EXAMPLES USING THE IMOS USER CODE LIBRARY (Python VERSION)

Version 1.0

Prepared by *Laurent Besnard*

IMOS - eMII

15/05/2013

Integrated **Marine Observing** System

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

This document intends to present how to load IMOS NetCDF data into a Python environment, and offers some suggestions about how to use the data once loaded. All the examples below will use the netCDF4 Python module (http://code.google.com/p/netcdf4-python/)

The examples provided in this document only present a tiny bit of the content of most of the NetCDF files. There are usually many more variables available in a NetCDF file, and therefore many other ways to display data.

1.1 Installation of the IMOS User Code Library (Python) and required packages

The IMOS User Code Library for Python can be downloaded from:
https://github.com/aodn/imos_user_code_library/tree/master/Python/
Each dataset-specific example below can be found in the **demos** folder as a separate Python script.

The code library can be checked out using a Git client, or be downloaded as a zip file :
https://github.com/aodn/imos_user_code_library/archive/master.zip

The examples rely on the following Python packages, which need to be installed:

- **numpy** – standard package for scientific computing in Python, provides versatile numerical array objects (**http://www.numpy.org/**).
- **matplotlib** – for plotting (http://matplotlib.org/).
- **netCDF4** – for accessing netCDF files (http://code.google.com/p/netcdf4-python/, follow the Documentation link for installation instructions).

The examples below have been tested in Python 2.7.3, with numpy 1.6.1, matplotlib 1.1.1, and version 1.0.2 of netCDF4. Later versions of these packages will usually work. Earlier versions *may* work also.

1.2 Finding an IMOS NetCDF File

In order to find a dataset you are interested in, please refer to the portal help:
http://portalhelp.aodn.org.au/Portal2_help/
A HOWTO has been written to help any user in his way to find an IMOS NetCDF file.

For users who are already familiar with IMOS facilities and datasets, IMOS NetCDF files are also directly accessible via an OPeNDAP catalog at :
http://thredds.aodn.org.au/thredds/catalog/IMOS/catalog.html

Once a NetCDF file has been chosen to work with (See http://portalhelp.aodn.org.au/Portal2_help/?q=node/112), the user needs to go to the ***'OPeNDAP Dataset Access Form'*** page. The **'Data URL'** located just above the **'Global Attributes'** field is the URL which needs to be replaced in the examples which we'll present in the following sections.

The NetCDF file can also be downloaded to the user’s local machine using the HTTP Server option on the THREDDS options page. In this case, the user has to replace the 'Data URL' variable with the local address of the NetCDF file when using the function '**Dataset**' in the following examples.

2. General Features of the netCDF4 module

The first step consists of opening a NetCDF file, whether this file is available locally or remotely on an OPeNDAP server.

Type in your Python command window:

```
from netCDF4  import Dataset

aatams_URL = 'http://thredds.aodn.org.au/thredds/dodsC/IMOS/eMII/demos/AATAMS/marine_mammal_ctd-tag/2009_2011_ct64_Casey_Macquarie/ct64-M746-09/IMOS_AATAMS-SATTAG_TSP_20100205T043000Z_ct64-M746-09_END-20101029T071000Z_FV00.nc'
aatams_DATA = Dataset(aatams_URL)
```

This creates a netCDF `Dataset` object, through which you can access all the contents of the file.

2.1 Output structure

Please refer to the netCDF4 module documentation for a complete description of the `Dataset` object:
http://netcdf4-python.googlecode.com/svn/trunk/docs/netCDF4.Dataset-class.html
(or type `help(Dataset)` at the Python prompt).

2.2 Discover Metadata

In order to see all the global attributes and some other information about the file, type in your command window:

```
print aatams_DATA
```

```
<type 'netCDF4.Dataset'>
root group (NETCDF3_64BIT file format):
    project: Integrated Marine Observing System (IMOS)
    conventions: IMOS-1.2
    date_created: 2012-09-13T07:27:03Z
    title: Temperature, Salinity and Depth profiles in near real time
    institution: AATAMS
    site: CTD Satellite Relay Data Logger
    abstract: CTD Satellite Relay Data Loggers are used to explore how marine mammal behaviour
relates to their oceanic environment. Loggers developped at the University of St Andrews Sea
Mammal Research Unit transmit data in near real time via the Argo satellite system
    source: SMRU CTD Satellite relay Data Logger on marine mammals
…
    dimensions: obs, profiles
    variables: TIME, LATITUDE, LONGITUDE, TEMP, PRES, PSAL, parentIndex,
TIME_quality_control, LATITUDE_quality_control, LONGITUDE_quality_control,
TEMP_quality_control, PRES_quality_control, PSAL_quality_control
    groups:
```

Global attributes in the netCDF file become attributes of the `Dataset` object. A list of global attribute names is returned by the `ncattrs()` method of the object. The standard `__dict__` attribute of the object is a dictionary of all netCDF attribute names and values.

```
# store the dataset's title in a local variable
title_str = aatams_DATA.title

# list all global attribute names
aatams_DATA.ncattrs()

# store the complete set of attributes in a dictionary (OrderedDict) object (similar to a standard Python dict, but
# maintains the order in which items are entered)
globalAttr = aatams_DATA.__dict__

# now you can also do (same effect as first command above)
title_str = globalAttr['title']
```

2.3 Discover Variables

To list all the variables available in the NetCDF file, type:

```
aatams_DATA.variables.keys()
```

```
[u'TIME',
u'LATITUDE',
u'LONGITUDE',
u'TEMP',
u'PRES',
u'PSAL',
u'parentIndex',
u'TIME_quality_control',
u'LATITUDE_quality_control',
u'LONGITUDE_quality_control',
u'TEMP_quality_control',
u'PRES_quality_control',
u'PSAL_quality_control']
```

(The 'u' means each variable name is represented by a Unicode string.)

Each variable is accessed via a `Variable` object, in a similar way to the `Dataset` object. To access the Temperature variable :

```
# netCDF4 Variable object
TEMP = aatams_DATA.variables['TEMP']

# now you can print the variable's attributes and other info
print TEMP

# access variable attributes, e.g. its standard_name
TEMP.standard_name

# extract the data values (as a numpy array)
TEMP[:]

# the variable's dimensions (as a tuple)
TEMP.dimensions
```

-

3. Dataset examples – Using the NetCDF Parser for Plotting

3.1 AATAMS – Animal Tagging and Monitoring - non QC'd data

The Australian Animal Tagging And Monitoring System (AATAMS) is a coordinated marine animal tagging project. CTD Satellite Relay Data Loggers are used to explore how marine mammal behaviour relates to their oceanic environment.

NetCDF files can be found at :
http://thredds.aodn.org.au/thredds/catalog/IMOS/AATAMS/marine_mammal_ctd-tag/catalog.html

In the example below, we demonstrate how to use the netCDF4 module to plot all the animal's dives as a single profile time-series of temperature, measured by CTD tag.

```
from netCDF4 import Dataset
from datetime import datetime, timedelta
from pylab import *
import numpy
import matplotlib.pyplot as plt
from imosNetCDF import *

## AATAMS - Animal Tagging and Monitoring
aatams_URL = 'http://thredds.aodn.org.au/thredds/dodsC/IMOS/eMII/demos/AATAMS/marine_mammal_ctd-
tag/2009_2011_ct64_Casey_Macquarie/ct64-M746-09/IMOS_AATAMS-SATTAG_TSP_20100205T043000Z_ct64-M746-
09_END-20101029T071000Z_FV00.nc';
aatams_DATA = Dataset(aatams_URL)
metadata = getAttNC(aatams_DATA)

nProfiles = len(aatams_DATA.dimensions['profiles']) # the number of profiles undertaken by the seal
parentIndex = aatams_DATA.variables['parentIndex'][:] #for each obs which profile it is linked to

# loading of the variable objects
TEMP = aatams_DATA.variables['TEMP']
PRES = aatams_DATA.variables['PRES']
PSAL = aatams_DATA.variables['PSAL']
TIME = aatams_DATA.variables['TIME']

# creation of a 2 dimension array for temperature, pressure and salinity
psalData = aatams_DATA.variables['PSAL'][:]
tempData = aatams_DATA.variables['TEMP'][:]
presData = aatams_DATA.variables['PRES'][:]

# we want to know the maximum number of observations (or depth level) per profile
# for all the profile. This number 'maxObsProfile' will be used to create a 2d
# array for Temperature salinity and pressure.
maxObsProfile = 0.
for profileNumber in range(1,nProfiles):
    indexVar = where(parentIndex == profileNumber)
    if size(indexVar) > maxObsProfile:
        maxObsProfile = size(indexVar)

# we recreate those variables to have a 2d array
TEMP_DATA_reshaped = numpy.empty((nProfiles,maxObsProfile,))
PSAL_DATA_reshaped = numpy.empty((nProfiles,maxObsProfile,))
PRES_DATA_reshaped = numpy.empty((nProfiles,maxObsProfile,))

for profileNumber in range(nProfiles):
    indexVar = where(parentIndex == profileNumber)
    TEMP_DATA_reshaped[profileNumber,0:size(indexVar)] = tempData[indexVar]
    PSAL_DATA_reshaped[profileNumber][range(0,size(indexVar))] = psalData[indexVar]
    PRES_DATA_reshaped[profileNumber][range(0,size(indexVar))] = presData[indexVar]

# we load the latitude and longitude values for all the profiles
latProfile =  numpy.array(aatams_DATA.variables['LATITUDE'][:])
lonProfile = numpy.array(aatams_DATA.variables['LONGITUDE'][:])

#longitude in the original dataset goes from -180 to +180
#For a nicer plot, we change the values to the [0 360] range
lonProfile[lonProfile < 0 ] = lonProfile[lonProfile < 0 ] +360

# we convert the time values into a python time object
timeData = convertTime(TIME) # one value per profile

# creation of a profile variable array
sizer = ones((1,maxObsProfile),'float')
#observation = range(nProfiles)
profIndex = array(range(nProfiles))
profIndex = profIndex.reshape(nProfiles,1)
prof_2D =  profIndex * sizer

## PLOT
#plot all the profiles as a timeseries
figure1 = figure(num=None, figsize=(15, 10), dpi=80, facecolor='w', edgecolor='k')
subplot(311)
pcolor(prof_2D, -PRES_DATA_reshaped, TEMP_DATA_reshaped)
cbar = colorbar()
cbar.ax.set_ylabel(TEMP.long_name + ' in ' + TEMP.units)
title(metadata['species_name'] + ' - released in ' + metadata['release_site'] +' \n animal
reference number : ' + metadata['unique_reference_code'])
xlabel('Profile Index')
ylabel(PRES.long_name + ' in negative ' + PRES.units)

from matplotlib.dates import MONTHLY, DateFormatter, rrulewrapper, RRuleLocator
rule = rrulewrapper(MONTHLY, bymonthday=1, interval=1)
formatter = DateFormatter('%d/%m/%y')
loc = RRuleLocator(rule)

#plot the LON timeseries
ax3 = subplot(234)
plot(timeData,lonProfile)
ax3.xaxis.set_major_locator(loc)
ax3.xaxis.set_major_formatter(formatter)
labels = ax3.get_xticklabels()
setp(labels, rotation=30, fontsize=10)
xlabel(TIME.long_name  + ' in ' +  'dd/mm/yy' )
ylabel(aatams_DATA.variables['LONGITUDE'].long_name + ' in ' +
aatams_DATA.variables['LONGITUDE'].units)

#plot the LAT timeseries
ax4 = subplot(235)
plot(timeData,latProfile)
ax4.xaxis.set_major_locator(loc)
ax4.xaxis.set_major_formatter(formatter)
labels = ax4.get_xticklabels()
setp(labels, rotation=30, fontsize=10)
xlabel(TIME.long_name  + ' in ' +  'dd/mm/yy' )
ylabel(aatams_DATA.variables['LATITUDE'].long_name  + ' in ' +
aatams_DATA.variables['LATITUDE'].units)

#plot the profile index with time values
ax5 = subplot(236)
plot(timeData,profIndex)
ax5.xaxis.set_major_locator(loc)
ax5.xaxis.set_major_formatter(formatter)
labels = ax5.get_xticklabels()
setp(labels, rotation=30, fontsize=10)
xlabel(TIME.long_name  + ' in ' +  'dd/mm/yy' )
```

```
ylabel('Profile Index')

#plot of a single profile
profileToPlot = 1# this is arbitrary. We can plot all profiles from 1 to nProfiles, modify
profileToPlot if desired
figure2 = figure(num=None, figsize=(7, 10), dpi=80, facecolor='w', edgecolor='k')
plot (TEMP_DATA_reshaped[profileToPlot,:],-PRES_DATA_reshaped[profileToPlot,:])
title(metadata['title'] + '\nlocation ' + "%0.2f" % latProfile[profileToPlot] + '/' + "%0.2f" %
lonProfile[profileToPlot] + '\n' + timeData[profileToPlot].strftime('%d/%m/%Y'))
xlabel(TEMP.long_name +  ' in ' + TEMP.units)
ylabel(PRES.long_name +  ' in negative ' + PRES.units)
plt.show()
```

Variables to modify :

- aatams_URL : the opendap url of the chosen file
- ProfileToPlot : the profile number to plot.

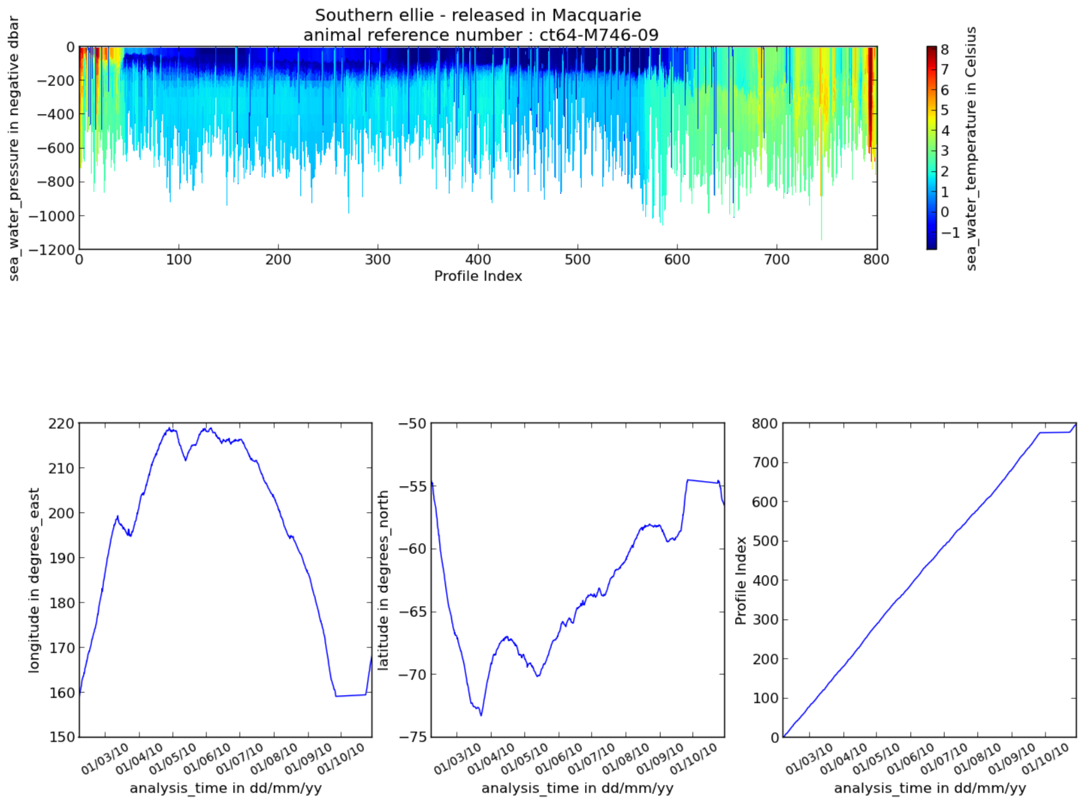


Illustration 1: Example of a Temperature Profile Time-series from AATAMS data

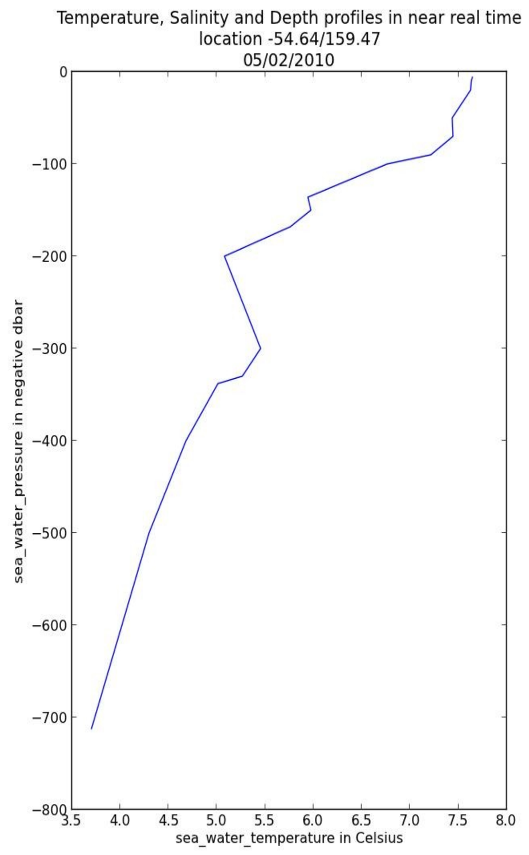


Illustration 2: Example of a Temperature profile from AATAMS data

3.2 ABOS – Deep Water Moorings

3.2.1 Southern Ocean Time-series - non QC'd data

The Southern Ocean Time Series (SOTS) sub-facility provides high temporal resolution observations in sub-Antarctic waters. Observations are broad and include measurements of physical, chemical and biogeochemical parameters from multiple deep-water moorings.

NetCDF files can be found at :
http://thredds.aodn.org.au/thredds/catalog/IMOS/ABOS/SOTS/catalog.html

In the example below, the netCDF4 module is used to extract temperature data from a Pulse mooring instrument and print the abstract for the dataset. A temperature time series plot is produced using matplotlib.

```
import numpy
from netCDF4 import Dataset, num2date
from matplotlib.pyplot import figure, subplot, plot, xlabel, ylabel, title, setp, show
from matplotlib.dates import MONTHLY, DateFormatter, rrulewrapper, RRuleLocator

############# ABOS
abos_URL = 'http://thredds.aodn.org.au/thredds/dodsC/IMOS/eMII/demos/ABOS/SOTS/Pulse/IMOS_ABOS-
SOTS_20110803T000000Z_PULSE_FV01_PULSE-8-2011_END-20120719T000000Z_C-20121009T214808Z.nc'
abos_DATA = Dataset(abos_URL)

tempDataStructure = abos_DATA.variables['TEMP_85_1']
TIME = abos_DATA.variables['TIME']

tempData = tempDataStructure[:]
timeData = num2date(TIME[:], TIME.units, TIME.calendar)

print abos_DATA.abstract

figure1 =figure( figsize=(10, 10), dpi=80, facecolor='w', edgecolor='k')
ax = subplot(111)

# exclude not-a-number (NaN) values from plot (otherwise it doesn't work)
indexNoNan = ~ numpy.isnan(tempData)
plot(timeData[indexNoNan],tempData[indexNoNan])

xlabel(TIME.long_name  + ' in ' +  'dd/mm/yy' )
ylabel(tempDataStructure.standard_name + ' in ' + tempDataStructure.units)
title(abos_DATA.title  + '\nat ' +  "%0.2f" %tempDataStructure.sensor_depth + ' m depth' )

# time ticks
rule = rrulewrapper(MONTHLY, bymonthday=1, interval=1)
formatter = DateFormatter('%d/%m/%y')
loc = RRuleLocator(rule)
ax.xaxis.set_major_locator(loc)
ax.xaxis.set_major_formatter(formatter)
labels = ax.get_xticklabels()
setp(labels, rotation=30, fontsize=10)
show()
```

Variables to modify :

- abos_URL : the opendap url of the chosen file
- tempDataStructure : we arbitrarily chose the variable called TEMP_85_1, but many more are available.

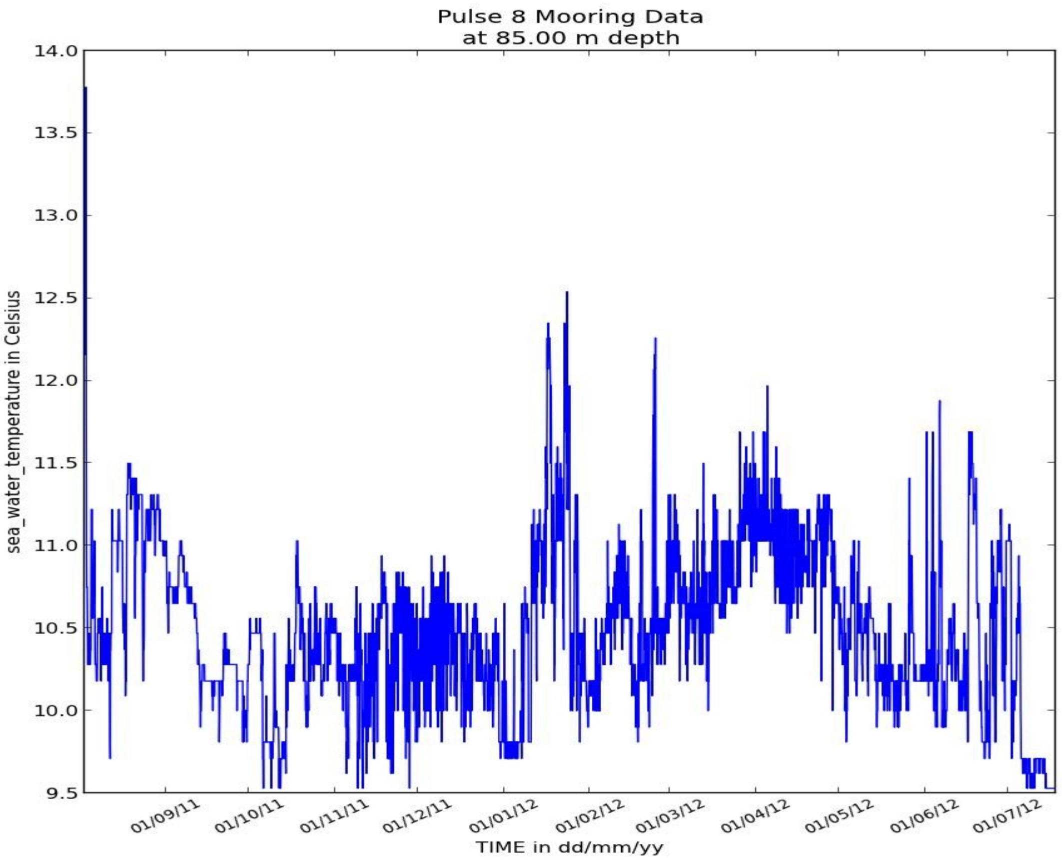


Illustration 3: Example of a Sea Water Temperature timeseries from a Pulse Mooring

3.3 ACORN – Ocean Radar - non QC'd data

The Australian Coastal Ocean Radar Network (ACORN) facility comprises a coordinated network of HF radars delivering real-time, non-quality controlled and delayed-mode, quality controlled surface current data into a national archive.

NetCDF files can be found at :
http://thredds.aodn.org.au/thredds/catalog/IMOS/ACORN/catalog.html

Monthly aggregated files are also available in the following folders:

- monthly_gridded_1h-avg-current-map_QC
- monthly_gridded_1h-avg-current-map_non-QC

In the example below, we demonstrate how to use the netCDF4 module to plot velocity data for one time value only in a latitude / longitude grid.

```
from netCDF4 import Dataset
from datetime import datetime, timedelta
from pylab import *
import numpy
import matplotlib.pyplot as plt
from imosNetCDF import *

############# ACORN
acorn_URL = 'http://thredds.aodn.org.au/thredds/dodsC/IMOS/eMII/demos/ACORN/monthly_gridded_1h-avg-current-
map_non-QC/TURQ/2012/IMOS_ACORN_V_20121001T000000Z_TURQ_FV00_monthly-1-hour-avg_END-
20121029T180000Z_C-20121030T160000Z.nc.gz'
acorn_DATA = Dataset(acorn_URL)
metadata = getAttNC(acorn_DATA)

SPEED = acorn_DATA.variables['SPEED']
LAT = acorn_DATA.variables['LATITUDE']
LON = acorn_DATA.variables['LONGITUDE']
TIME =  acorn_DATA.variables['TIME']

# Only one time value is being plotted. modify timeIndex if desired (value between 1 and length(timeData)
timeIndex = 4
speedData = SPEED[timeIndex,:,:]
latData = LAT[:]
lonData = LON[:]

# sea water U and V components
uData = acorn_DATA.variables['UCUR'][timeIndex,:,:]
vData = acorn_DATA.variables['VCUR'][timeIndex,:,:]

figure1 = figure( figsize=(13, 10), dpi=80, facecolor='w', edgecolor='k')
pcolor(lonData ,latData , speedData)
cbar = colorbar()
cbar.ax.set_ylabel(SPEED.long_name + ' in ' + SPEED.units)

title(metadata['title'] +'\n' + convertTime(TIME)[timeIndex].strftime('%d/%m/%Y'))
xlabel(LON.long_name +  ' in ' + LON.units)
ylabel(LAT.long_name +  ' in ' + LAT.units)
draw()

#plot velocity field
Q = quiver( lonData[:], latData[:], uData, vData, units='width')
show()
```

Variables to modify :

- acorn_URL : the opendap url of the chosen file
- timeIndex : the time index number to plot

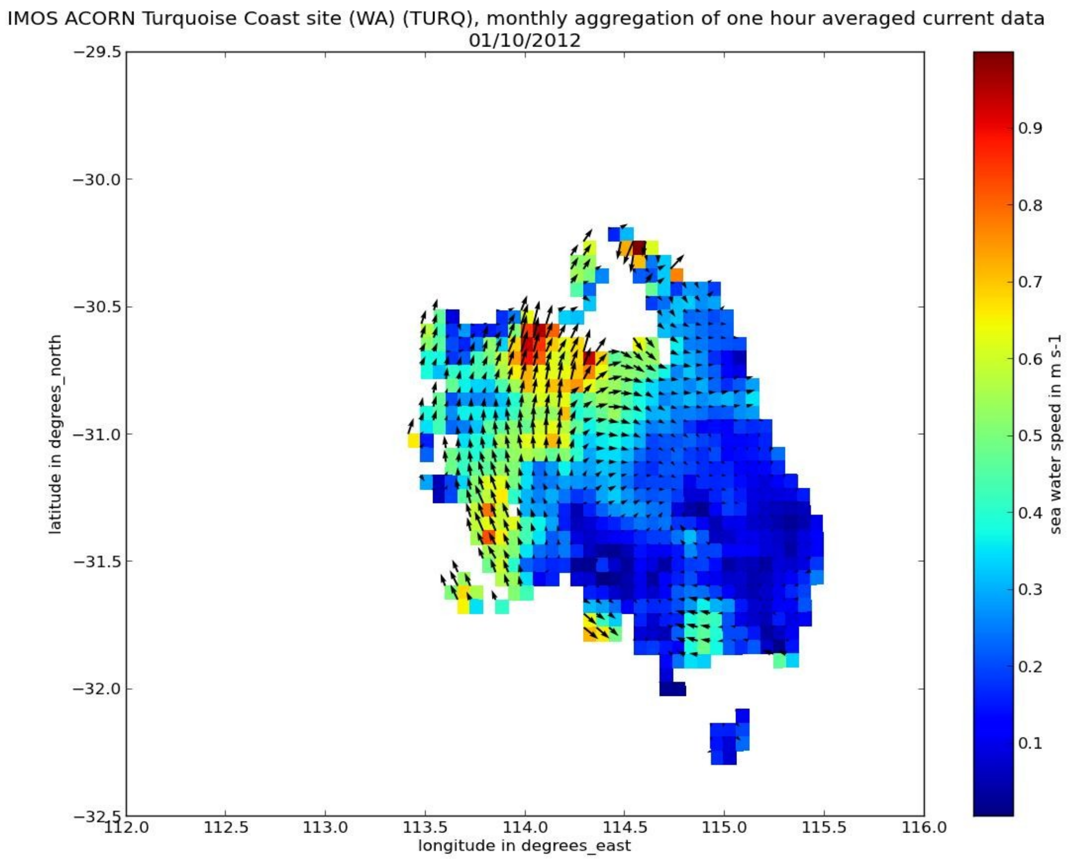


Illustration 4: Example of a Sea Water Speed gridded data with a Velocity Field from ACORN data

3.4 ANFOG – Ocean Gliders - QC'd good data

The Australian National Facility for Ocean Gliders (ANFOG), with IMOS/NCRIS funding, deploys a fleet of eight gliders around Australia.

NetCDF files can be found at :
http://thredds.aodn.org.au/thredds/catalog/IMOS/ANFOG/seaglider/catalog.html

In the example below, we demonstrate how to use the netCDF4 module to plot salinity data as well as depth data in a same graph. Only the data points with a Quality Control flag greater than 1 (which means 'good data' , please refers to IMOS NetCDF User Manual for a description of the Quality Control, available at http://imos.org.au/facility_manuals.html)

```
from netCDF4 import Dataset
from datetime import datetime, timedelta
from pylab import *
import numpy
import matplotlib.pyplot as plt
from imosNetCDF import *

############# ANFOG
anfog_URL =
'http://thredds.aodn.org.au/thredds/dodsC/IMOS/eMII/demos/ANFOG/seaglider/SOTS20110420/IMOS_ANFOG_BCEOST
UV_20110420T111022Z_SG517_FV01_timeseries_END-20110420T140511Z.nc'
anfog_DATA = Dataset(anfog_URL)
metadata = getAttNC(anfog_DATA)

PSAL = anfog_DATA.variables['PSAL']
DEPTH =  anfog_DATA.variables['DEPTH']
PSAL_qcFlag = anfog_DATA.variables['PSAL_quality_control']

qcLevel = 1 # we use the quality control flags to only select the good_data
index_qcLevel = where( PSAL_qcFlag[:] == qcLevel)

psalData = PSAL[index_qcLevel]
timeData = convertTime(anfog_DATA.variables['TIME'])[index_qcLevel]
depthData = DEPTH[index_qcLevel]

figure1 = figure( figsize=(13, 10), dpi=80, facecolor='w', edgecolor='k')

ax1 = figure1.add_subplot(111)
ax1.plot(timeData,psalData, 'b-')
ax1.set_xlabel('time (s)')
# Make the y-axis label and tick labels match the line color.
ax1.set_ylabel(PSAL.standard_name + ' in ' + PSAL.units, color='b')
for tl in ax1.get_yticklabels():
    tl.set_color('b')

ax2 = ax1.twinx()
ax2.plot(timeData,depthData, 'r.')
ax2.set_ylabel(DEPTH.standard_name + ' in ' + DEPTH.units, color='r')
for tl in ax2.get_yticklabels():
    tl.set_color('r')

xlabel(anfog_DATA.variables['TIME'].standard_name)
title(metadata['title'] +  ' starting at ' + metadata['time_coverage_start']  + 'UTC')
plt.show()
```

Variables to modify :

- anfog_URL : the opendap url of the chosen file

- qcLevel : quality control value (varies from 0 to 9)

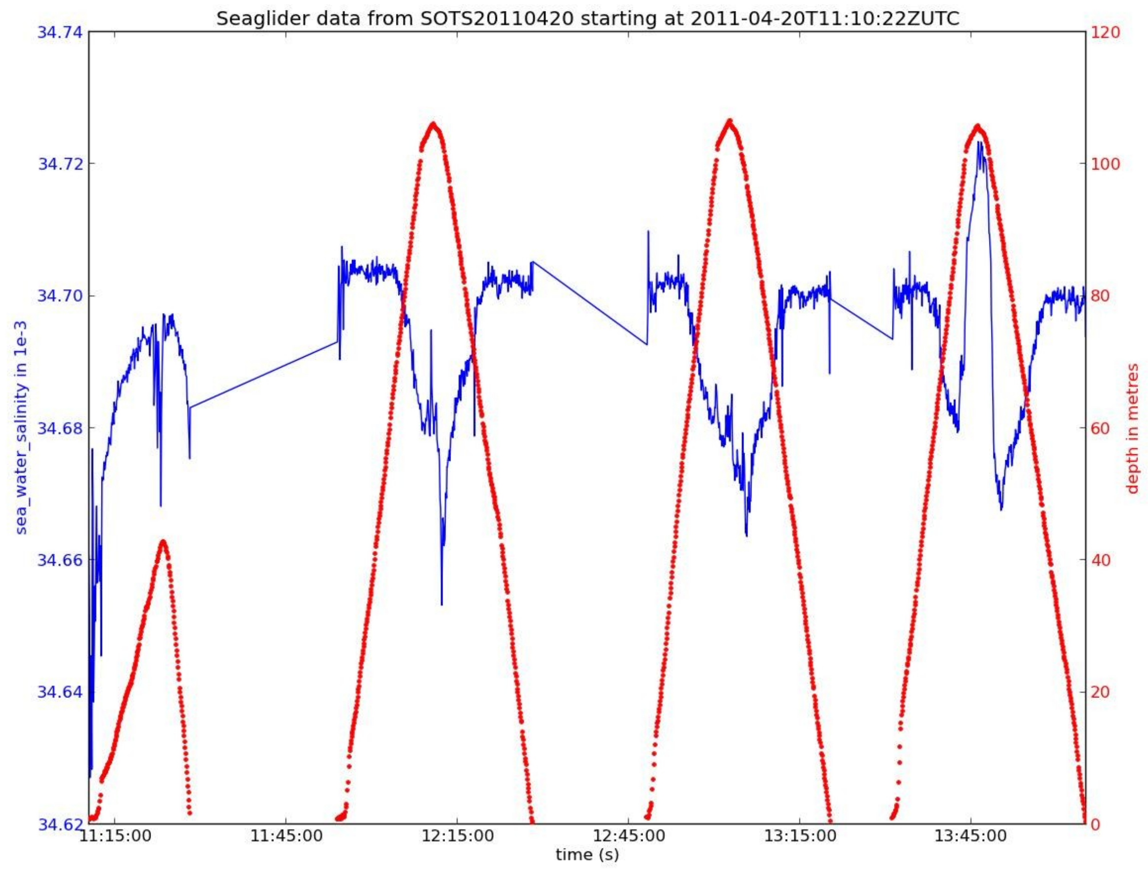


Illustration 5: Example of Sea Water Time-series taken during a SeaGlider Dive. Filtered to plot good data only

3.5 ANMN – National Mooring Network - QC'd good data

The Australian National Mooring Network Facility is a series of national reference stations and regional moorings designed to monitor particular oceanographic phenomena in Australian coastal ocean waters.

NetCDF files can be found at :
http://thredds.aodn.org.au/thredds/catalog/IMOS/ANMN/catalog.html

In the example below, we demonstrate how to use the netCDF4 module to plot the U current variable measured with an ADCP instrument (in Western Australia).

```
from netCDF4 import Dataset
from imosNetCDF import *
from datetime import datetime, timedelta
from pylab import *
import numpy
import matplotlib.pyplot as plt

anmn_URL = 'http://thredds.aodn.org.au/thredds/dodsC/IMOS/eMII/demos/ANMN/WA/WATR50/Velocity/IMOS_ANMN-
WA_VATPE_20120516T040000Z_WATR50_FV01_WATR50-1205-Workhorse-ADCP-498_END-20121204T021500Z_C-
20121207T023956Z.nc'
anmn_DATA = Dataset(anmn_URL)
metadata = getAttNC(anmn_DATA)

UCUR = anmn_DATA.variables['UCUR']
DEPTH = anmn_DATA.variables['HEIGHT_ABOVE_SENSOR']

uCurrentData = UCUR[:]
timeData = convertTime(anmn_DATA.variables['TIME'])
depthData = DEPTH[:]

#it is a lot more relevant for ADCP data to plot the good and probably good data only (flags 1 and 2).
qcLevel = []
qcLevel.append(1)
qcLevel.append(2)
qcIndex = ( anmn_DATA.variables['UCUR_quality_control'][:] == qcLevel[0]) |
( anmn_DATA.variables['UCUR_quality_control'][:] == qcLevel[1] )

# get the flag meaning values to add it later in the figure title
flag_meanings = (anmn_DATA.variables['UCUR_quality_control'].flag_meanings).split()

uCurrentData =  anmn_DATA.variables.UCUR.data;
uCurrentData [~qcIndex] = anmn_DATA.variables['UCUR']._FillValue
# we modify the mask in order to change the boolean, since some previous non Fillvalue data are now Fillvalue
uCurrentData = ma.masked_values(uCurrentData, anmn_DATA.variables['UCUR']._FillValue )

# creation of a observation/profile variable  because pcolor can't handle a time object in the x axis
sizer = ones((1,len(depthData)),'float')
profIndex = array(range(len(timeData)))
profIndex = profIndex.reshape(len(timeData),1)
prof_2D =  profIndex * sizer

# we create a matrix of similar size to be used afterwards with pcolor
[depthData_mesh,prof_2D_mesh] = meshgrid(depthData,profIndex)


import matplotlib.colors as mcolors
# creation of a blue and red colormap centered in white
levs = range(64)
assert len(levs) % 2 == 0, 'N levels must be even.'
```

```
cmap = mcolors.LinearSegmentedColormap.from_list(name='red_white_blue',
                                                 colors =[(0, 0, 1),
                                                          (1, 1., 1),
                                                          (1, 0, 0)],
                                                 N=len(levs)-1,
                                                 )

# plot adcp
figure1 =figure( figsize=(13, 18), dpi=80, facecolor='w', edgecolor='k')
ax1 = subplot(211)

pcolor(prof_2D_mesh , depthData_mesh , uCurrentData[:,:,0,0],cmap=cmap)
clim(UCUR.valid_min, UCUR.valid_max)
cbar = colorbar()
cbar.ax.set_ylabel(UCUR.long_name + ' in ' + UCUR.units)

title(metadata['title'] + '\nplot of ' + flag_meanings[qcLevel[0]] + ' and ' + flag_meanings[qcLevel[1]] + ' only')
xlabel('Profile Index')
ylabel(DEPTH.long_name +' in ' + DEPTH.units)

# plot profile index with time
ax2 = subplot(212)
plot(timeData,profIndex)
ylabel('Profile Index')
xlabel(anmn_DATA.variables['TIME'].long_name +' in DD/MM/YY')

from matplotlib.dates import MONTHLY, DateFormatter, rrulewrapper, RRuleLocator
rule = rrulewrapper(MONTHLY, bymonthday=1, interval=1)
formatter = DateFormatter('%d/%m/%y')
loc = RRuleLocator(rule)

ax2.xaxis.set_major_locator(loc)
ax2.xaxis.set_major_formatter(formatter)
labels = ax2.get_xticklabels()
setp(labels, rotation=30, fontsize=10)

plt.show()
```

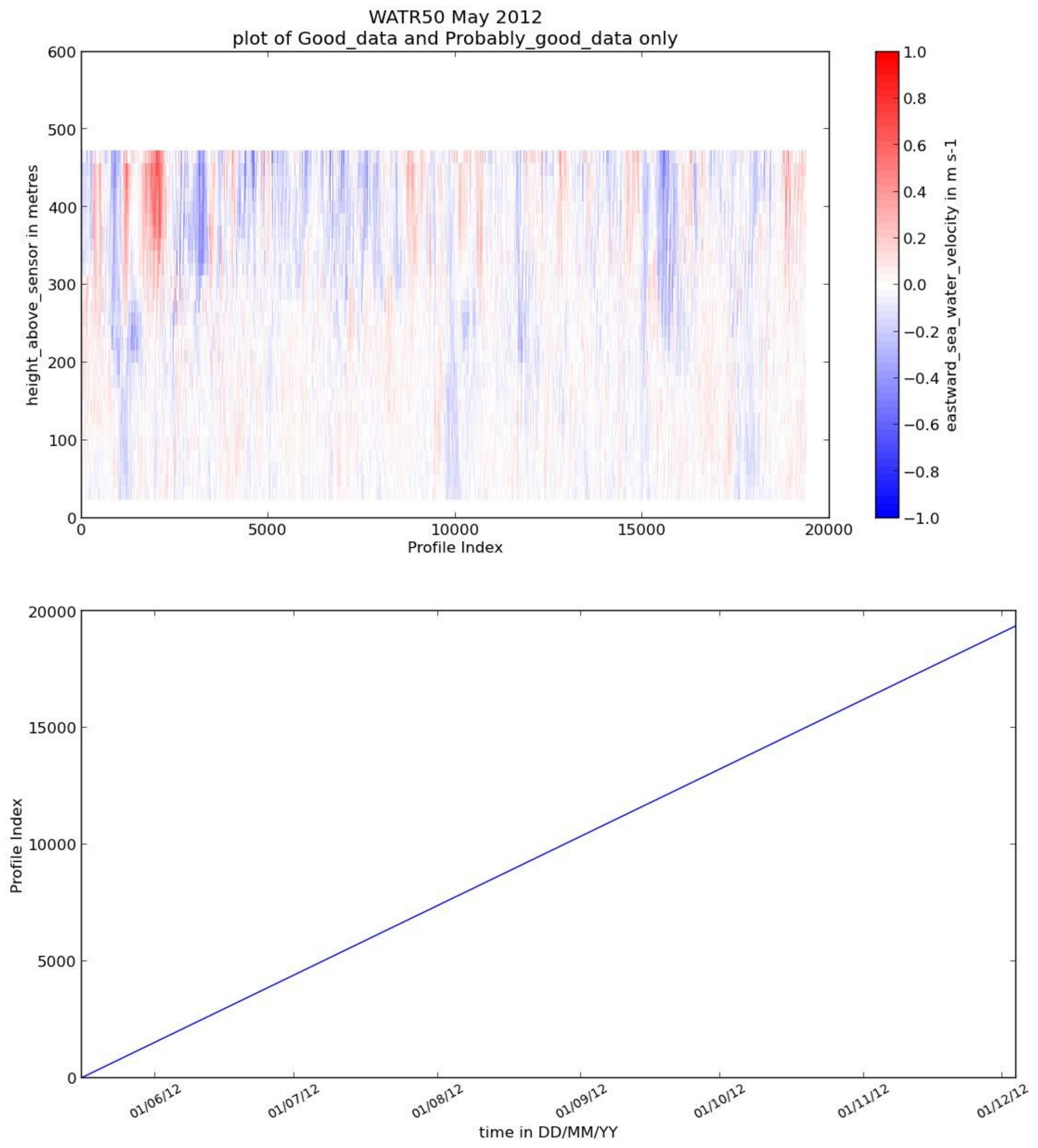


Illustration 6: Example of a Sea Water Velocity plot from ADCP data

Variables to modify :

- anmn_URL : the opendap url of the chosen file. (The example URL used here may not work if the file has been replaced by a newer version. A currently available file can be selected as described in section 1.2 above.)
- qcLevel : quality control value (varies from 0 to 9)

3.6 AUV – Autonomous Underwater Vehicle - non QC'd data

The IMOS Autonomous Underwater Vehicle (AUV) Facility operates an ocean going AUV called Sirius capable of undertaking high resolution, geo-referenced survey work.

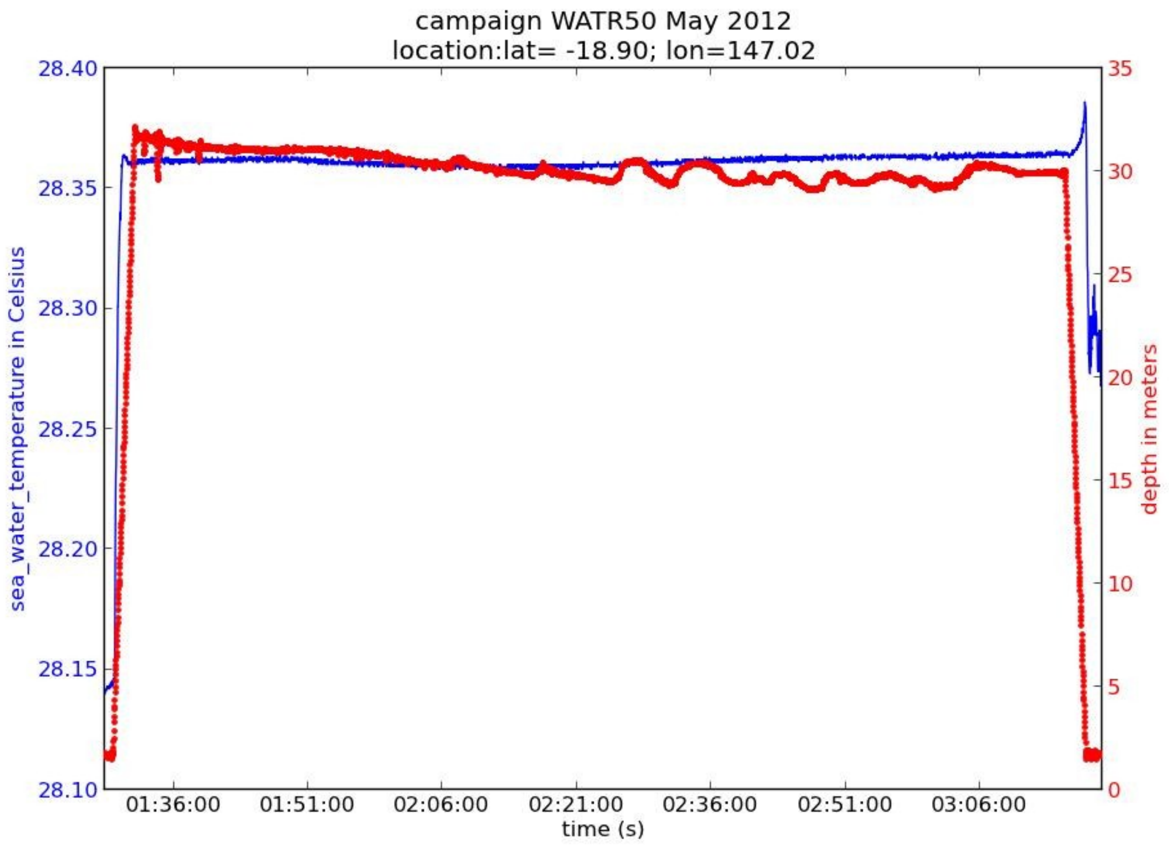
NetCDF files can be found at :
http://thredds.aodn.org.au/thredds/catalog/IMOS/AUV/catalog.html

In the example below, the netCDF4 module is used to extract depth, temperature, and time data and then produce a multiple time-series plot showing the variation of water temperature with depth and time during the robot's dive.

```
from netCDF4 import Dataset
from datetime import datetime, timedelta
from pylab import *
import numpy
import matplotlib.pyplot as plt
from imosNetCDF import *

############# AUV
auv_URL =
'http://thredds.aodn.org.au/thredds/dodsC/IMOS/eMII/demos/AUV/GBR201102/r20110301_012810_station1195_09_tr
ansect/hydro_netcdf/IMOS_AUV_ST_20110301T012815Z_SIRIUS_FV00.nc'
auv_DATA = Dataset(auv_URL)
metadata = getAttNC(auv_DATA)

tempData = auv_DATA.variables['TEMP']
timeData = convertTime(auv_DATA.variables['TIME'])
depthData = auv_DATA.variables['DEPTH']

averageLat = auv_DATA.variables['LATITUDE'][:].mean()
averageLon = auv_DATA.variables['LONGITUDE'][:].mean()

figure1 = figure( figsize=(10, 7), dpi=80, facecolor='w', edgecolor='k')

ax1 = figure1.add_subplot(111)
ax1.plot(timeData,tempData[:], 'b-')
ax1.set_xlabel('time (s)')
# Make the y-axis label and tick labels match the line color.
ax1.set_ylabel(tempData.standard_name + ' in ' + tempData.units, color='b')
for tl in ax1.get_yticklabels():
    tl.set_color('b')


ax2 = ax1.twinx()
ax2.plot(timeData,depthData[:], 'r.')
ax2.set_ylabel(depthData.standard_name + ' in ' + depthData.units, color='r')
for tl in ax2.get_yticklabels():
    tl.set_color('r')


xlabel(auv_DATA.variables['TIME'].standard_name)

title('campaign ' + metadata['title'] +  '\nlocation:lat= ' + "%0.2f" % averageLat + '; lon='  + "%0.2f" % averageLon )
plt.show()
```

Variables to modify :

- auv_URL : the opendap url of the chosen file


campaign WATR50 May 2012
location:lat= -18.90; lon=147.02
sea_water_temperature in Celsius
depth in meters
time (s)
28.40
28.35
28.30
28.25
28.20
28.15
28.10
35
30
25
20
15
10
5
0
01:36:00
01:51:00
02:06:00
02:21:00
02:36:00
02:51:00
03:06:00

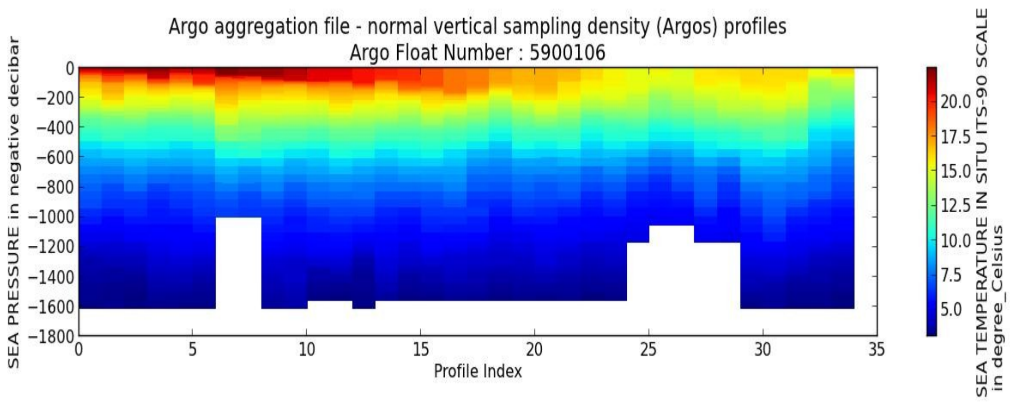
3.7 Argo – Argo Floats Program

Argo floats have revolutionised our understanding of the broad scale structure of the oceans to 2000 m depth. In the past 10 years more high resolution hydrographic profiles have been provided by Argo floats then from the rest of the observing system put together. Each Argo float is identified by a unique identification number called a WMO ID.

NetCDF files can be found at :
http://thredds.aodn.org.au/thredds/catalog/IMOS/Argo/aggregated_datasets/catalog.html

In the examples below, we demonstrate how to use the netCDF4 module to plot Argo data from an aggregated file (One file per year per basin : Atlantic, Indian, Pacific North, Pacific South).

```
from netCDF4 import Dataset
from datetime import datetime, timedelta
from pylab import *
import numpy
import matplotlib.pyplot as plt
from imosNetCDF import *

############# Argo
argo_URL =
'http://thredds.aodn.org.au/thredds/dodsC/IMOS/eMII/demos/Argo/aggregated_datasets/south_pacific/IMOS_Argo_TPS
-20020101T000000_FV01_yearly-aggregation-South_Pacific_C-20121102T220000Z.nc'
argo_DATA = Dataset(argo_URL)
metadata = getAttNC(argo_DATA)

nProfData = len(argo_DATA.dimensions['N_PROF'])  #Number of profiles contained in the file.
nLevelData = len(argo_DATA.dimensions['N_LEVELS'])  #Maximum number of pressure levels contained in a profile.

# we list all the argo floats number in the variable 'argoFloatNumber' and
#chose one value
argoFloatNumber = unique(argo_DATA.variables['PLATFORM_NUMBER'][:])

argoFloatNumberChosen = 5900106 # we randomly chose one float number
# we load the data for this float

argoFloatProfilesIndexes = argo_DATA.variables['PLATFORM_NUMBER'][:] == argoFloatNumberChosen
tempData = argo_DATA.variables['TEMP_ADJUSTED'][argoFloatProfilesIndexes]
psalData =argo_DATA.variables['PSAL_ADJUSTED'][argoFloatProfilesIndexes]
presData =argo_DATA.variables['PRES_ADJUSTED'][argoFloatProfilesIndexes]
latProfile = argo_DATA.variables['LATITUDE'][argoFloatProfilesIndexes]
lonProfile = argo_DATA.variables['LONGITUDE'][argoFloatProfilesIndexes]
timeProfile = convertTime(argo_DATA.variables['JULD'])[argoFloatProfilesIndexes]

# creation of a profile variable array
nProfForFloat = sum(argoFloatProfilesIndexes == True)
sizer = ones((1,nLevelData),'float')
profIndex = array(range(nProfForFloat))
profIndex = profIndex.reshape(nProfForFloat,1)
prof_2D =  profIndex * sizer


figure1 = figure(num=None, figsize=(15, 10), dpi=80, facecolor='w', edgecolor='k')
subplot(311)
pcolor(prof_2D, -presData, tempData)
cbar = colorbar()
cbar.ax.set_ylabel(argo_DATA.variables['TEMP_ADJUSTED'].long_name + '\n in ' +
argo_DATA.variables['TEMP_ADJUSTED'].units)
xlabel('Profile Index')
ylabel(argo_DATA.variables['PRES_ADJUSTED'].long_name + ' in negative ' +
argo_DATA.variables['PRES_ADJUSTED'].units)

title(metadata['description'] + '\nArgo Float Number : ' + "%0.0f" % argoFloatNumberChosen )


from matplotlib.dates import MONTHLY, DateFormatter, rrulewrapper, RRuleLocator
rule = rrulewrapper(MONTHLY, bymonthday=1, interval=1)
formatter = DateFormatter('%d/%m/%y')
loc = RRuleLocator(rule)

#plot the LON timeseries
ax3 = subplot(234)
plot(timeProfile,lonProfile)
ax3.xaxis.set_major_locator(loc)
ax3.xaxis.set_major_formatter(formatter)
labels = ax3.get_xticklabels()
setp(labels, rotation=30, fontsize=10)
xlabel(argo_DATA.variables['JULD'].long_name  + ' in ' +  'dd/mm/yy' )
ylabel(argo_DATA.variables['LONGITUDE'].long_name + ' in ' + argo_DATA.variables['LONGITUDE'].units)

#plot the LAT timeseries
ax4 = subplot(235)
plot(timeProfile,latProfile)
ax4.xaxis.set_major_locator(loc)
ax4.xaxis.set_major_formatter(formatter)
labels = ax4.get_xticklabels()
setp(labels, rotation=30, fontsize=10)
xlabel(argo_DATA.variables['JULD'].long_name  + ' in ' +  'dd/mm/yy' )
ylabel(argo_DATA.variables['LATITUDE'].long_name  + ' in ' +  argo_DATA.variables['LATITUDE'].units)

#plot the profile index with time values
ax5 = subplot(236)
plot(timeProfile,profIndex)
ax5.xaxis.set_major_locator(loc)
ax5.xaxis.set_major_formatter(formatter)
labels = ax5.get_xticklabels()
setp(labels, rotation=30, fontsize=10)
xlabel(argo_DATA.variables['JULD'].long_name  + ' in ' +  'dd/mm/yy' )
ylabel('Profile Index')


#plot of a single profile
profileToPlot = 1# this is arbitrary. We can plot all profiles from 1 to nProfiles, modify profileToPlot if desired
figure2 = figure(num=None, figsize=(7, 10), dpi=80, facecolor='w', edgecolor='k')
plot (tempData[profileToPlot,:],-presData[profileToPlot,:])
title(metadata['description'] + '\nlocation ' + "%0.2f" % latProfile[profileToPlot] + '/' + "%0.2f" %
lonProfile[profileToPlot] + '\n' + timeProfile[profileToPlot].strftime('%d/%m/%Y'))
xlabel(argo_DATA.variables['TEMP_ADJUSTED'].long_name +  ' in ' + argo_DATA.variables['TEMP_ADJUSTED'].units)
ylabel(argo_DATA.variables['PRES_ADJUSTED'].long_name +  ' in negative ' +
argo_DATA.variables['PRES_ADJUSTED'].units)

plt.show()
```



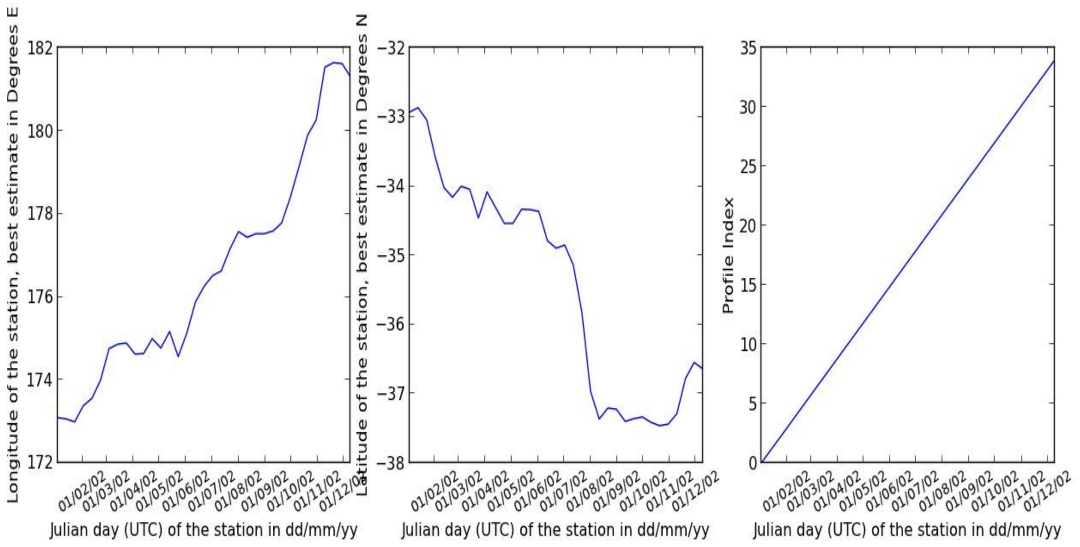


Illustration 7: Example of a Sea Water Temperature Time-series profile from an Argo float with its location over time

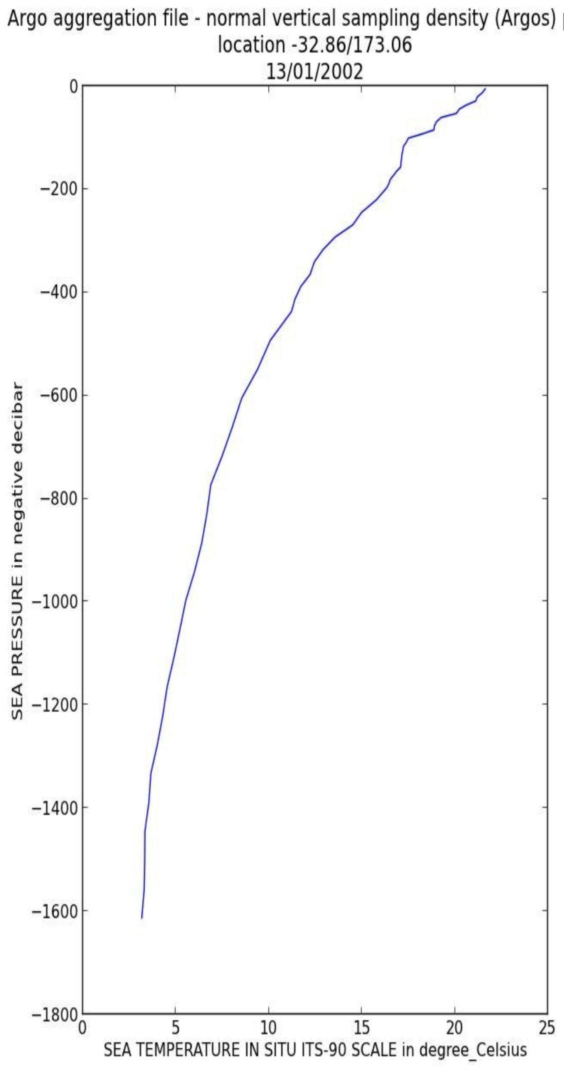


Illustration 8: Example of a Sea Water Temperature profile from an Argo float

3.8 FAIMMS – Wireless Sensor Networks - QC'd good data

The IMOS Facility for Intelligent Monitoring of Marine Systems is a sensor network established in the Great Barrier Reef off the coast of Queensland, Australia. A 'sensor network' is an array of small, wirelessly interconnected sensors that collectively stream sensor data to a central data aggregation point. Sensor networks can be used to provide spatially dense bio-physical measurements in real-time.

NetCDF files can be found at :
http://thredds.aodn.org.au/thredds/catalog/IMOS/FAIMMS/catalog.html

In the example below, we demonstrate how to use the netCDF4 module to plot a temperature time-series. Only data points which have a flag value equal to 1 are used (which means 'good data', please refers to IMOS NetCDF User Manual for a description of the Quality Control, available at http://imos.org.au/facility_manuals.html).

```python
from netCDF4 import Dataset
from datetime import datetime, timedelta
from pylab import *
import numpy
import matplotlib.pyplot as plt
from imosNetCDF import *

############# FAIMMS
FAIMMS_URL = 
'http://thredds.aodn.org.au/thredds/dodsC/IMOS/eMII/demos/FAIMMS/Myrmidon_Reef/Sensor_Float_1/water_temperature/sea_water_temperature@5.0m_channel_114/2012/QAQC/IMOS_FAIMMS_T_20121201T000000Z_FV01_END-20130101T000000Z_C-20130426T102459Z.nc'
faimms_DATA = Dataset(FAIMMS_URL)
metadata = getAttNC(faimms_DATA)

qcLevel = 1 # only good data are being used

TEMP = faimms_DATA.variables['TEMP'][:]
TEMP_qcFlag = faimms_DATA.variables['TEMP_quality_control']
index_qcLevel = where( TEMP_qcFlag[:,0,0] == qcLevel)


timeData = convertTime(faimms_DATA.variables['TIME'])[index_qcLevel]
tempData = TEMP[index_qcLevel[0][:],0,0]

figure1 = figure(num=None, figsize=(15, 10), dpi=80, facecolor='w', edgecolor='k')
ax1 = subplot(111)
plot (timeData,tempData[:,])

title(metadata['title'] + '\n' + "%0.2f" % faimms_DATA.variables['TEMP'].sensor_depth + ' m depth' + '\nlocation:lat=' 
+ "%0.2f" % faimms_DATA.variables['LATITUDE'][:] + '; lon=' + "%0.2f" % faimms_DATA.variables['LONGITUDE'][:] )
xlabel( faimms_DATA.variables['TIME'].long_name)
ylabel( faimms_DATA.variables['TEMP'].standard_name +' in ' + faimms_DATA.variables['TEMP'].units)

from matplotlib.dates import DAILY, DateFormatter, rrulewrapper, RRuleLocator
rule = rrulewrapper(DAILY,  interval=1)
formatter = DateFormatter('%d/%m/%y')
loc = RRuleLocator(rule)
ax1.xaxis.set_major_locator(loc)
ax1.xaxis.set_major_formatter(formatter)
labels = ax1.get_xticklabels()
setp(labels, rotation=30, fontsize=10)

plt.show()
```

Variables to modify :

- faimms_URL : the opendap url of the chosen file
- qcLevel : quality control value (varies from 0 to 9)

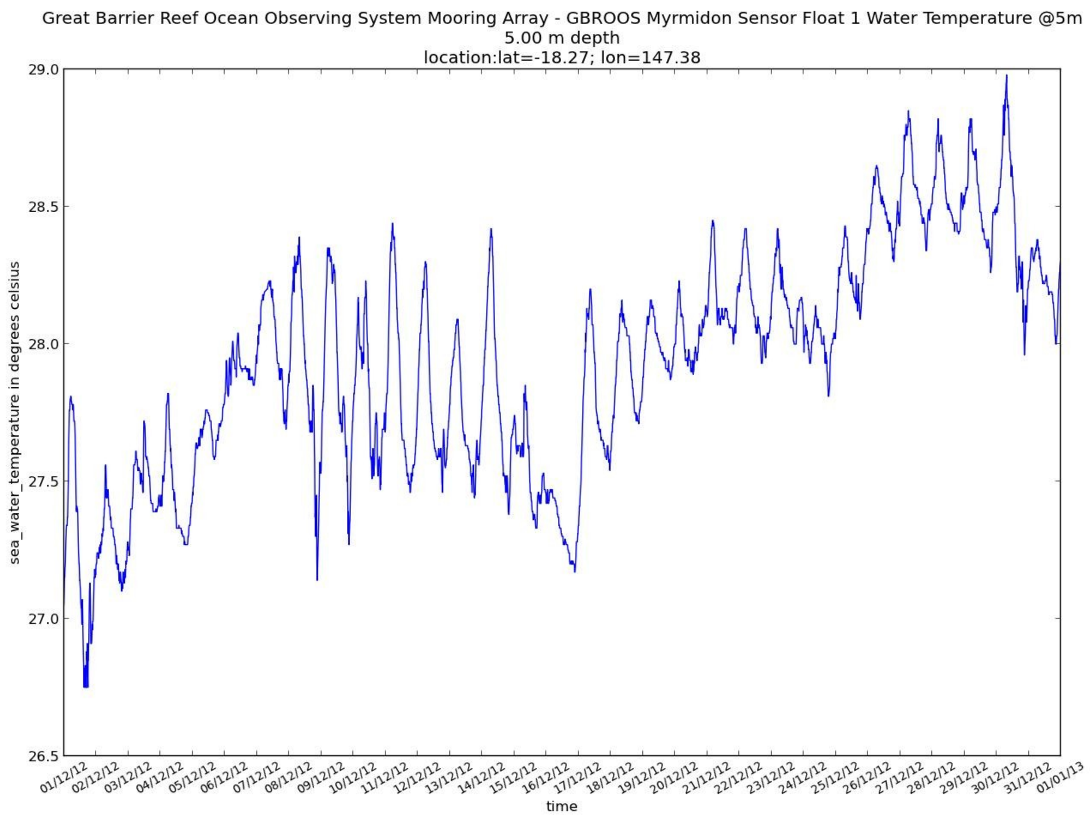


Illustration 9: Example of a Sea Water Temperature at 5m depth on the Great Barrier Reaf from FAIMMS data

3.9 SOOP – Ship Of Opportunities

3.9.1 XBT - expandable bathythermographs - QC'd data

IMOS Ship of Opportunity Underway Expandable Bathythermographs (XBT) group is a research and data collection project working within the IMOS Ship of Opportunity Multi-Disciplinary Underway Network sub-facility.

NetCDF files can be found at :
http://thredds.aodn.org.au/thredds/catalog/IMOS/SOOP/SOOP-XBT/catalog.html

In the example below, we demonstrate how to use the ncParse function in order to plot a XBT temperature profile.

```
from netCDF4 import Dataset
from datetime import datetime, timedelta
from pylab import *
import numpy
import matplotlib.pyplot as plt
from imosNetCDF import *

#### XBT
xbt_URL = 'http://thredds.aodn.org.au/thredds/dodsC/IMOS/eMII/demos/SOOP/SOOP-
XBT/aggregated_datasets/line_and_year/IX1/IMOS_SOOP-XBT_T_20040131T195300Z_IX1_FV01_END-
20041221T214400Z.nc'
xbt_DATA = Dataset(xbt_URL)
metadata = getAttNC(xbt_DATA)

qcFlag = 4 # flag value to eliminate (bad data)

maxSample = len(xbt_DATA.variables['MAXZ'][:]) # 'maximum_number_of_samples_in_vertical_profile'
nProfiles = len(xbt_DATA.variables['INSTANCE'][:]) # number of profiles

import string, re
## we look for all the profiles of a similar cruise
cruiseData = xbt_DATA.variables['cruise_ID'][:]
cruiseID = []
for iiCruise in range( len(cruiseData)) :
    cruiseID.append ( string.join(cruiseData[iiCruise,:]).replace(" ", ""))

uniqueCruiseIds = unique(cruiseID)
cruiseToPlot = uniqueCruiseIds[5] #  'tb408504' , this is arbitrary. This value can be moified to plot the cruise of
choice
indexCruiseToPlot = [item for item in range(len(cruiseID)) if cruiseID[item] == cruiseToPlot]

TEMP = xbt_DATA.variables['TEMP']
DEPTH = xbt_DATA.variables['DEPTH']
TIME = xbt_DATA.variables['TIME']

# we load the data for each cruise
timeCruise =  convertTime(TIME)[indexCruiseToPlot]
latCruise =  xbt_DATA.variables['LATITUDE'][indexCruiseToPlot]
lonCruise =  xbt_DATA.variables['LONGITUDE'][indexCruiseToPlot]

# we load only the data which does not have a quality control value equal to qcFlag (see above)
indexGoodData = xbt_DATA.variables['TEMP_quality_control'][:,indexCruiseToPlot] != qcFlag
tempCruise =  TEMP[:,indexCruiseToPlot]
depthCruise = DEPTH[:,indexCruiseToPlot]


import numpy.ma as ma
# we modify the values which we don't want to plot to replace them with the Fillvalue
```

```
tempCruise[~indexGoodData] = xbt_DATA.variables['TEMP']._FillValue
depthCruise[~indexGoodData] = xbt_DATA.variables['DEPTH']._FillValue
# we modify the mask in order to change the boolean, since some previous non Fillvalue data are now Fillvalue
tempCruise = ma.masked_values(tempCruise, xbt_DATA.variables['TEMP']._FillValue)
depthCruise = ma.masked_values(depthCruise, xbt_DATA.variables['DEPTH']._FillValue)

# creation of a profile array to use it with pcolor. same dimension of temp and depth
[nline, ncol] = shape(tempCruise)
sizer = ones((nline,1),'float')
profileIndex = range(ncol)
prof_2D =  sizer * profileIndex

##### creation of the plots
figure1 = figure(num=None, figsize=(15, 10), dpi=80, facecolor='w', edgecolor='k')
# Profile timeseries
subplot(311)
pcolor(prof_2D, -depthCruise, tempCruise)
cbar = colorbar()
cbar.ax.set_ylabel(TEMP.long_name + ' in ' + TEMP.units)
title(metadata['title'] + '\n Cruise  ' + cruiseToPlot + '-' + metadata['XBT_line_description'])
xlabel('Profile Index')
ylabel(DEPTH.long_name + ' in negative ' + DEPTH.units)

#plot the LON timesexbt_DATAries
ax3 = subplot(234)
plot(profileIndex,lonCruise)
xlabel('Profile Index')
ylabel(xbt_DATA.variables['LONGITUDE'].long_name + ' in ' + xbt_DATA.variables['LONGITUDE'].units)

#plot the LAT timeseries
ax4 = subplot(235)
plot(profileIndex,latCruise)
xlabel('Profile Index')
ylabel(xbt_DATA.variables['LATITUDE'].long_name  + ' in ' +  xbt_DATA.variables['LATITUDE'].units)

#plot the profile index with time values
# create the time label ticks
from matplotlib.dates import MONTHLY, DateFormatter, rrulewrapper, RRuleLocator
rule = rrulewrapper(MONTHLY, bymonthday=1, interval=1)
formatter = DateFormatter('%d/%m/%y')
loc = RRuleLocator(rule)

ax5 = subplot(236)
plot(timeCruise,profileIndex)
ax5.xaxis.set_major_locator(loc)
ax5.xaxis.set_major_formatter(formatter)
labels = ax5.get_xticklabels()
setp(labels, rotation=30, fontsize=10)
xlabel(TIME.long_name  + ' in ' +  'dd/mm/yy' )
ylabel('Profile Index')

# plot of a single profile of this cruise
profileToPlot = 1  # this is arbitrary. We can plot all profiles from 1 to ncol, modify profileToPlot if desired

figure2 = figure(num=None, figsize=(13, 9.2), dpi=80, facecolor='w', edgecolor='k')
plot (tempCruise[:,profileToPlot],-depthCruise[:,profileToPlot])

xlabel(TEMP.long_name +' in ' +TEMP.units)
ylabel(DEPTH.long_name + ' in negative '  + DEPTH.units)

title(metadata['title'] +   '\n Cruise  ' + cruiseToPlot + '-' + metadata['XBT_line_description']+ '\nlocation ' + "%0.2f"
% latCruise[profileToPlot] + '/' + "%0.2f" % lonCruise[profileToPlot] + '\n' + timeCruise[profileToPlot].strftime('%d/
%m/%Y'))

plt.show()
```

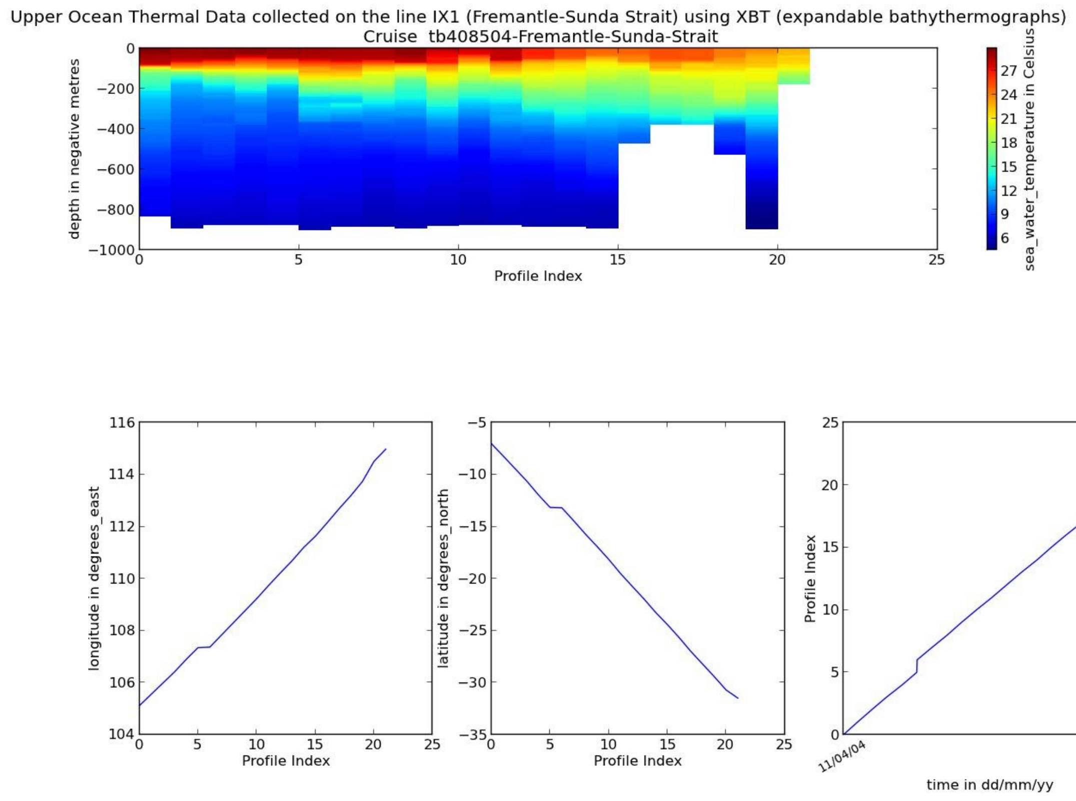


Illustration 10: Example of Sea Water Temperature Time-series Profile from XBT data with the profiles' location

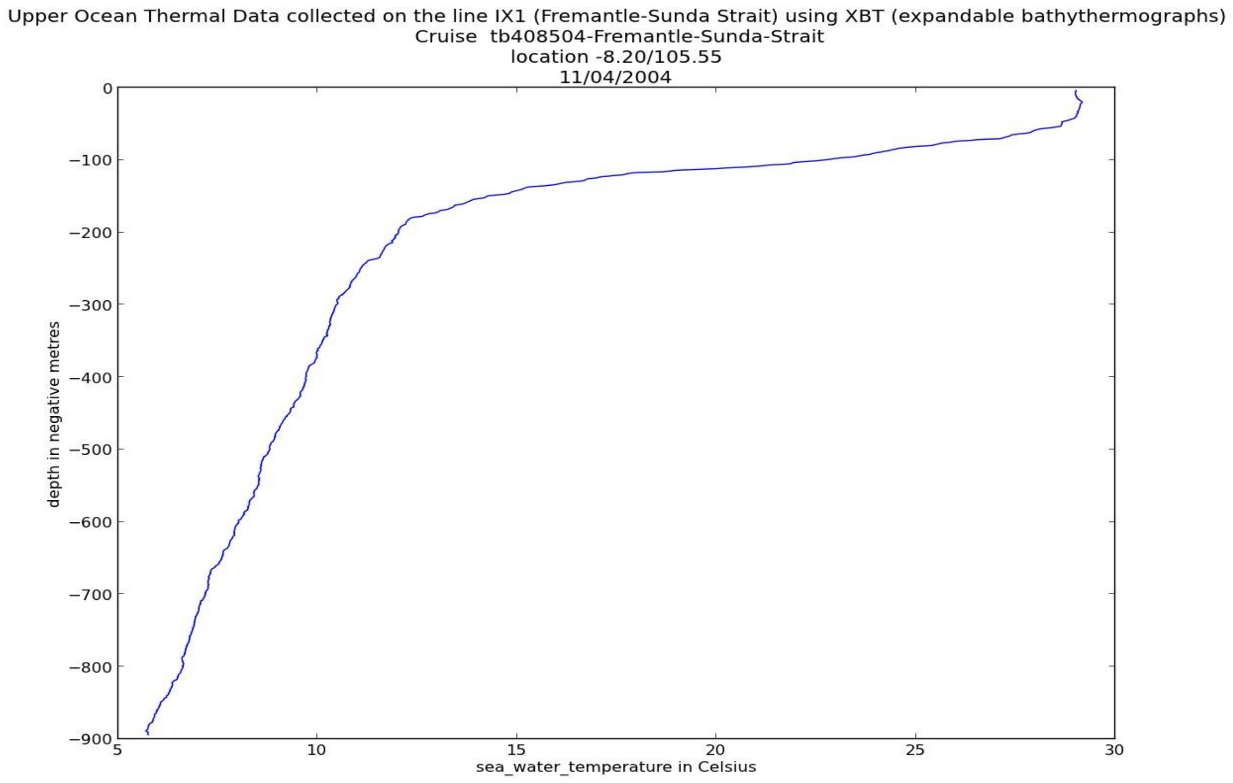


Illustration 11: Example of Sea Water Temperature Profile from XBT data

3.10 SRS – Satellite Remote Sensing

3.10.1 Bio-Optical database – Pigment data

The bio-optical data base underpins the assessment of ocean colour products in the Australian region (e.g. chlorophyll a concentrations, phytoplankton species composition and primary production).

NetCDF files can be found at :
http://thredds.aodn.org.au/thredds/catalog/IMOS/SRS/BioOptical/catalog.html

In the example below, we demonstrate how to use the netCDF4 module to plot a Chlorophyll-a profile (High Performance Liquid Chromatography of pigments in discrete sea-water samples)

```
from netCDF4 import Dataset
from datetime import datetime, timedelta
from pylab import *
import numpy
import matplotlib.pyplot as plt
from imosNetCDF import *

############# BioOptic pigment
srs_pigment_URL = 'http://thredds.aodn.org.au/thredds/dodsC/IMOS/eMII/demos/SRS/BioOptical/1997_cruise-FR1097/pigment/IMOS_SRS-OC-BODBAW_X_19971201T052600Z_FR1097-pigment_END-19971207T220700Z_C-20121129T120000Z.nc'
srs_pigment = Dataset(srs_pigment_URL)
metadata = getAttNC(srs_pigment)

nProfiles = len(srs_pigment.dimensions['profile'])
# we choose the first profile
ProfileToPlot = 9 # this is arbitrary. We can plot all profiles from 0 to nProfiles
nObsProfile = srs_pigment.variables['rowSize'][ProfileToPlot] #number of observations for ProfileToPlot
timeProfile = convertTime(srs_pigment.variables['TIME'])[ProfileToPlot]
latProfile = srs_pigment.variables['LATITUDE'][ProfileToPlot]
lonProfile = srs_pigment.variables['LONGITUDE'][ProfileToPlot]

# we look for the observations indexes related to the choosen profile
indexObservationStart = sum( srs_pigment.variables['rowSize'][range(0,ProfileToPlot)])
indexObservationEnd = sum(srs_pigment.variables['rowSize'][range(0,ProfileToPlot+1)])
indexObservation = range(indexObservationStart,indexObservationEnd  )

cphl_aData = srs_pigment.variables['CPHL_a'][indexObservation] # for ProfileToPlot
depthData = srs_pigment.variables['DEPTH'][indexObservation]

figure1 = figure(num=None, figsize=(15, 10), dpi=80, facecolor='w', edgecolor='k')
plot (cphl_aData,depthData)

title(metadata['source'] +  timeProfile.strftime('%d/%m/%Y') + '\nlocation:lat=' + "%0.2f" % latProfile + '; lon=' + "%0.2f" %lonProfile )
xlabel(srs_pigment.variables['CPHL_a'].long_name + ' in ' + srs_pigment.variables['CPHL_a'].units)
ylabel( srs_pigment.variables['DEPTH'].long_name + ' in ' +  srs_pigment.variables['DEPTH'].units + ';positive ' + srs_pigment.variables['DEPTH'].positive )

plt.show()
```

Variables to modify :

- srs_URL : the opendap url of the chosen file
- ProfileToPlot : the profile number to plot

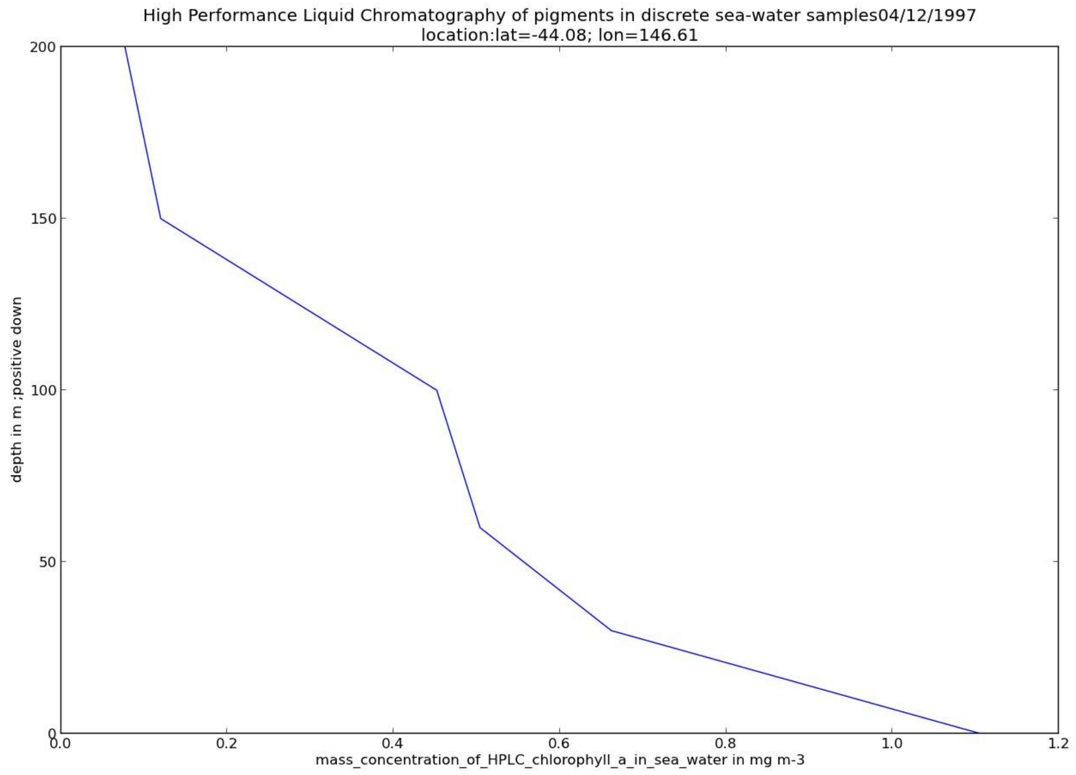


Illustration 12: Example of Pigment Data Profile from the BioOptical database dataset

3.10.2 Bio-Optical database – Absorption data

The bio-optical data base underpins the assessment of ocean colour products in the Australian region (e.g. chlorophyll a concentrations, phytoplankton species composition and primary production).

NetCDF files can be found at :
http://thredds.aodn.org.au/thredds/catalog/IMOS/SRS/BioOptical/catalog.html

In the example below, we demonstrate how to use the netCDF4 module to plot (1) the variation of Absorption coefficients of CDOM (gilvin) in discrete sea-water samples at different wavelengths and (2) the variation of absorption coefficients of CDOM at different wavelengths and different depths.

```
from netCDF4 import Dataset
from datetime import datetime, timedelta
from pylab import *
import numpy
import matplotlib.pyplot as plt
from imosNetCDF import *

############# BioOptic absorption
srs_absorption_URL = 'http://thredds.aodn.org.au/thredds/dodsC/IMOS/eMII/demos/SRS/BioOptical/1997_cruise-
FR1097/absorption/IMOS_SRS-OC-BODBAW_X_19971201T052600Z_FR1097-absorption-CDOM_END-
19971207T180500Z_C-20121129T130000Z.nc'
srs_absorption = Dataset(srs_absorption_URL)
metadata = getAttNC(srs_absorption)

nProfiles = len(srs_absorption.dimensions['profile'])
# we choose the first profile
ProfileToPlot = 9 # this is arbitrary. We can plot all profiles from 0 to nProfiles
nObsProfile = srs_absorption.variables['rowSize'][ProfileToPlot] #number of observations for ProfileToPlot
timeProfile = convertTime(srs_absorption.variables['TIME'])[ProfileToPlot]
latProfile = srs_absorption.variables['LATITUDE'][ProfileToPlot]
lonProfile = srs_absorption.variables['LONGITUDE'][ProfileToPlot]

# we look for the observations indexes related to the choosen profile
indexObservationStart = sum( srs_absorption.variables['rowSize'][range(0,ProfileToPlot)])
indexObservationEnd = sum(srs_absorption.variables['rowSize'][range(0,ProfileToPlot+1)])
indexObservation = range(indexObservationStart,indexObservationEnd  )

agData = srs_absorption.variables['ag'][indexObservation,:]
wavelengthData = srs_absorption.variables['wavelength']
depthData = srs_absorption.variables['DEPTH'][indexObservation]

#we create a matrix of similar size to be used afterwards with pcolor
[wavelengthData_mesh,depthData_mesh] = meshgrid(wavelengthData,depthData)

figure1 = figure(num=None, figsize=(15, 10), dpi=80, facecolor='w', edgecolor='k')
pcolor(wavelengthData_mesh , depthData_mesh , agData)
cbar = colorbar()
cbar.ax.set_ylabel(srs_absorption.variables['ag'].long_name + '\n in ' + srs_absorption.variables['ag'].units)


title(metadata['source'])
xlabel( srs_absorption.variables['wavelength'].long_name + ' in: ' + srs_absorption.variables['wavelength'].units)
ylabel(srs_absorption.variables['DEPTH'].long_name + ' in ' + srs_absorption.variables['DEPTH'].units + '; positive
'+srs_absorption.variables['DEPTH'].positive )


##
nDepth = len(depthData)
figure2 = figure(num=None, figsize=(15, 10), dpi=80, facecolor='w', edgecolor='k')
```

```
labels = []
for iplot in range(shape(agData)[0]):
    plot(wavelengthData[:],agData[iplot,:],'x')
    labels.append(r'Depth = %i m' % depthData[iplot])

plt.legend(labels,loc='upper right')

ylabel(srs_absorption.variables['ag'].long_name + ' in: ' + srs_absorption.variables['ag'].units)
xlabel( srs_absorption.variables['wavelength'].long_name + ' in: ' + srs_absorption.variables['wavelength'].units)

title(srs_absorption.variables['ag'].long_name + 'in units:' + srs_absorption.variables['ag'].units + '\nstation :' +
'\nlocation:lat=' + "%0.2f" % latProfile + '; lon=' + "%0.2f" %lonProfile  + timeProfile.strftime('%d/%m/%Y') )

plt.show()
```

Variables to modify :

- srs_URL : the opendap url of the chosen file
- ProfileToPlot : the profile number to plot

3.10.3 GHRSST – L3P mosaic

Please refer to the SRS product Help page : http://portalhelp.aodn.org.au/Portal2_help/?q=node/149

NetCDF files can be found at :
http://thredds.aodn.org.au/thredds/dodsC/IMOS/SRS/GHRSST-SSTsubskin/

In the example below, we demonstrate how to use the netCDF4 module to plot the Sea Surface Temperature from a gridded data product.

```
from netCDF4 import Dataset
from datetime import datetime, timedelta
from pylab import *
import numpy
import matplotlib.pyplot as plt
from imosNetCDF import *

######## GHRSST – L3P mosaic
srs_L3P_URL = 'http://thredds.aodn.org.au/thredds/dodsC/IMOS/eMII/demos/SRS/SRS-SST/L3P/2013/20130315-ABOM-
L3P_GHRSST-SSTsubskin-AVHRR_MOSAIC_01km-AO_DAAC-v01-fv01_0.nc'
srs_L3P_DATA = Dataset(srs_L3P_URL)
metadata = getAttNC(srs_L3P_DATA)

step = 20 # we take one point out of 'step'. Only to make it faster to plot
sst = srs_L3P_DATA.variables['sea_surface_temperature'][0,::step,::step]
lat =srs_L3P_DATA.variables['lat'][::step]
lon = srs_L3P_DATA.variables['lon'][::step]
[lon_mesh,lat_mesh] = meshgrid(lon,lat)  #we create a matrix of similar size to be used afterwards with pcolor

figure1 =  figure(num=None, figsize=(15, 10), dpi=80, facecolor='w', edgecolor='k')
pcolor(lon_mesh,lat_mesh ,sst)

title( metadata['title'] + '-' +  metadata['start_date'] )
xlabel(srs_L3P_DATA.variables['lon'].long_name +  ' in ' + srs_L3P_DATA.variables['lon'].units)
ylabel(srs_L3P_DATA.variables['lat'].long_name +  ' in ' + srs_L3P_DATA.variables['lat'].units)

cbar = colorbar()
cbar.ax.set_ylabel(srs_L3P_DATA.variables['sea_surface_temperature'].long_name + '\n in ' +
srs_L3P_DATA.variables['sea_surface_temperature'].units)
plt.show()
```

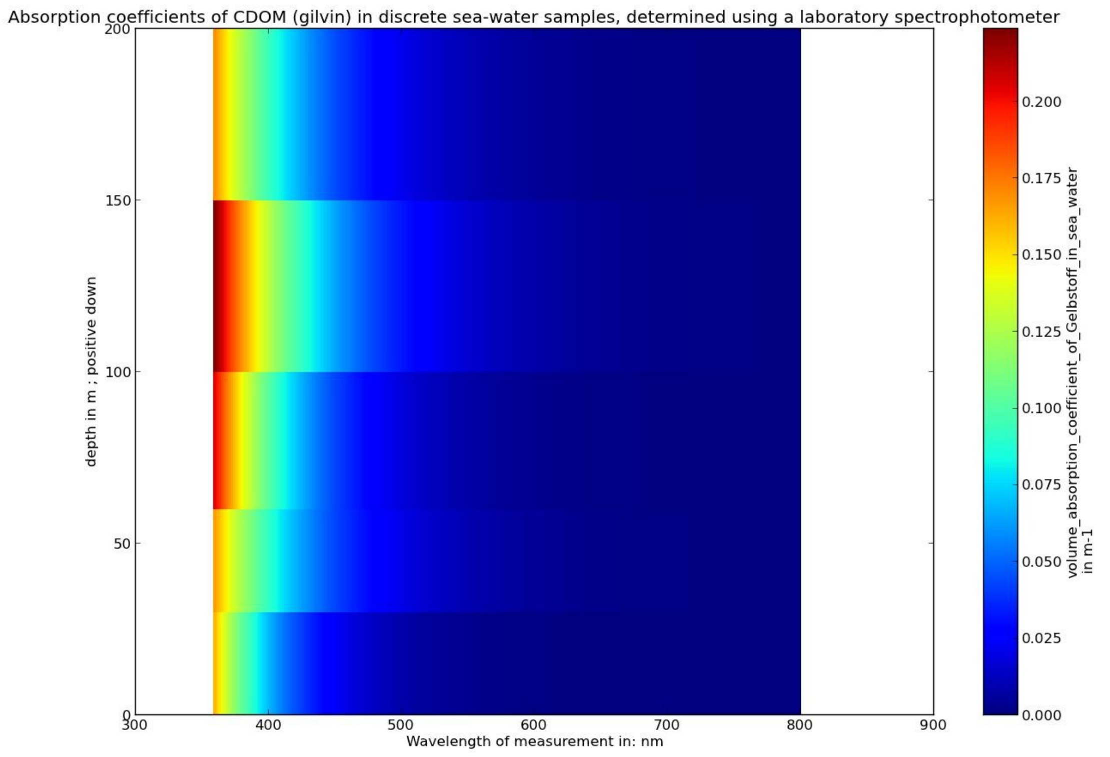


Illustration 13: Example of Absorption Data plot from the BioOptical database dataset

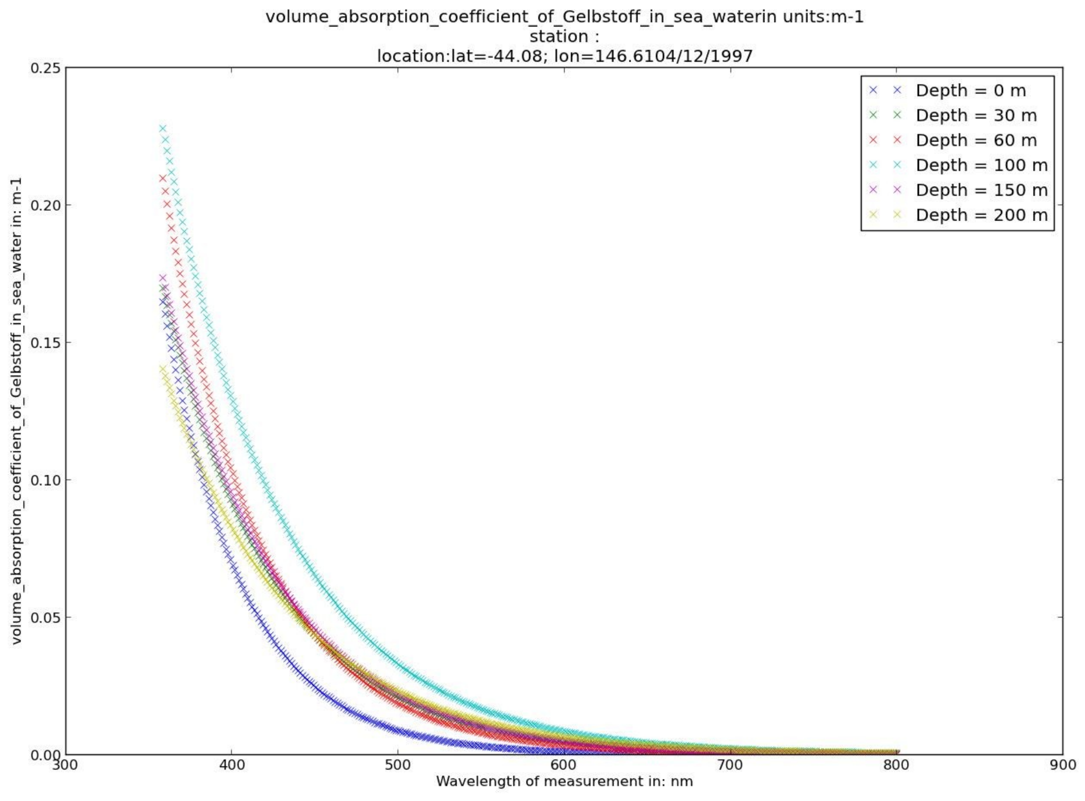


Illustration 14: Example of Absorption Data at different depth from the BioOptical database dataset

Variables to modify :

- srs_URL : the opendap url of the chosen file
- step : a number to lower the resolution. This helps to reduce memory issues.

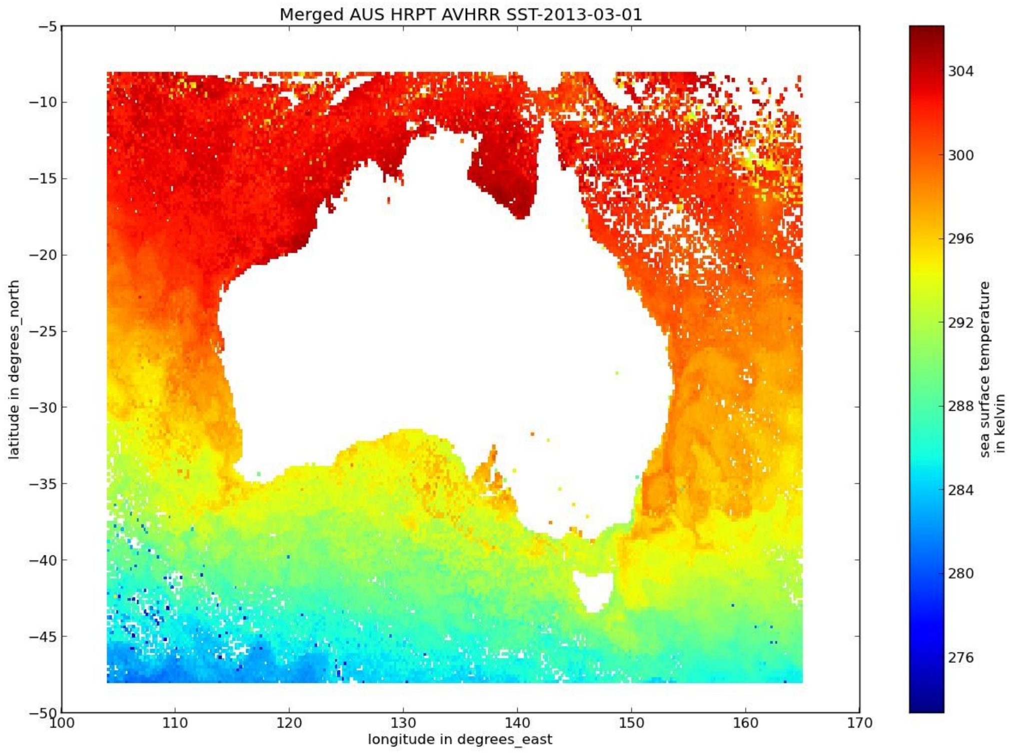

Merged AUS HRPT AVHRR SST-2013-03-01
latitude in degrees_north
longitude in degrees_east
sea surface temperature in kelvin
−5
−10
−15
−20
−25
−30
−35
−40
−45
−50
100
110
120
130
140
150
160
170
304
300
296
292
288
284
280
276

Illustration 15: Example of Sea Surface Temperature plot from a L3P product

3.10.4 GHRSST – L3S – multi swath, multi sensor, one day

Please refer to the SRS product Help page : http://portalhelp.aodn.org.au/Portal2_help/?q=node/149

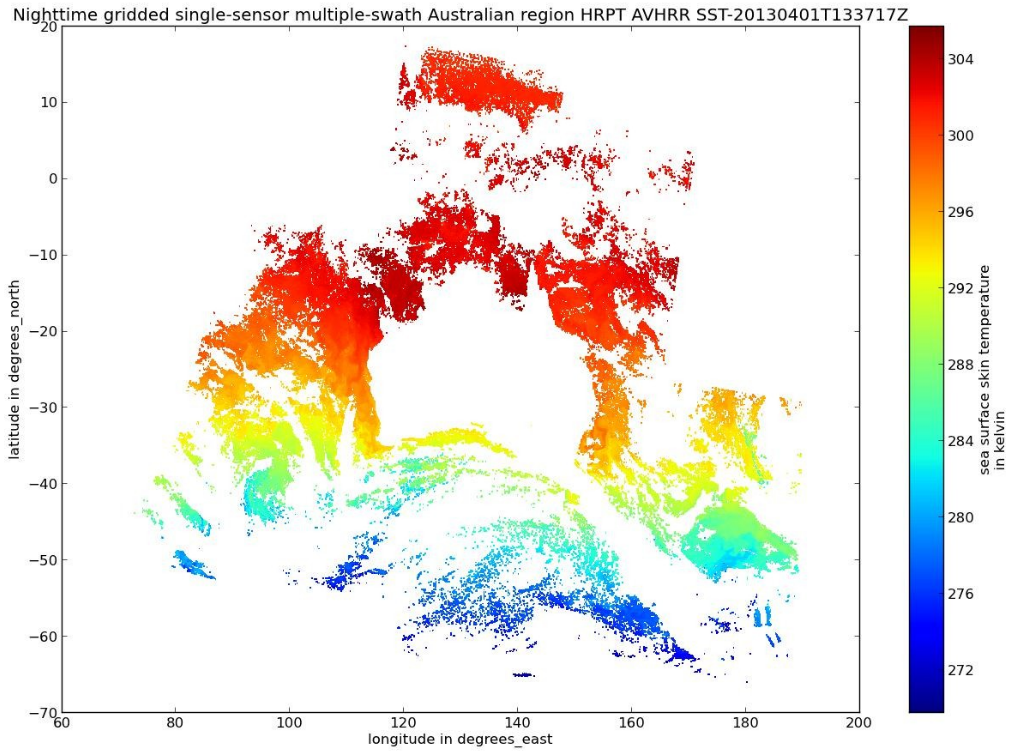
NetCDF files can be found at :
http://thredds.aodn.org.au/thredds/dodsC/IMOS/SRS/SRS-SST/L3S-01day/

In the example below, we demonstrate how to use the netCDF4 module to plot the Sea Surface Temperature from a gridded data product.

```
from netCDF4 import Dataset
from datetime import datetime, timedelta
from pylab import *
import numpy
import matplotlib.pyplot as plt
from imosNetCDF import *

######## GHRSST – L3S mosaic
srs_L3S_URL = 'http://thredds.aodn.org.au/thredds/dodsC/IMOS/eMII/demos/SRS/SRS-SST/L3S-
01day/L3S_1d_night/2013/20130401152000-ABOM-L3S_GHRSST-SSTskin-AVHRR_D-1d_night-v02.0-fv01.0.nc.gz'
srs_L3S_DATA = Dataset(srs_L3S_URL)
metadata = getAttNC(srs_L3S_DATA)


step = 5 # we take one point out of 'step'. Only to make it faster to plot
sst = srs_L3S_DATA.variables['sea_surface_temperature'][0,::step,::step]
lat =srs_L3S_DATA.variables['lat'][::step]
lon = srs_L3S_DATA.variables['lon'][::step]
# modify the longitude values which are across the 180th meridian
if sum(lon<0) > 0:
    lon[lon<0] = lon[lon<0]+360

[lon_mesh,lat_mesh] = meshgrid(lon,lat)#we create a matrix of similar size to be used afterwards with pcolor

figure1 =  figure(num=None, figsize=(15, 10), dpi=80, facecolor='w', edgecolor='k')
pcolor(lon_mesh,lat_mesh ,sst)
title( metadata['title'] + '-' +  metadata['start_time'] )
xlabel(srs_L3S_DATA.variables['lon'].long_name +  ' in ' + srs_L3S_DATA.variables['lon'].units)
ylabel(srs_L3S_DATA.variables['lat'].long_name +  ' in ' + srs_L3S_DATA.variables['lat'].units)

cbar = colorbar()
cbar.ax.set_ylabel(srs_L3S_DATA.variables['sea_surface_temperature'].long_name + '\n in ' +
srs_L3S_DATA.variables['sea_surface_temperature'].units)
plt.show()
```

Variables to modify :

- srs_URL : the opendap url of the chosen file
- step : a number to lower the resolution. This helps to reduce memory issues.



Nighttime gridded single-sensor multiple-swath Australian region HRPT AVHRR SST-20130401T133717Z
latitude in degrees_north
longitude in degrees_east
sea surface skin temperature in kelvin
20
10
0
−10
−20
−30
−40
−50
−60
−70
60
80
100
120
140
160
180
200
304
300
296
292
288
284
280
276
272

Illustration 16: Example of Sea Surface Temperature plot from a L3S product at night