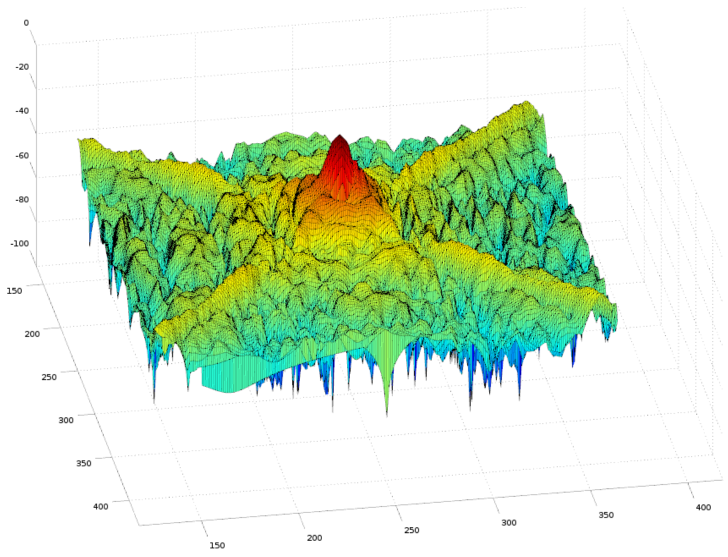


Figure 9: Radiation pattern obtained from uuvv.dat.

As those thresholds affect the phase unwrapping, it could be necessary to try different thresholds (by 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)
********************************
```

The *phase* printout shows some information about the amplitude and phase map on the aperture plane: dx is the map pixel size, dfx and dfy are the map center offset calculated by the Gaussian fitting (or set by means of *pr* and *pc* parameters). Moreover, *phase* plots four panels in figure 10. The top panels show the results of the Gaussian fitting along the map columns (left) and rows (right) defining the map center. Only the amplitude values above *trs* [%] are considered for the fitting. The bottom panels display amplitude (left) and the wrapped phase (right) maps.

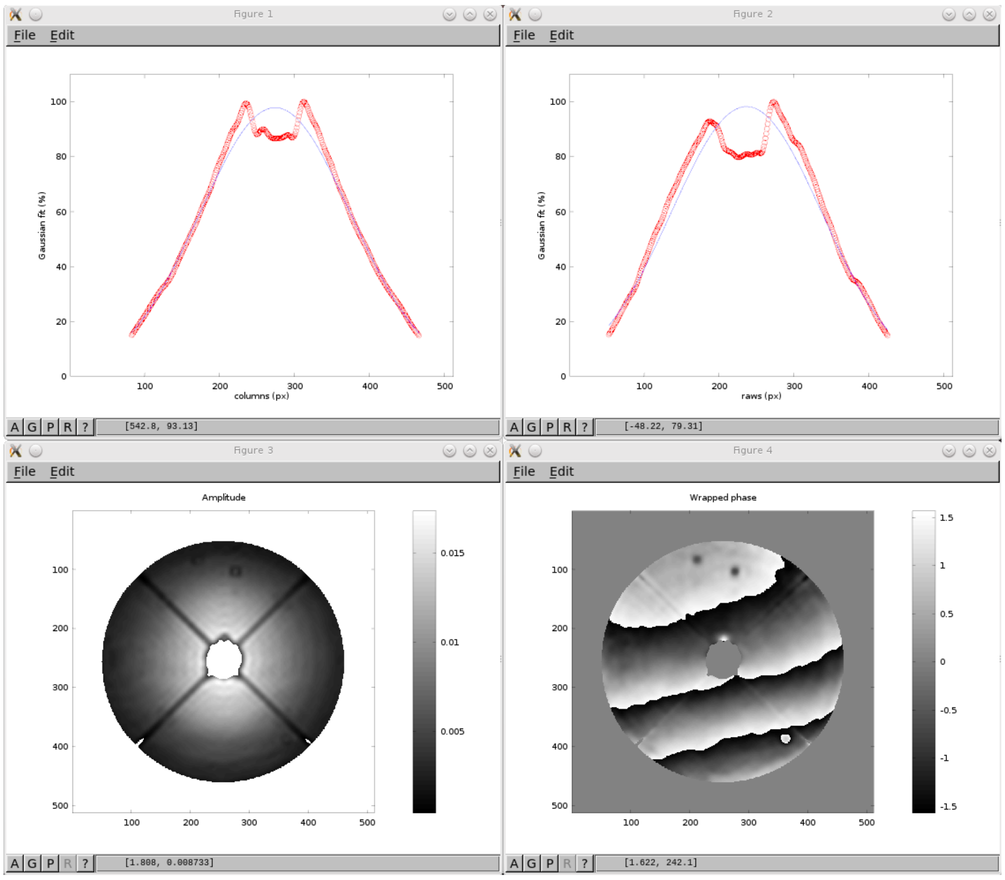


Figure 10: *phase* plots: Gaussian fitting along the map columns (left-bottom) and rows (right-top) to define the map center; amplitude map (left-bottom) and wrapped phase map (right-bottom).

The next step concerns with 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.
```

As the unwrapping algorithm is relatively slow, we use tic/toc to evaluate the execution time. Figure 11 shows the phase map after running *unwr*. If a smooth and regular map is shown, then the unwrapping algorithm worked well. Otherwise, if phase discontinuities are still evident, you should consider to modify the masking parameters and/or to choose a new entry for $i1$ and $j1$ (algorithm starting point).
The following parameters are needed to run *errmap* properly: $tol3$ defines the color map limits; $sfactor$ is the scale factor which can be inferred by a reference panel intentionally moved on the reflector surface before starting the holographic map acquisition; *cmap* is the color map type to be used; *aber* enables the optics' aberrations to be subtracted and *panels* enables or disables the panel or actuator graphic representation.

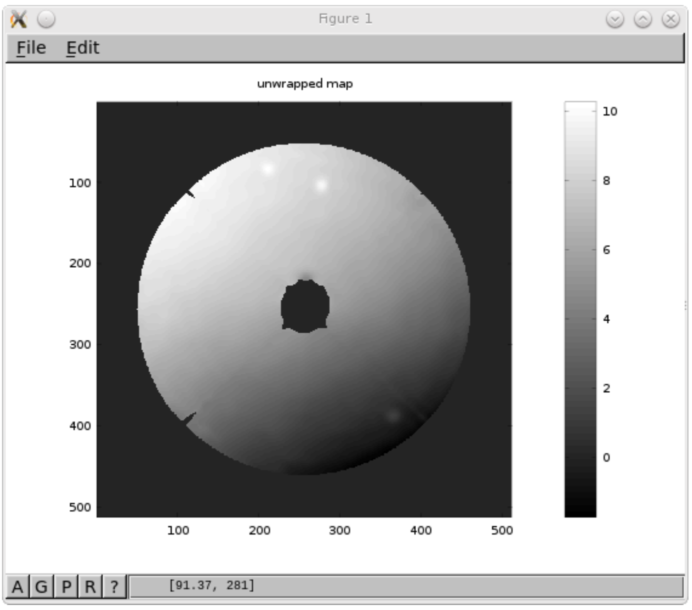


Figure 11: *unwr* plot: unwrapped phase map.

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

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

By running *errmap* we finally obtained the reflector surface deformations (see figure 12).

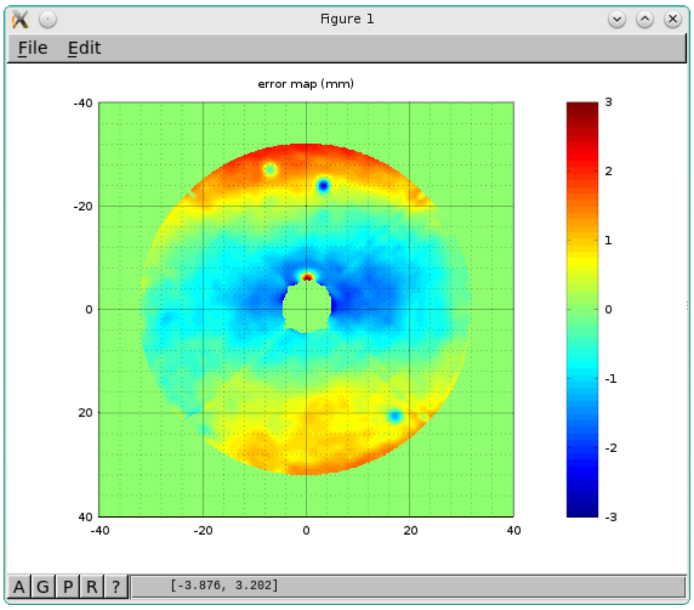


Figure 12: *errmap* plot: error map.

If you are familiar with deformations of large reflector antennas probably you recognize the astigmatism (Z(2,2)) in figure 12. By adding a few parameters we can characterize the surface aberrations in terms of Zernike polynomials. So, in our analysis we may consider the first six terms including astigmatism.
tol4 is the color map limit and *datapath* is the path to the panels' geometry folder. At this point *zern* can be run:

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

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

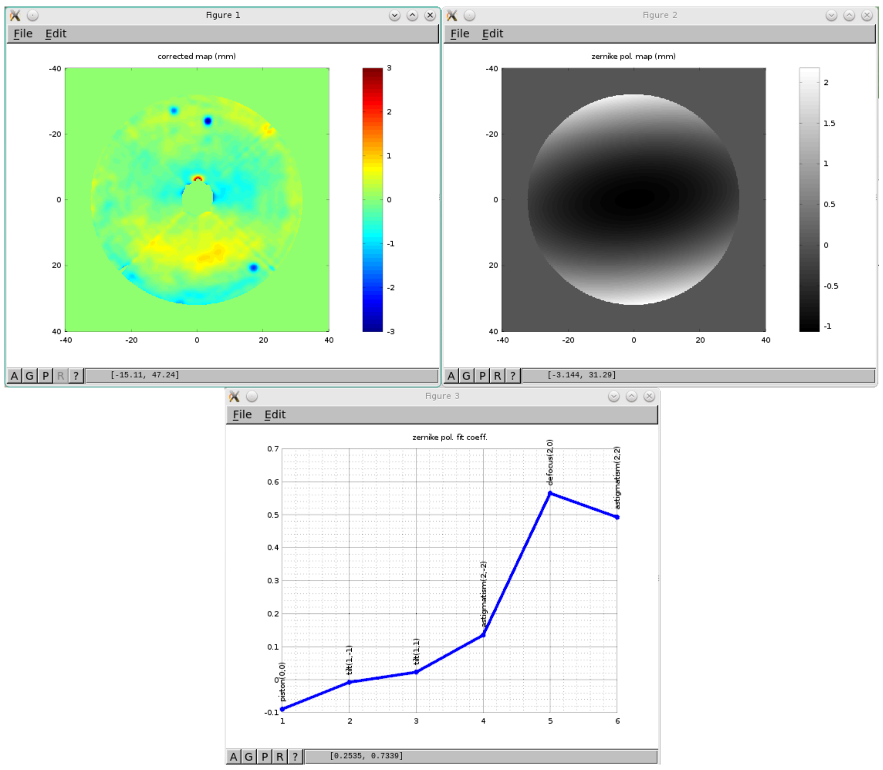


Figure 13: *zern* plots: error map after removing the large-scale deformations (top-left), map of the removed large-scale deformations depicted by Zernike polynomials (top-right), Zernike coefficients used for the large-scale deformations analysis (bottom).

```
octave:5> zern('param')
********************************
small scale RMS = 0.357700 [mm]
large scale RMS = 0.751653 [mm]
global RMS = 0.826307 [mm]
********************************
```

First of all, *zern* prints on the display screen the surface RMS of the small- and large-scale deformations and even the global one. Then, it plots in figure 13 three panels: the deformation map after removing the large-scale deformations (top-left) and the map of the removed large-scale deformations (top-right). Note that coma is still present, and the analysis could go further by considering more polynomials (e.g. Z(3,1), see table 3). The third panel (bottom) represents the coefficients of the Zernike polynomials considered in this analysis. Let's suppose to use the AS system to correct even the defocus and astigmatism aberrations, then we run again *zern* removing only tilt and piston from the error map.

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

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

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

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

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

Finally *lut* creates the lut.txt file containing deflections or corrections depending on the sign of $sfactor$. Taking a look at the lut.txt file, the first two columns are the actuator codes, the columns from 3th to 6th are the result of the panel fitting evaluated at the position of each actuator. The 7th column is the average the previuos four 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.

```
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
```

It is worth noting that astigmatism (but also defocus and coma) might also be due to holographic receiver feed displacements and/or feed phase center problems [12]. 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 by reaching the optimal feed focusing, before correcting for the large-scale deformations.

References

[1] Bolli P., Orlati A., Stringhetti L., et al., *Sardinia Radio Telescope: General Description, Technical Commissioning and First Light*, Journal of Astronomical Instrumentation, **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, **608**, A40, 2017.

[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 Lopez-Perez 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] Rochblatt D. J., et al. *Effects of Measurement Errors on Microwave Antenna Holography*, IEEE Trans. Antennas Propagat., **39**, 933—942, 1991.

[9] Herraez M. A., Burton D. R., Lalor M. J. and Gdeisat M. A. *Fast two-dimensional phase-unwrapping algorithm based on sorting by reliability following a noncontinuous path*, Applied Optics, **41**, 7437—7444, 2002.

[10] Juarez-Salazar R., Robledo-Sanchez C., and Guerrero-Sanchez F. *Phase-unwrapping algorithm by a rounding-least-squares approach*, Optical Engineering, **53**, 024102, 2014.

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

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