SkewTplus Documentation

Author

May 13, 2021

Contents

The SkewTplus package provides tools to easily read atmospheric sounding data from different formats (University of Wyoming and ARM) and create SkewT sounding plots along with parcel diagnostics (CAPE,CIN,etc.).

This package is based on the SkewT Python package developed by Thomas Chubb (https://pypi.python.org/pypi/SkewT/)

The main difference with the original *SkewT package* is that the vertical soundings plots are handled by a special class (SkewT). The new *SkewT* class extends the base matplolib's Figure class with an interface similar to matplolib's pyplot. It also allows to create Skew-T type plots in a simple way. This new class allows a complete control over the Figure properties like multiple plots (normal axis and Skew-T axis).

In addition, the **thermodynamics** module was improved. All the intensive computations were migrated to Cython and parallelized.

The SkewT Python package was a cornerstone of this project. We are grateful to all its collaborators.

Technology builds on technology

CHAPTER 1

Documentation

The documentation is separated in two big branches. The *User Reference* and the *Developer Reference*. The user reference provides a quick overview of the most important features of the package. For more detailed and a comprehensive understanding of the package the reader must consult the *Developer Reference*.

1.1 User Reference

This reference guide is intended to go through the most important capabilities of SkewTplus package. For further details on the full contents of each module, see *Developer Reference*.

The User Reference has the following chapters

1.1.1 Fist-Steps

This chapter offers a quick overview of the main package capabilities: To read a sounding from a txt file and create a quick SkewT diagram.

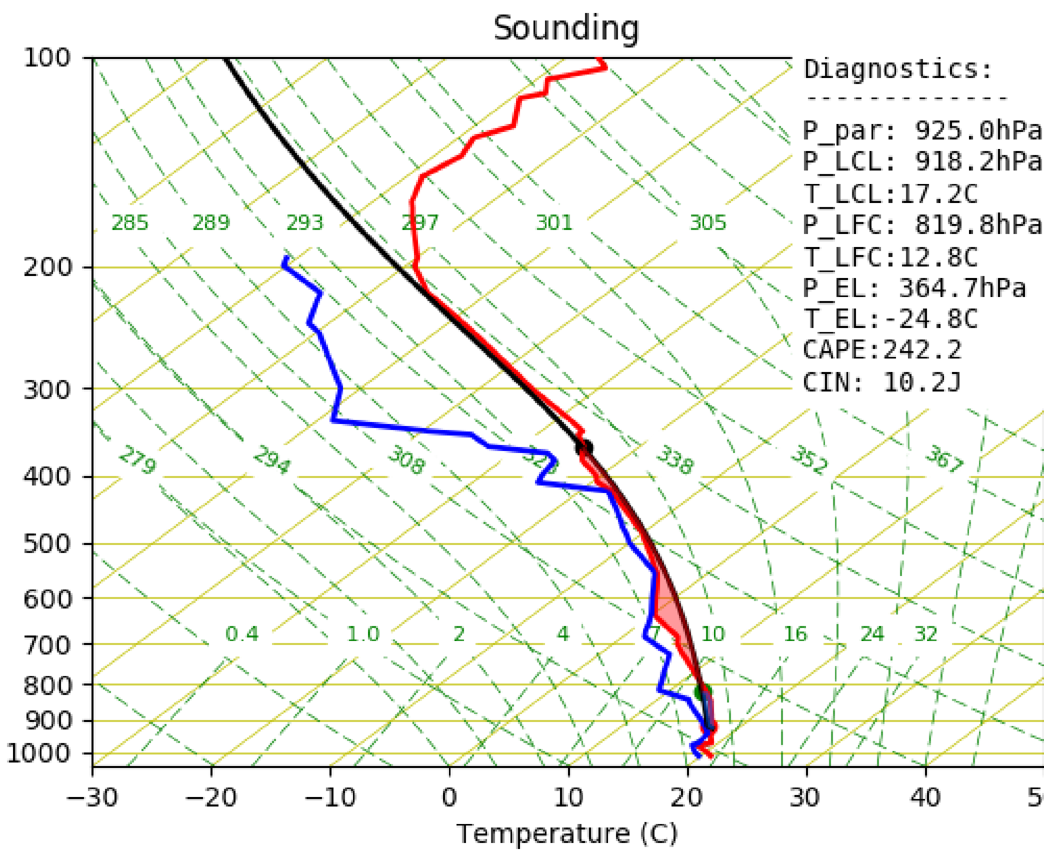
Fist steps using SkewTplus

From now on, it's assumed that the package is installed and the current working directory is the *examples* one, included in this package.

To read a sounding from a txt file and create a quick plot using the default parameters we only have to do:

```
from SkewTplus.sounding import Sounding

#Load the sounding data
mySounding = sounding("./exampleSounding.txt")
mySounding.quickPlot()
```

The resulting plot will look like this:



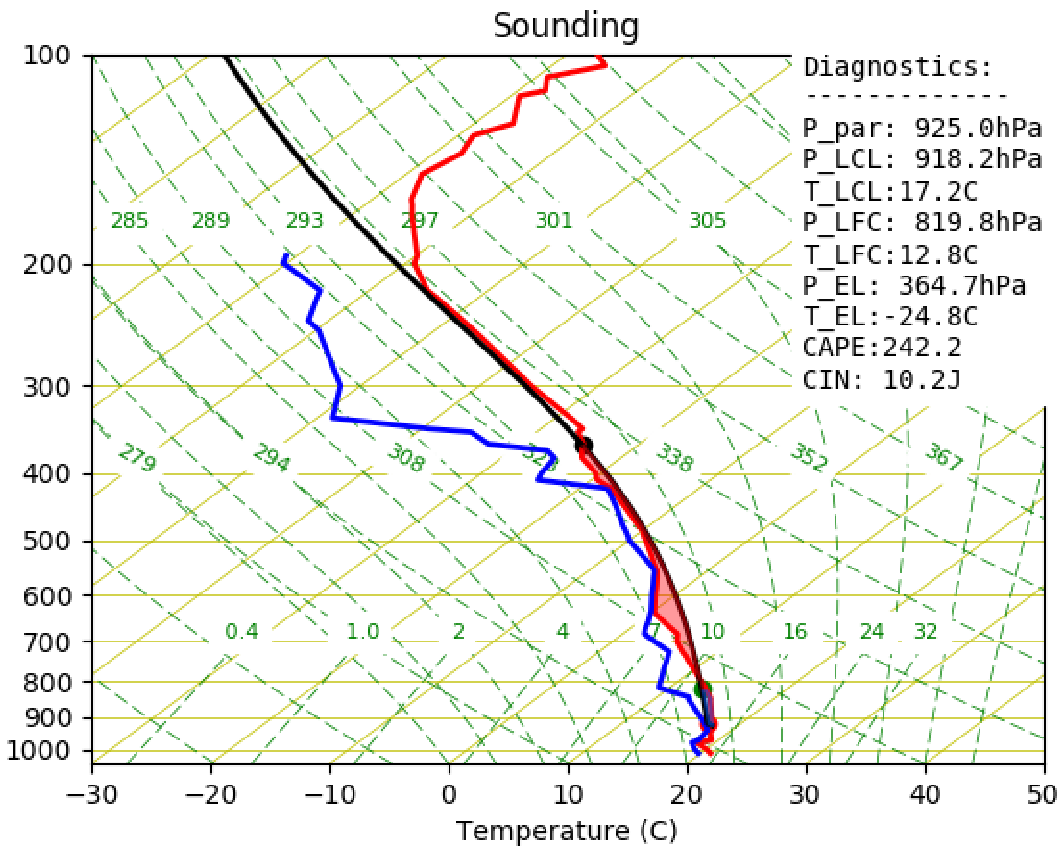
Now we can do the same thing, but with more control over the Figure:

```
# Import the new figure class
from SkewTplus.skewT import figure

from SkewTplus.sounding import sounding


#Load the sounding data
mySounding = sounding("./exampleSounding.txt")

# Create a Figure Manager
mySkewT_Figure = figure()

# Add an Skew-T axes to the Figure
mySkewT_Axes = mySkewT_Figure.add_subplot(111, projection='skewx')


# Extract the data from the Sounding
pressure = mySounding.soundingdata['pres']
temperature =  mySounding.soundingdata['temp']
dewPointTemperature = mySounding.soundingdata['dwpt']

# Add a profile to the Skew-T diagram
```

(continues on next page)

(continued from previous page)

```
#  method=0 -> Most unstable parcel
#  diagnostics -> add a text box in the Figure with the parcel analysis results
mySkewT_Axes.addProfile(pressure,temperature, dewPointTemperature ,
                            hPa=True, celsius=True, method=0, diagnostics=True)

# Show the figure
mySkewT_Figure.show()
```



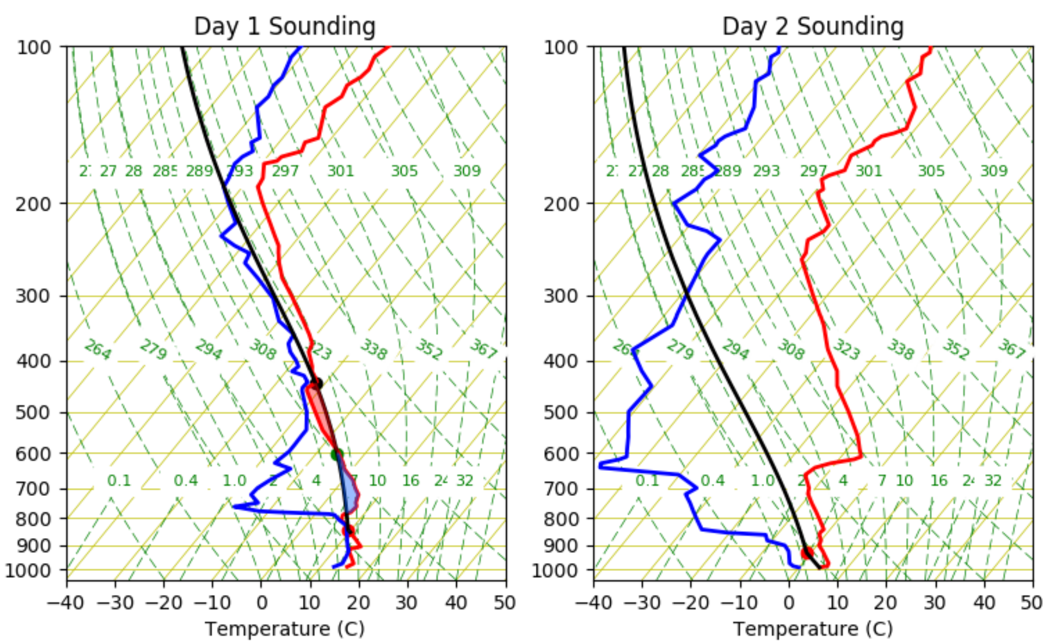
Lets now complicate the things a little bit and show one of the new capabilities of the package. Let suppose that we want to compare two soundings, with the parcel analysis, and plot them side to side:

```
#Load the sounding data
mySounding1 = sounding("./bna_day1.txt")
mySounding2 = sounding("./bna_day2.txt")

# Create a Figure Manager with a suitable size for both plots
mySkewT_Figure = figure(figsize=(9,5))

# Now we want to create two axes side to side

# Add the first Skew-T axes to the Figure
mySkewT_Axes1 = mySkewT_Figure.add_subplot(121, projection='skewx',tmin=-40)
```

(continues on next page)

(continued from previous page)

```
# Extract the data from the Sounding
pressure = mySounding1['pres']
temperature =  mySounding1['temp']
dewPointTemperature = mySounding1['dwpt']

# Add a profile to the Skew-T diagram
mySkewT_Axes1.addProfile(pressure,temperature, dewPointTemperature ,
                        hPa=True, celsius=True, method=0, diagnostics=False)


mySkewT_Axes1.set_title("Day 1 Sounding")

# Add the second Skew-T axes to the Figure
mySkewT_Axes2 = mySkewT_Figure.add_subplot(122, projection='skewx',tmin=-40)

# Extract the data from the Sounding
pressure = mySounding2['pres']
temperature =  mySounding2['temp']
dewPointTemperature = mySounding2['dwpt']

# Add a profile to the Skew-T diagram
mySkewT_Axes2.addProfile(pressure,temperature, dewPointTemperature ,
                        hPa=True, celsius=True, method=0, diagnostics=False)

mySkewT_Axes2.set_title("Day 2 Sounding")

# Show the figure
mySkewT_Figure.show_plot()
```



The different sounding sources supported to initialize the `sounding` class are described in greater detail in the next chapter: *Sounding Initialization*

1.1.2 Sounding Initialization

The `sounding` class supports the following initialization modes:

- From a University of Wyoming txt file
- From a University of Wyoming Website
- Form an ARM sounding Netcdf file
- From a dictionary with the field names, field values pairs
- Adding individual fields manually

University of Wyoming Sounding Data

Fetch from txt file

The easiest way to get sounding data is to visit the University of Wyoming's website:

http://weather.uwyo.edu/upperair/sounding.html

To get some sounding data, follow the link and find the date, and location you are interested in, view the data as a text file and just save the file to your system. If you want to get loads of data please be considerate about the way you go about doing this! (Lots of wget requests makes the server administrators unhappy).

You can also pass your own data to SkewT by writing a text file in **identical** format to the University of Wyoming files, which are constant-width columns. Here's a sample of the first few lines of one of the bundled examples:

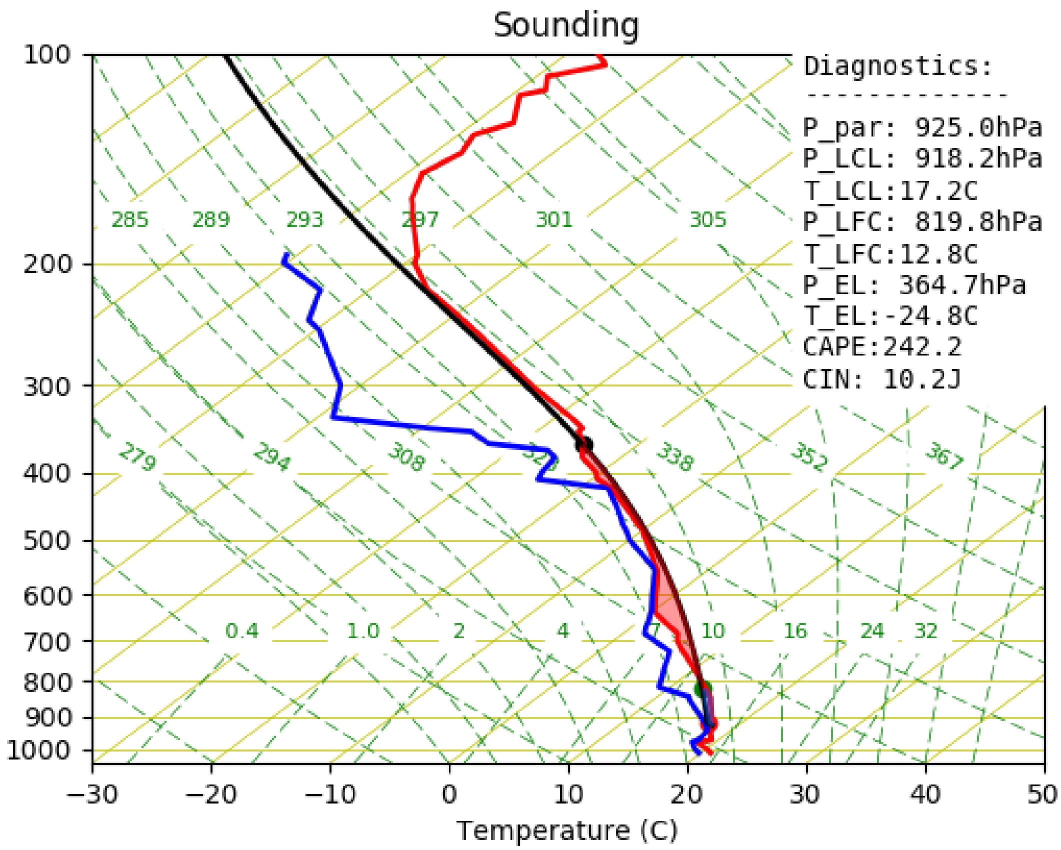
```
94975 YMHB Hobart Airport Observations at 00Z 02 Jul 2013

-----------------------------------------------------------------------------
   PRES   HGHT   TEMP   DWPT   RELH   MIXR   DRCT   SKNT   THTA   THTE   THTV
    hPa     m      C      C      %    g/kg    deg   knot     K      K      K
-----------------------------------------------------------------------------
 1004.0     27   12.0   10.2     89   7.84    330     14  284.8  306.7  286.2
 1000.0     56   12.4   10.3     87   7.92    325     16  285.6  307.8  286.9
  993.0    115   12.8    9.7     81   7.66    311     22  286.5  308.1  287.9
```

From now on, it's assumed that the package is installed and the current working directory is the *examples* one, included in this package.

To read a sounding from a txt file and create a quick plot using the default parameters we only have to do:

```
from SkewTplus.sounding import sounding

#Load the sounding data
mySounding = sounding("./exampleSounding.txt",fileFormat='txt')

#Do a quick plot
mySounding.quickPlot()
```

The resulting plot will look like this:



Fetch from University Of Wyoming Website

The sounding class supports getting University of Wyoming sound data directly from the UWYO website, we only need to specify the date and the station id.

For example, to initialize the class with with the sounding from "72558 OAX Omaha" station, at April 10th of 2017 OO UTC we simple do:

```
from SkewTplus.sounding import sounding

#Load the sounding data
mySounding = sounding("20170410:00",fileFormat='web', stationId= "OAX")

#Do a quick plot
mySounding.quickPlot()
```

ARM Sounding Data

The `sounding` class also supports initialization from ARM sounding data (Netcdf files). For example:

```
from SkewTplus.sounding import sounding

#Load the sounding data
```

(continues on next page)

(continued from previous page)

```
mySounding = sounding("./armSoundingExample.cdf",fileFormat='arm')

#Do a quick plot
mySounding.quickPlot()
```

From a dictionary

The `sounding` class can be initialized from a dictionary with "field names" , "field values" pairs. The Temperature should be in Celsius degrees and the pressure in hPa.

The next is an example of a dictionary initialization used to plot a sounding from a WRF output file:

```
from netCDF4 import Dataset
import numpy
from SkewTplus.sounding import sounding

#Load the WRF File
wrfOutputFile = Dataset("wrfOutputExample.nc")
theta = wrfOutputFile.variables["T"][:] + 300 # Potential temperature

# Pressure in hPa
pressure = (wrfOutputFile.variables['P'][:] + wrfOutputFile.variables['PB'][:])


qvapor = wrfOutputFile.variables['QVAPOR'][:]

qvapor = numpy.ma.masked_where(qvapor <0.00002, qvapor)

T0 = 273.15
referencePressure = 100000.0  # [Pa]
epsilon = 0.622  # Rd / Rv

# Temperatures in Celsius
temperature = theta* numpy.power((pressure / referencePressure), 0.2854) - T0
vaporPressure = pressure * qvapor / (epsilon + qvapor)

dewPointTemperature = 243.5 / ((17.67 / numpy.log(vaporPressure * 0.01 / 6.112)) - 1.
↪) #In celsius
dewPointTemperature = numpy.ma.masked_invalid(dewPointTemperature)


# Now we have the pressure, temperature and dew point temperature in the whole domain

# Select one vertical column , t =0 , x=30, y=30

inputData = dict(pressure=pressure[0,:,30,30]/100,
                 temperature=temperature[0,:,30,30],
                 dewPointTemperature=dewPointTemperature[0,:,30,30])

mySounding = sounding(inputData)
mySounding.quickPlot()
```

Adding Fields Manually

The `sounding` class supports an empty initialization (without any fields). With the `addField()` method, new fields can be added to the class. With this kind of initialization full control ever the fields added can be obtained. Internally, the class stores the field data values as `soundingArray` classes. This class is a MaskedArray with metadata (long Name,units and missing data value).

To exemplify the use of this initialization, the previous example of the sounding with WRF data coded to use the `addField()` method:

```
from netCDF4 import Dataset
import numpy
from SkewTplus.sounding import sounding


#Load the WRF File
wrfOutputFile = Dataset("wrfOutputExample.nc")
theta = wrfOutputFile.variables["T"][:] + 300 # Potential temperature

# Pressure in hPa
pressure = (wrfOutputFile.variables['P'][:] + wrfOutputFile.variables['PB'][:])


qvapor = wrfOutputFile.variables['QVAPOR'][:]

qvapor = numpy.ma.masked_where(qvapor <0.00002, qvapor)

T0 = 273.15
referencePressure = 100000.0   # [Pa]
epsilon = 0.622   # Rd / Rv

# Temperatures in Celsius
temperature = theta* numpy.power((pressure / referencePressure), 0.2854) - T0
vaporPressure = pressure * qvapor / (epsilon + qvapor)

dewPointTemperature = 243.5 / ((17.67 / numpy.log(vaporPressure * 0.01 / 6.112)) - 1.
↪) #In celsius
dewPointTemperature = numpy.ma.masked_invalid(dewPointTemperature)


# Now we have the pressure, temperature and dew point temperature in the whole domain

# Select one vertical column , t =0 , x=30, y=30

mySounding = sounding() # Create an empty sounding

#Add fields
mySounding.addField('pressure', pressure[0,:,30,30], "Pressure", "Pa")
mySounding.addField('temperature', temperature[0,:,30,30], "Temperature", "C")
mySounding.addField('dewPointTemperature', dewPointTemperature[0,:,30,30], "Dew Point␣
↪Temperature", "C")

mySounding.quickPlot()
```

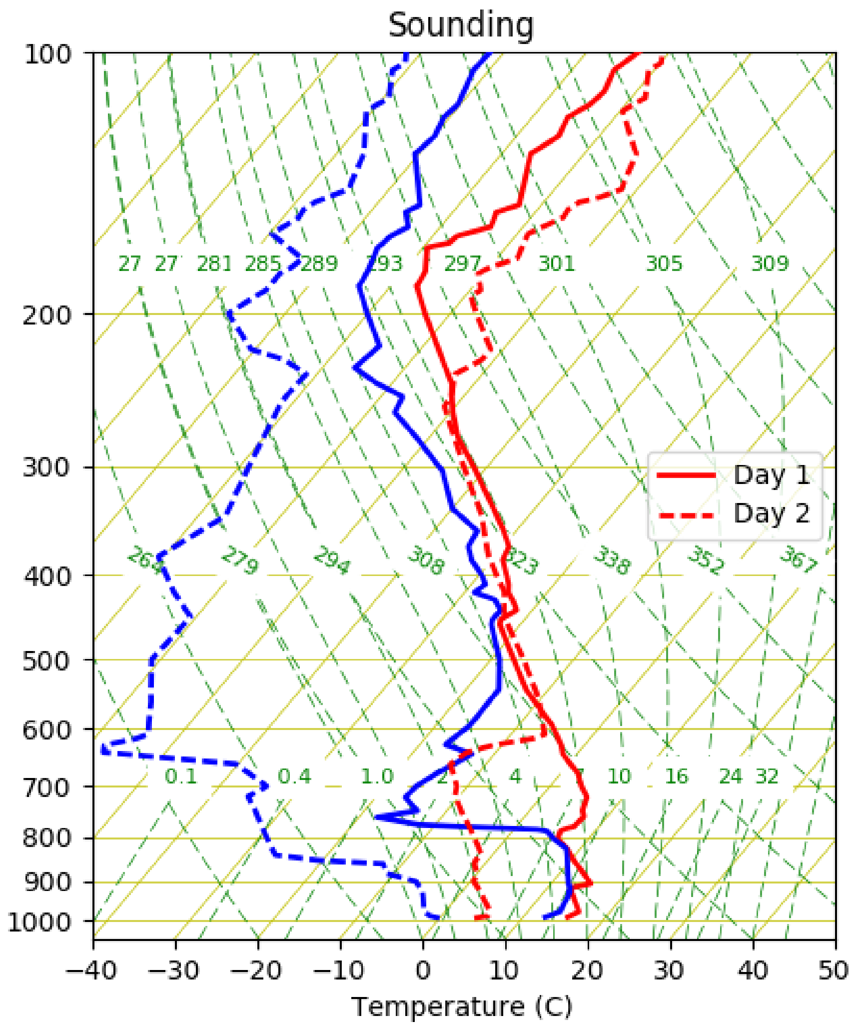
The profile plotting capabilities are described in greater detail in the next chapter: *Profile Plotting*

1.1.3 Profile Plotting

In this chapter the profile plotting capabilities are described of the SkewTplus package are described in greater detail.

In the following example show how to plot two soundings in the same Skew-T diagram without any parcel analysis:

```
from SkewTplus.skewT import figure
from SkewTplus.sounding import sounding

#Load the sounding data
mySounding1 = sounding("./bna_day1.txt")
mySounding2 = sounding("./bna_day2.txt")

# Create a Figure Manager with a suitable size for both plots
mySkewT_Figure = figure(figsize=(5,6))

# Add the Skew-T axes to the Figure
mySkewT_Axes1 = mySkewT_Figure.add_subplot(111, projection='skewx',tmin=-40)


# Add one profile to the Skew-T diagram
# The line style is set to be a solid line and a label is added
# to the plot. Since the label is not None, a legend will be added
# automatically to the plot
mySkewT_Axes1.addProfile(*mySounding1.getCleanSounding(),
                         hPa=True, celsius=True, parcel=False,
                         label='Day 1', linestyle='-')


# Add a second profile to the Skew-T diagram
# The line style is set to be a dashed line
# The location of the legend is specified to be
# 'center right'
mySkewT_Axes1.addProfile(*mySounding2.getCleanSounding(),
                         hPa=True, celsius=True, parcel=False,
                         label='Day 2', linestyle='--',loc='center right')

# Show the figure
mySkewT_Figure.show_plot()
```



For more details about the different profile plotting options see `SkewTplus.skewT.SkewXAxes.addProfile()`

In the next chapter the Parcel Analysis included in the SkewTplus package are described in greater detail: *Parcel Analysis*

1.1.4 Parcel Analysis

The SkewTplus package comes with the `SkewTplus.thermodynamics` module that allows the following computations for a parcel:

- `SkewTplus.thermodynamics.parcelAnalysis()`
- `SkewTplus.thermodynamics.liftParcel()`
- `SkewTplus.thermodynamics.moistAscent()`
- `SkewTplus.thermodynamics.getLCL()`

parcelAnalysis Function

The `SkewTplus.thermodynamics.parcelAnalysis()` function not only supports computations on 1D vertical soundings, also it allows to do the analysis in a 3D domain (height,latitude or y ,longitude or x) or 4D=(3D + time) ones (time,height,latitude or y ,longitude or x].

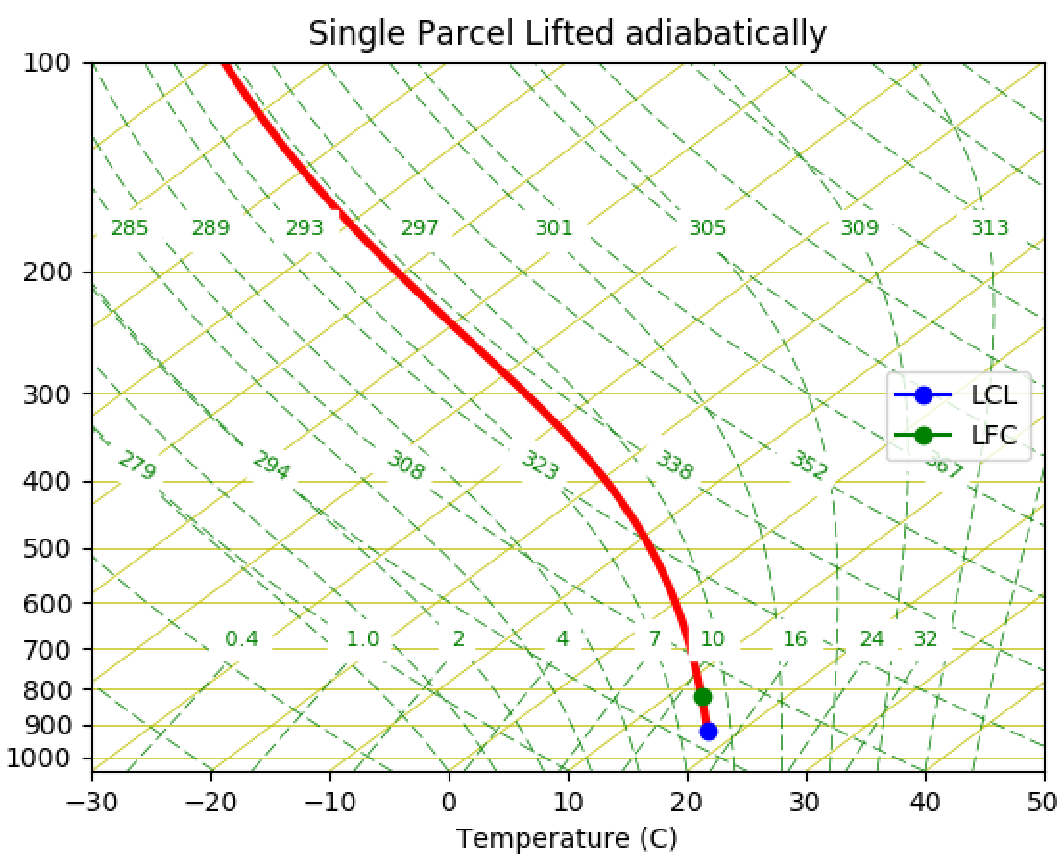
Below is simple example of how to perform a parcel analysis, print the results and then plot the parcel trajectory. For this example you need netCDF4 and Basemap packages installed:

```
from SkewTplus.skewT import figure
from SkewTplus.sounding import sounding
from SkewTplus.thermodynamics import parcelAnalysis, liftParcel


#Load the sounding data
mySounding = sounding("./exampleSounding.txt")

pressure, temperature, dewPointTemperature = mySounding.getCleanSounding()

# Perform a parcel analysis
# The full parcel analysis field is returned
# Most Unstable  parcel : method=0
# Start looking for the most unstable parcel from the first level (initialLevel=0)
# Use at maximum 5 iterations in the bisection method to find the LCL
# Since the sounding temperature and pressure are expressed in Celsius and hPa
# we set the corresponding keywords
myParcelAnalysis = parcelAnalysis(pressure,
                                  temperature,
                                  dewPointTemperature,
                                  hPa=True,
                                  celsius=True,
                                  fullFields=1,
                                  method=1,
                                  initialLevel=0,
                                  tolerance=0.1,
                                  maxIterations=20)

# Print the contents of the dictionary
for key,value in myParcelAnalysis.items():
    if isinstance(value, float) :
        print("%s = %.1f"%(key,value))
    else:
        print("%s = %s"%(key,str(value)))



#Plot the parcel trajectory in the SkewT diagram

# First we lift the parcel adiabatically
initialLevel = myParcelAnalysis['initialLevel']

parcelTemperature = liftParcel(temperature[initialLevel],
                               pressure,
                               myParcelAnalysis['pressureAtLCL'],
                               initialLevel=initialLevel,
                               hPa=True,
                               celsius=True)
```

(continues on next page)

(continued from previous page)

```
# Create a Figure Manager
mySkewT_Figure = figure()

# Add an Skew-T axes to the Figure
mySkewT_Axes = mySkewT_Figure.add_subplot(111, projection='skewx')

# Plot the parcel temperature
mySkewT_Axes.plot(parcelTemperature, pressure, linewidth=3, color='r' )

# Add a marker for the LCL and the LFC
mySkewT_Axes.plot(myParcelAnalysis['temperatureAtLCL'],
                  myParcelAnalysis['pressureAtLCL'],
                  marker='o', color='b' , label='LCL')
mySkewT_Axes.plot(myParcelAnalysis['temperatureAtLFC'],
                  myParcelAnalysis['pressureAtLFC'],
                  marker='o', color='g' , label='LFC')

# Add a legend
mySkewT_Axes.legend(loc='center right')

mySkewT_Axes.set_title("Single Parcel Lifted adiabatically")

mySkewT_Figure.show_plot()
```



In the next chapter, a more intensive use of the parcelAnalysis function is used to compute CAPE for a 3D domain from a WRF output file: *WRF Output CAPE plot*

1.1.5 WRF Output CAPE plot

The `SkewTplus.thermodynamics.parcelAnalysis()` function allows to compute CAPE and CIN not only in a single vertical sounding, it also supports the computation over 3D domains (height, latitude or y , longitude or x) or 4D=(3D + time) ones (time, height, latitude or y, longitude or x].

Here is an example for computing CAPE from a WRF output file and plot the values as a color plot over a map:

```
from mpl_toolkits.basemap import Basemap
from netCDF4 import Dataset
import numpy

from SkewTplus.thermodynamics import parcelAnalysis
import matplotlib.pyplot as plt

#Load the WRF File
wrfOutputFile = Dataset("wrfOutputExample.nc")

theta = wrfOutputFile.variables["T"][:] + 300 # Potential temperature
pressure = wrfOutputFile.variables['P'][:] + wrfOutputFile.variables['PB'][:]

qvapor = wrfOutputFile.variables['QVAPOR'][:]

qvapor = numpy.ma.masked_where(qvapor <0.00002, qvapor)

T0 = 273.15
referencePressure = 100000.0  # [Pa]
epsilon = 0.622  # Rd / Rv

temperature = theta* numpy.power((pressure / referencePressure), 0.2854) - T0

vaporPressure = pressure * qvapor / (epsilon + qvapor)

dewPointTemperature = 243.5 / ((17.67 / numpy.log(vaporPressure * 0.01 / 6.112)) - 1.
↪) #In celsius
dewPointTemperature = numpy.ma.masked_invalid(dewPointTemperature)


# Now we have the pressure, temperature and dew point temperature in the whole domain
# Compute the parcel analysis for each vertical column and each time
#
# fullFields =0 , only return CAPE and CIN
# Most Unstable  parcel : method=0
# Start looking for the most unstable parcel from the first level (initialLevel=0)
# Use at maximum 5 iterations in the bisection method to find the LCL
# Since the sounding temperature and pressure are expressed in Celsius and hPa
# we set the corresponding keywords
myParcelAnalysis = parcelAnalysis(pressure,
                                  temperature,
                                  dewPointTemperature,
                                  hPa=False,
                                  celsius=True,
                                  fullFields=0,
                                  method=0,
```

(continues on next page)

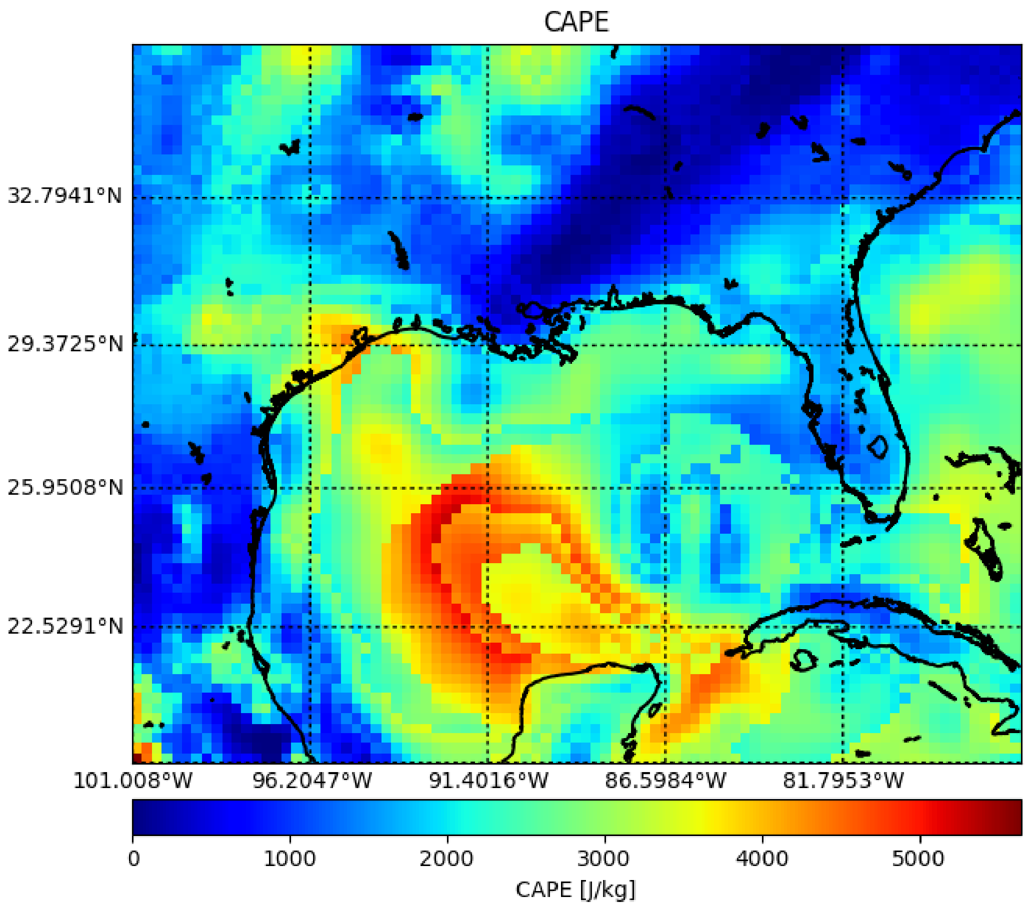
(continued from previous page)

```
                                  initialLevel=0,
                                  tolerance=0.1,
                                  maxIterations=20)


## Create the Base Map for the CAPE color plot

# Read the temperature and pressure fields
lon = wrfOutputFile.variables["XLONG"][0, :, :]
lat = wrfOutputFile.variables["XLAT"][0, :, :]


#---Read lat,lon for plotting
lon = wrfOutputFile.variables["XLONG"][0, :, :]
lat = wrfOutputFile.variables["XLAT"][0, :, :]


# Define and plot the meridians and parallels
min_lat = numpy.amin(lat)
max_lat = numpy.amax(lat)
min_lon = numpy.amin(lon)
max_lon = numpy.amax(lon)

# Create the basemap object
myBaseMap = Basemap(projection="merc",
                    llcrnrlat=min_lat,
                    urcrnrlat=max_lat,
                    llcrnrlon=min_lon,
                    urcrnrlon=max_lon,
                    resolution='h')

# Create the figure and add axes
myFigure = plt.figure(figsize=(8,8))
myAxes = myFigure.add_axes([0.1,0.1,0.8,0.8])

# Make only 5 parallels and meridians
parallel_spacing = (max_lat - min_lat) / 5.0
merid_spacing = (max_lon - min_lon) / 5.0
parallels = numpy.arange(min_lat, max_lat, parallel_spacing)
meridians = numpy.arange(min_lon, max_lon, merid_spacing)

myBaseMap.drawcoastlines(linewidth=1.5)
myBaseMap.drawparallels(parallels,labels=[1,0,0,0],fontsize=10)
myBaseMap.drawmeridians(meridians,labels=[0,0,0,1],fontsize=10)

# Plot CAPE at time 0
CAPE = myParcelAnalysis['CAPE'][0,:]

myColorPlot = myBaseMap.pcolormesh(lon,lat, myParcelAnalysis['CAPE'][0,:],latlon=True,
↪ cmap='jet')

# Create the colorbar
cb = myBaseMap.colorbar(myColorPlot,"bottom", size="5%", pad="5%")
cb.set_label("CAPE [J/kg]")

# Set the plot title
myAxes.set_title("CAPE")
```

(continues on next page)

(continued from previous page)

```
plt.show()
```



1.2 Developer Reference

The intended audience of this guide is mainly to developers who use skewTplus. It also serves as a more comprehesive description of the packages modules. For a more general introduction aimed at users see the *User Reference*.

This guide provides documentation for all modules, function, methods, and classes within SkewTplus both those in the public API and private members.

Documentation is broken down by module (in order of relevance)

1.2.1 SkewTplus.sounding module

1.2.2 SkewTplus.skewT module

1.2.3 SkewTplus.thermodynamics module

1.2.4 SkewTplus._thermodynamics module

1.2.5 SkewTplus.errorHandling module

1.3 SkewTplus – License

The following copyright information is retained for the parts of the package that were provided originally. These include the matplotlib SkewAxes classes amongst others, as indicated within the source code.

CHAPTER 2

Dependencies

The SkewTplus package need the following dependencies

- matplotlib
- numpy
- cython (optional)
- netCDF4
- six
- future (python2)
- hdf4
- libgcc >=5
- requests

For running the WRF data example:

- Basemap

CHAPTER 3

Installing SkewTplus

3.1 IMPORTANT - OSX installation

Before installing the package, be sure that Numpy is installed. Then, install the apple's Xcode application by running:

```
xcode-select --install
```

Before running the pip or the setup commands execute:

```
export CC=clang ; export CXX=clang
```

Then you can continue with any of the following installation procedures.

Nevertheless **pip** is highly recommended.

3.2 PIP install

To install the package using **pip** the numpy package must be already installed. If is not installed, you can install it by running:

```
pip install numpy
```

After the numpy package was installed, to install the SkewTplus package run:

```
pip install SkewTplus
```

3.3 Install from source

The latest version can be installed manually by downloading the sources from https://github.com/aperezhortal/SkewTplus

To install the package manually, the numpy package must be already installed. If is not installed, you can install it by running:

```
pip install numpy
```

Then, you can install the SkewTplus package executing:

```
python setup.py install
```

If you want to put it somewhere different than your system files, you can do:

```
python setup.py install --prefix=/path/to/local/dir
```

IMPORTANT: If you install it using this way, all the dependencies need to be already installed!

3.4 Conda install - Only available linux users

If you are using an anaconda environment, to install the package execute:

```
conda install -c andresperezcba skewtplus
```

CHAPTER 4

Contributions

SkewTplus is an open source software project. Contributions to the package are welcomed from all users. Feel free to suggest enhancements or report bugs by opening an issue in the github project page:

https://github.com/aperezhortal/SkewTplus/issues

Thanks for using the SkewTplus package, for any feedback feel free to write to andresperezcba AT gmail DOT com

4.1 Code

The latest source code can be obtained with the command:

```
git clone https://github.com/aperezhortal/SkewTplus.git
```

If you are planning on making changes that you would like included in SkewTplus, forking the repository is highly recommended.