IJCSN International Journal of Computer Science and Network, Volume 3, Issue 4, August 2014
ISSN (Online) : 2277-5420 **www.IJCSN.org**
Impact Factor: 0.274

Processing HDF to FITS Image: Python Pipeline Mode

[1] Alexander Akoto-Danso, [2] Felix Tetteh Madjitey, [3] Emmanuel Proven-Adziri , [4] Theophilus Ansah-Narh, [5] Marcellin Atemkeng

[1, 2, 3] Ghana Space Science and Technology Institute (GSSTI), Ghana Atomic Energy Commission (GAEC).

[4, 5] RATT, Rhodes University, South Africa.

Abstract - Research scientists and students especially those in astronomy normally use image data in Flexible Image Transport System (FITS) format for their simulations. Simulation tools like Meqtrees, Oskar, Map simulator and Swarp all run using FITS file. Sometimes, the image data is stored in different formats such as Hierarchical Data Format (HDF) and other formats. We need to convert the data into FITS form before using for simulations. The paper therefore presents a python program on how to extract data from HDF into FITS file. The h5py helps us to extract the image data from HDF, numpy then converts the data into an array and pyfits helps us to write the extracted data in FITS format. The h5py, numpy and pyfits are imported into python for the conversion to take place. The developed code and output are presented in Appendices A and B respectively.

Keywords - *Hierarchical Data Format (HDF), Flexible Image Transport System (FITS), Python, h5py, numpy, pyfits.*

1. Introduction

Hierarchical Data Format (HDF) was formally developed at the National Center for Supercomputing Applications with its former version being HDF4 and now changed to HDF5. It is designed to store any large amount of scientific data in an organized form. The current version, HDF5 simplifies the file structure into two major types of objects:

- *Datasets*:- these are multidimensional arrays of a homogenous type.
- *Groups*:- these are structures in a form of containers that hold the datasets and other groups.

Detailed explanations on how to use HDF format are [1 and 2].

Most simulation tools such as MeqTrees [3], OSKAR [4], MIT Array Performance Simulator (MAPS) [5] and SWarp [6] all run using FITS file. Sometimes, the image data is stored in different formats such as HDF and other formats and we need to convert the data into FITS form before using for simulations.

FITS is also used to store both image and non-image data, such as spectra, photon lists, data cubes and multi-table databases. It is one of the commonest file format used in astronomy. Detailed explanations on how to use FITS file are [7, 8 and 9].

2. Program Structure

Figure 1 is a flow chart describing clearly how HDF is converted into FITS file. The h5py, numpy and pyfits are python modules or packages that must be installed to be able to import them into the python platform for conversion.

Appendix A shows the python script developed to convert HDF to FITS file.

3. Results and Discussions

Appendices B – 1 to B – 4 show the outputs of the FITS file created. Appendix B – 1 shows the running process during the conversion by going through the flowchart in Fig. 1. Appendix B – 2 shows the output of the headers when no update is made and appendix B – 3 is the updated output of the headers using World Coordinate System (WCS) [10] which can always be updated to suit the required image data.

Appendix B – 4 is the image data in array form extracted into FITS file. The FITS file created consists of more *header* and *data units,* normally referred to as HDUs. The HDU consists of the header which describes the data contained in it, followed by the data itself.

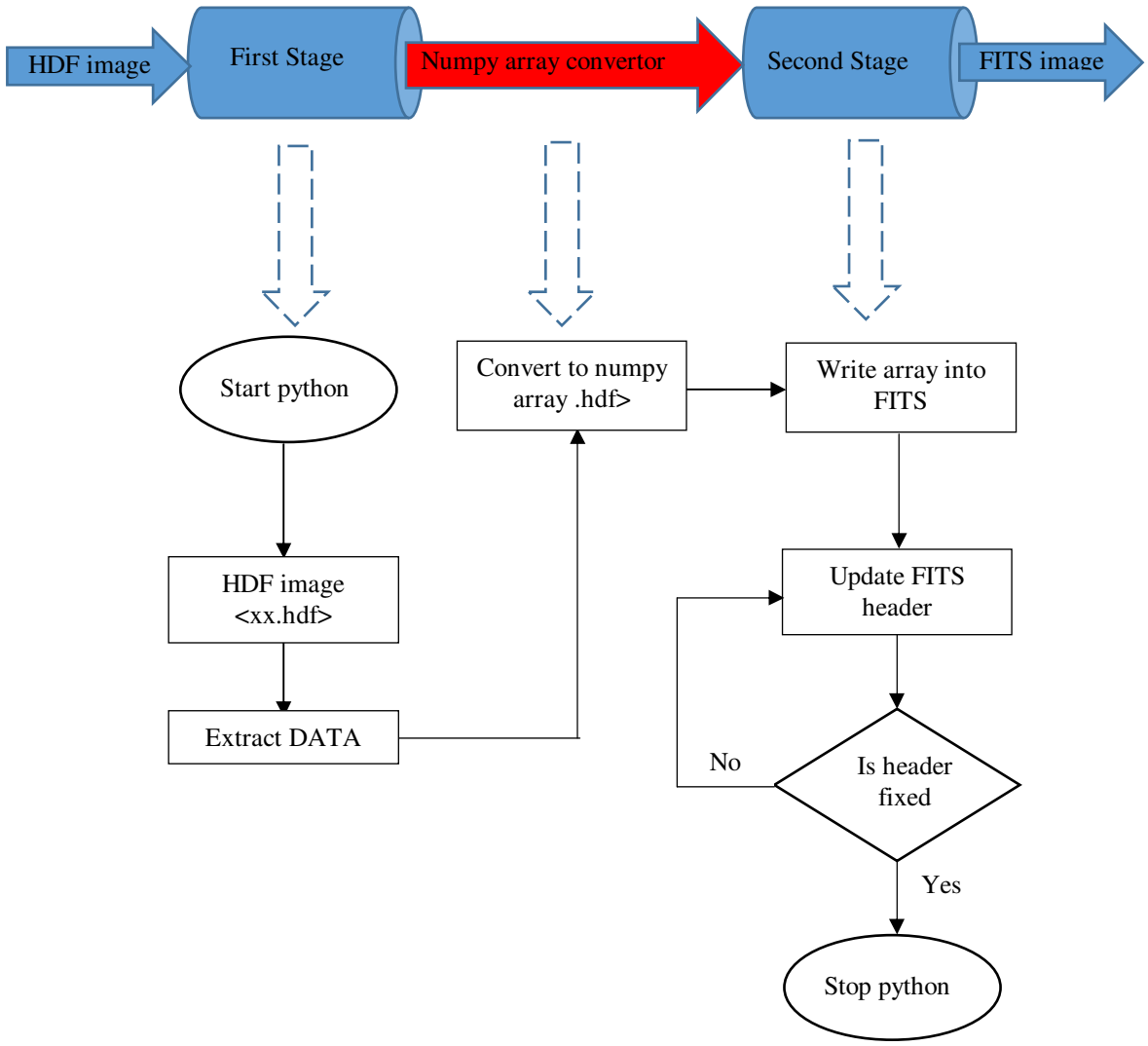

Fig. 1: Procedure for Converting HDF to FITS

4. Conclusion and Future Work

In this paper, we provided how to convert HDF to FITS using python module pyfits. Clearly, the paper showed the basic operations with FITS file such as *reading, modifying* and *writing an image data.*

The FITS file created in this paper is the Primary HDU and therefore, a continuation of this study is to develop a *Multi – Extension FITS* (MEF) file in *BINTABLE* format. This would be very important particularly in the application of *Hierarchical*, *Equal Area* and *Iso-Latitude Pixelisation* (HEALPIX) for spherical designs such as sky maps.

The National Center for Supercomputing Applications (NCSA) HDF group and others are currently working together to convert all HDF objects to FITS and vice versa, making it possible to access FITS files with HDF software and vice versa [11] but until then this paper should easily help in converting HDF to FITS.

Appendix A

```
#!/usr/bin/env python
"""
hdf2fits.py
=========
Converts HDF to FITS format
"""
__version__ = "1.0"
import h5py
```

```
import pyfits
import numpy as np
from pyfits import getdata, getheader
print " ----------------------------------------------------"
print "\n >> Please enter HDF5 file name to convert (including extension):"
print " ----------------------------------------------------"
filename=raw_input()    #Input .hdf5 file
print " >> Opening file and reading data"
print " ----------------------------------------------------"
hdf_data =h5py.File(filename,'r')
print " >> Extracting data from HDF file"
print " ----------------------------------------------------"
extract_data = hdf_data.values()[:]
print " >> Converting HDF data into arrays"
print " ----------------------------------------------------"
a_ray = np.array(extract_data)
##################################################################
print " >> Converting HDF data into FITS"
print " ----------------------------------------------------"
hdu = pyfits.PrimaryHDU(a_ray)
hdu.writeto('newhd2fits.fits')
##################################################################
print " ----------------------------------------------------"
print "\n >> Please checking & fixing headers:"
print " ----------------------------------------------------"
filename1='newhd2fits.fits'  #Input .fits file

print " >> Opening FITS file and reading header"
print " ----------------------------------------------------"

hdulist = pyfits.open(filename1)
in_header=hdulist[0].header
in_header.update('BITPIX', -64)
print " >> BITPIX Fixed\n >> Now checking for all required fits header keywords"
print " ----------------------------------------------------"

try:
   in_header['COMMENT']
except KeyError:
   print "\t No COMMENT Found, adding entry for this"
   in_header.update('COMMENT', 'Standard WCS reduction:')
try:
   in_header['BZERO']
except KeyError:
   print "\t No BZERO Found, adding entry for this"
   in_header.update('BZERO',0.0,'PhysValue = BZERO + BSCALE * ArrayValue')
try:
   in_header['BSCALE']
except KeyError:
   print "\t No BSCALE Found, adding entry for this"
   in_header.update('BSCALE',1.0,'PhysValue = BZERO + BSCALE * ArrayValue')

try:
   in_header['CRVAL1']
except KeyError:
   print "\t No CRVAL1 Found, adding entry for this"
   in_header.update('CRVAL1',11.88798983,'WCS Ref value (RA in decimal degrees)')
try:
   in_header['CRVAL2']
except KeyError:
   print "\t No 'EPOCH' Found, adding entry for this"
   in_header.update('EPOCH',1999.55329832624,'Epoch in Julain Year at start of observation')
try:
   in_header['CRPIX1']
except KeyError:
   print "\t No 'CRPIX1' Found, adding entry for this"
   in_header.update('CRPIX1',256.5,'WCS Coordinate reference pixel')
try:
   in_header['CRPIX2']
except KeyError:
   print "\t No 'CRPIX2' Found, adding entry for this"
   in_header.update('CRPIX2',356.5,'WCS Coordinate reference pixel')
try:
   in_header['CD1_1']
except KeyError:
   print "\t No 'CD1_1' Found, adding entry for this"
   in_header.update('CD1_1',-0.000277777783923599,'WCS Coordinate scale matrix')
try:
   in_header['CD1_2']
except KeyError:
   print "\t No 'CD1_2' Found, adding entry for this"
   in_header.update('CD1_2',1.78947724260645E-08, 'WCS Coordinate scale matrix')
try:
   in_header['CD2_1']
except KeyError:
   print "\t No CD2_1 Found, adding fake entry for this"
   in_header.update('CD2_1',1.78947724260645E-08,'WCS Coordinate scale matrix')
try:
   in_header['CD2_2']
except KeyError:
   print "\t No CD2_2 Found, adding fake entry for this"
```

```
in_header.update('CD2_2',0.000277777783923599,'WCS
Coordinate scale matrix')
try:
   in_header['CD3_1']
except KeyError:
   print "\t No 'CD3_1' Found, adding entry for this"
   in_header.update('CD3_1',1.78947724260645E-08,
'WCS Coordinate scale matrix')
try:
   in_header['CD3_2']
except KeyError:
   print "\t No 'CD3_2' Found, adding entry for this"
   in_header.update('CD3_2',1.78947724260645E-08,
'WCS Coordinate scale matrix')
try:
   in_header['CTYPE1']
except KeyError:
   print "\t No CTYPE1 Found, adding  entry for this"
   in_header.update('CTYPE1','RA---SIN', 'WCS
Coordinate type')
try:
   in_header['CTYPE2']
except KeyError:
   print "\t No CTYPE2 Found, adding  entry for this"
   in_header.update('CTYPE2','DEC---SIN', 'WCS
Coordinate type')
try:
   in_header['EQUINOX']
except KeyError:
   print "\t No EQUINOX Found, adding entry for this"
   in_header.update('EQUINOX',2000.0,' Equinox')
try:
   in_header['CDELT1']
except KeyError:
   print "\t No CDELT1 Found, adding entry for this"
   in_header.update('CDELT1',0.0002777777845,'WCS
Coordinate scale matrix')
try:
   in_header['CDELT2']
except KeyError:
   print "\t No CDELT2 Found, adding entry for this"
   in_header.update('CDELT2',0.0002777777845,'WCS
Coordinate scale matrix')
try:
   in_header['CTYPE3']
except KeyError:
   print "\t No CTYPE3 Found, adding fake entry for this"
   in_header.update('CTYPE3','STOKES', 'axes 3 is the
spectra')
try:
   in_header['CNPIX1']
except KeyError:
   print "\t No CNPIX1 Found, adding fake entry for this"
   in_header.update('CNPIX1','0','New CNPIX1')
try:
 in_header['MJD-OBS']
except KeyError:
   print "\t No MJD-OBS Found, adding fake entry for
this"
   in_header.update('MJD-
OBS',51381.3422136574,'modified Julian date at start of
observation')
try:
   in_header['CRPIX3']
except KeyError:
   print "\t No CRPIX3 Found, adding fake entry for this"
   in_header.update('CRPIX3',1.0)
################################
try:
   in_header['CUNIT1']
except KeyError:
   print "\t No CUNIT1 Found, adding entry for this"
   in_header.update('CUNIT1','DEG', 'RA coordinate')

try:
   in_header['CUNIT2']
except KeyError:
   print "\t No CUNIT2 Found, adding entry for this"
try:
   in_header['CDELT3']
except KeyError:
   print "\t No CDELT3 Found, adding entry for this"
   in_header.update('CDELT3', 1.0)
try:
   in_header['CRPIX1']
except KeyError:
   print "\t No CRPIX1 Found, adding entry for this"
   in_header.update('CRPIX1', 256.5)
try:
   in_header['CRPIX2']
except KeyError:
   print "\t No CRPIX2 Found, adding entry for this"
################################################
####

print " >> All missing required fits header keywords
added\n >> except NAXISn. I'll check in a second."
print " ----------------------------------------------------"

data=getdata(filename1) #Read in fits data
fl_data=np.float64(data) #convert in array to float64
hdu_data=pyfits.PrimaryHDU(fl_data) #create FITS type
data

print " >> Data array Fixed"
```

```
print " ----------------------------------------------------"

naxisvals=fl_data.shape
print naxisvals[0]
print naxisvals[1]

try:
   in_header['NAXIS1']
except KeyError:
   print "\t No NAXIS1 Found, adding entry for this"
   in_header.update('NAXIS1',naxisvals[0])

try:
   in_header['NAXIS2']
except KeyError:
   print "\t No NAXIS2 Found, adding entry for this"
   in_header.update('NAXIS2',naxisvals[0])
try:
   in_header['NAXIS3']
except KeyError:
   print "\t No NAXIS3 Found, adding entry for this"
   in_header.update('NAXIS3',naxisvals[0])
print " >> NAXIS values now checked."
print " ----------------------------------------------------"
hdulist.close()
pyfits.writeto('fixed_'+filename1,fl_data,in_header)
#Write new fits file


print " >> Done!\n\n >> The corrected fits file is named fixed_"+filename1
print " ----------------------------------------------------"
```

Appendix B – 1

```
narh@elwood:~/theo/comp_sc$ python hdf2fits.py
 ----------------------------------------------------

 >> Please enter HDF5 file name to convert (including extension):
 ----------------------------------------------------
pol_map.hdf5
 >> Opening file and reading data
 ----------------------------------------------------
 >> Extracting data from HDF file
 ----------------------------------------------------
 >> Converting HDF data into arrays
 ----------------------------------------------------
 >> Converting HDF data into FITS
 ----------------------------------------------------
 ----------------------------------------------------

 >> Please checking & fixing headers:
 ----------------------------------------------------
 >> Opening FITS file and reading header
 ----------------------------------------------------
 >> BITPIX Fixed
 >> Now checking for all required fits header keywords
 ----------------------------------------------------
        No COMMENT Found, adding entry for this
        No BZERO Found, adding entry for this
        No BSCALE Found, adding entry for this
        No CRVAL1 Found, adding entry for this
        No 'CRVAL2' Found, adding entry for this
        No 'EPOCH' Found, adding entry for this
        No 'CRPIX1' Found, adding entry for this
        No 'CRPIX2' Found, adding entry for this
        No 'CD1_1' Found, adding entry for this
        No 'CD1_2' Found, adding entry for this
        No CD2_1 Found, adding  entry for this
        No CD2_2 Found, adding  entry for this
        No 'CD3_1' Found, adding entry for this
        No 'CD3_2' Found, adding entry for this
        No CTYPE1 Found, adding  entry for this
        No CTYPE2 Found, adding  entry for this
        No EQUINOX Found, adding entry for this
        No CDELT1 Found, adding entry for this
        No CDELT2 Found, adding entry for this
        No CTYPE3 Found, adding entry for this
        No CNPIX1 Found, adding entry for this
        No CNPIX2 Found, adding entry for this
        No RADESYS Found, adding entry for this
        No MJD-OBS Found, adding entry for this
        No CRPIX3 Found, adding entry for this
        No CUNIT1 Found, adding entry for this
        No CUNIT2 Found, adding entry for this
        No CDELT3 Found, adding entry for this
 >> All missing required fits header keywords added
 >> except NAXISn. I'll check in a second.
 ----------------------------------------------------
 >> Data array Fixed
 ----------------------------------------------------
1
560
 >> NAXIS values now checked.
 ----------------------------------------------------
 >> Done!

 >> The corrected fits file is named fixed_newhd2fits.fits
 ----------------------------------------------------
```

Appendix B – 2

```
In [1]: import pyfits

In [2]: hdu = pyfits.open('newhd2fits.fits')
```

In [2]: hdu = pyfits.open('newhd2fits.fits')

In [3]: print hdu[0].header
SIMPLE = T / conforms to FITS standard

BITPIX = -64 / array data type

NAXIS = 4 / number of array dimensions

NAXIS1 = 786432

NAXIS2 = 4

NAXIS3 = 560

NAXIS4 = 1

EXTEND = T

Appendix B – 3

In [4]: hduf = pyfits.open('fixed_newhd2fits.fits')

In [5]: print hduf[0].header

SIMPLE = T / conforms to FITS standard

BITPIX = -64 / array data type

NAXIS = 4 / number of array dimensions

NAXIS1 = 786432

NAXIS2 = 4

NAXIS3 = 560

NAXIS4 = 1

CRVAL1 = 11.88798983 / WCS Ref value (RA in decimal degrees)

CRVAL2 = -25.33718377 / WCS Ref value (DEC in decimal degrees)

EPOCH = 1999.55329832624 / Epoch in Julain Year at start of observation

CRPIX1 = 256.5 / WCS Coordinate reference pixel

CRPIX2 = 356.5 / WCS Coordinate reference pixel

CD1_1 = -0.00027777778392359 / WCS Coordinate scale matrix

CD1_2 = 1.78947724260645E-08 / WCS Coordinate scale matrix

CD2_1 = 1.78947724260645E-08 / WCS Coordinate scale matrix

CD2_2 = 0.000277777783923599 / WCS Coordinate scale matrix

CD3_1 = 1.78947724260645E-08 / WCS Coordinate scale matrix

CD3_2 = 1.78947724260645E-08 / WCS Coordinate scale matrix

CTYPE1 = 'RA---SIN' / WCS Coordinate type

CTYPE2 = 'DEC---SIN' / WCS Coordinate type

EQUINOX = 2000.0 / Equinox

CDELT1 = 0.0002777777845 / WCS Coordinate scale matrix

CDELT2 = 0.0002777777845 / WCS Coordinate scale matrix

CTYPE3 = 'STOKES ' / axes 3 is the spectra

CNPIX1 = '0 ' / New CNPIX1

CNPIX2 = '0 ' / New CNPIX2

RADESYS = 'FK5 ' / Coordinate system

MJD-OBS = 51381.3422136574 / modified Julian date at start of observation

CRPIX3 = 1.0

CUNIT1 = 'DEG ' / RA coordinate

CDELT3 = 1.0

COMMENT Standard WCS reduction:

Appendix B – 4

In [6]: print hduf[0].data

```
[[[[ 1.61262884e+04   1.59303920e+04   1.55145821e+04
...,

     1.18152153e+04   1.22493922e+04   1.24867062e+04]

   [  1.14865993e+01   1.89759293e+01   1.57714412e+01
...,

   -5.58747410e+01  -5.59940744e+01  -5.55064362e+01]

   [  7.43384419e+01   6.80194419e+01   6.34424053e+01
...,

   -3.51272863e+01  -3.31921756e+01  -3.74518863e+01]

   [  0.00000000e+00   0.00000000e+00   0.00000000e+00
...,

    0.00000000e+00                     0.00000000e+00
0.00000000e+00]]

  [[            1.39059342e+04                1.37400705e+04
1.33806071e+04 ...,

    1.02468788e+04   1.06163441e+04   1.08212835e+04]

   [  1.23021996e+01   1.98147810e+01   1.66081076e+01
...,

   -5.71332849e+01  -5.71451130e+01  -5.66618905e+01]

   [7.53536529e+01   6.86724097e+01   6.38527579e+01
...,

   -3.53120905e+01  -3.32438123e+01  -3.74934783e+01]

   [  0.00000000e+00   0.00000000e+00   0.00000000e+00
...,

    0.00000000e+00                     0.00000000e+00
0.00000000e+00]]

  [[1.20725163e+04   1.19310725e+04   1.16182750e+04
...,
 8.94446192e+03   9.26105631e+03   9.43923397e+03]

[1.13916453e+01   1.88206657e+01   1.57066625e+01 ...,
  -5.54365238e+01 -5.53249321e+01  -5.48905659e+01]

   [7.71029006e+01   7.06927265e+01   6.57704095e+01
...,

   -3.64073943e+01  -3.44865497e+01  -3.87505005e+01]

   [ 0.00000000e+00   0.00000000e+00   0.00000000e+00
...,
 0.00000000e+00   0.00000000e+00   0.00000000e+00]]
 ...,
  [[ 4.93552468e-01   4.96426918e-01   4.84608608e-01
...,
 4.00771575e-01   3.95754920e-01   4.08489146e-01]
 [6.31610901e-02   6.13774389e-02   5.80929026e-02 ...,
 -2.02320153e-02  -1.94703505e-02  -1.80827997e-02]
   [3.55320610e-02     3.19743144e-02     2.87056667e-02
...,

    1.36039364e-03   3.28667625e-03   2.06490153e-04]

   [0.00000000e+00   0.00000000e+00   0.00000000e+00
...,
 0.00000000e+00   0.00000000e+00   0.00000000e+00]]


  [[4.90875403e-01     4.93740518e-01     4.81988073e-01
...,

    3.98474596e-01   3.93474828e-01   4.06148925e-01]

   [6.29019447e-02     6.11264250e-02     5.78342543e-02
...,

    -2.03858111e-02  -1.96372520e-02  -1.82412362e-02]

   [3.56680031e-02     3.21116345e-02     2.88477180e-02
...,

    1.04425912e-03   2.95573873e-03  -1.24522009e-04]

   [0.00000000e+00    0.00000000e+00     0.00000000e+00
...,
 0.00000000e+00   0.00000000e+00   0.00000000e+00]]

  [[4.88217392e-01     4.91073208e-01     4.79386150e-01
...,

    3.96194360e-01   3.91211416e-01   4.03825772e-01]

   [6.26412185e-02     6.08738353e-02     5.75741862e-02
...,
```

-2.05357473e-02 -1.98003038e-02 -1.83959080e-02]

[3.57999089e-02 3.22452369e-02 2.89860593e-02
...,

7.33478306e-04 2.63014243e-03 -4.49641487e-04]

[0.00000000e+00 0.00000000e+00 0.00000000e+00
...,
0.00000000e+00 0.00000000e+00 0.00000000e+00]]]]

Acknowledgment

This research was supported by the facilities provided by Rhodes University in South Africa.

References

[1] http://www.hdfgroup.org/ftp/HDF5/examples/
[2] http://www.hdfgroup.org/hdf-java-html/hdfview
[3] meqtrees.net
[4] http://oerc.ox.ac.uk/~ska/oskar2/
[5] http://www.haystack.mit.edu/ast/arrays/maps/
[6] http://www.astromatic.net/software/swarp
[7] http://fits.gsfc.nasa.gov/fits_documentation.html
[8] http://fits.gsfc.nasa.gov/
[9] http://www.digitalpreservation.gov/formats/fdd/fdd000317.shtml
[10] Calabretta, M. R. and Greisen, E. W. (2002). "Representations of world coordinates in FITS", *Astronomy & Astrophysics*, Vol. 395, pp. 1061–1075.
[11] http://www.hdfgroup.org/ftp/newsletters/archive/Newsletter18.txt

Bibliography

Alexander Akoto-Danso works as an Assistant Research Scientist at the Ghana Space and Technology Institute. He holds a Master's in Computational Nuclear Science and Engineering (2011) and holds a Bsc. Mathematics. (2006) His interest is in High-performance Computing, Parallel computing, Computational Engineering, Digital Signal Processing, Climate modelling, Industrial and Systems engineering. He is a member of the Ghana Nuclear Society.

Felix Tetteh Madjitey works as an Assistant Research Scientist at the Ghana Space and Technology Institute. He holds a Master's in Computational Nuclear Science and Engineering and holds a Bsc. Physics. His research interest is in Structural and Mechanical Engineering.

Emmanuel Proven- Adzri is working as an Assistant Research Scientist at Ghana Space Science and Technology Institute. He has Masters in Computational Nuclear Science and Engineering and a BSc. in Physics. His research interest includes Instrumentation in Radio Astronomy, Methanol Masers, Radio Interferometry and Computational Modeling.

Theophilus Ansah-Narh is working as an Assistant Research Scientist at Ghana Space Science and Technology Institute. He has Masters in Statistics and Computational Nuclear Science and Engineering and also, BSc. Mathematics and Computer Science. His research interest includes Radio Interferometry, Stochastic Modeling, Bayesian Analysis, Computational Mathematics and Parallel Computing.

Marcellin Atemkeng holds a MSc Computer Sc. and BSc Mathematics and Computer Sc. His research interest includes Networking, Structure Documents, Workflow Systems, Simulation, Signal Correlation Algorithm Techniques, Synthesis Imaging in Radio Interferometry Arrays and Data Reduction.