User guide for a GFS prediction downloader module

Tsilias Kimon Sotirios

The scheduled GFS prediction scripts are separated in two files:

- get_gfs_kim_func.py : contains all the functions needed
- get_gfs_kim_schedule.py : is the scheduled script that downloads the data.

Both scripts need to be in the same path.

IMPORTANT: The getgfs module needs to be pip installed, through anaconda shell prompt (or any other).

Other modules needed :

- datetime
- numpy

The first cell of get_gfs_kim_schedule.py consists of the module imports and prints the current date and time. None of these should be changed.

```
import getgfs
from datetime import datetime, timedelta
import numpy as np
#import matplotlib.pyplot as plt
#import matplotlib.ticker as ticker
import sys
import re
from get_gfs_kim_func import *
print("Current datetime: "+str(datetime.now()))
```

The second cell contains

- Parameters: the total list of GFS parameters
- Cords: the coordinate edges (lat and long). By default Greece is set
- Sres: the special resolution. Default 0.25
- Pred_hrs: hours ahead to be downloaded. Default 12hr
- Rest_high: if sres is 0.25 this setting is available
- Path: the path where the files are saved
- Parm: the parameters which are saved into txt
- Mb: the mbl heigh values to be downloaded. If parameter which doest have a height value is selected in the list, then the output file has only 1 value column

All these variables can be altered according to the user's needs.

```
#%%

parameters=["acpcpsfc","cpratavesfc","rhprs","hgtprs","tmp2m","tmpprs","ugrdprs","vgrdprs","tcdcprs","vissfc","cape255_
cords=[32.75,43.75,17.75,31]
sres="0p25"
pred_hrs=12
tres_high=True
path=r'C:\Users\Kimon\Downloads\\'

parm=["rhprs","hgtprs","tmpprs","ugrdprs","vgrdprs","tcdcprs","tmp2m"]
mb= [100, 200 ,300, 400, 500, 600, 700, 750, 800, 850, 900, 950, 1000]

```

*Note if in the parm list a non height dependent parameter is selected only the base value will be imported.

The 3rd and 4th cell require no change or inputs. They pull and save the date respectively.

```
#%%
#setting up the data
pulldate,lat_str,lon_str,lat,lon,pdtxt=prepare_in(cords,sres,pred_hrs)
#pulling the data
gfs,res=download_data(pulldate=pulldate,tres_high=tres_high)


cols=["lat","lon"]

for i in mb:
    cols.append(str(i)+"mbl")
#%%
for par in parm:
    #getting the parameter's original name from the database
    lname=get_long_name(gfs, par)
    print('Parameter selected: '+par+'( '+lname+')')
    if par in ["rhprs","hgtprs","tmpprs","ugrdprs","vgrdprs","tcdcprs"]:
        matrix=np.empty((len(lat)*len(lon),2+len(mb)))
        e=0
        for i in range(len(lat)):
            for j in range(len(lon)):
                matrix[e,0]=lat[i]
                matrix[e,1]=lon[j]
                e+=1
        k=0
        for l in mb:

            #separating the specific parameter,selecting mb value and filtering missing values
            rt=get_data(gfs,res,par,l,fltr=True)

            e=0
            for i in range(rt.shape[0]):
                for j in range(rt.shape[1]):
                    matrix[e,2+k]=rt[i,j]
                    e+=1
            k+=1
        save_as_txt(matrix,pdtxt+" "+par,path,cols)
    if par in ["acpcpsfc","cpratavesfc","tmp2m","vissfc","cape255_0mb"]:
        matrix=np.empty((len(lat)*len(lon),3))
        #separating the specific parameter,selecting mb value and filtering missing values
        rt=get_data(gfs,res,par,False,fltr=True)


        e=0

        for i in range(rt.shape[0]):
            for j in range(rt.shape[1]):
                matrix[e,0]=lat[i]
                matrix[e,1]=lon[j]
                matrix[e,2]=rt[i,j]
                e+=1

        save_as_txt(matrix,pdtxt+" "+par,path,["lat","lon",par])
```

Each parameter is saved on a separate txt with columns marked on the top of the txt heading (Lat,long,mb1,etc).

There is an additional script named get_gfs_kim.py, a demo script that lets the user download data and plot them to visualize them. This script requires matplotlib for the plotting.

The first cell imports the required modules and downloads the date. The cell contains variables which can be altered:

- Parameters
- Cords
- Sres
- Pred_hrs
- Tres_high

```python
# -*- coding: utf-8 -*-
"""
Created on Sat Dec  2 18:11:37 2023

@author: Kimon
"""

import getgfs
from datetime import datetime, timedelta
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import sys
import re
from get_gfs_kim_func import *
print("Current datetime: "+str(datetime.now()))


parameters=["acpcpsfc","cpratavesfc","rhprs","hgtprs","tmp2m","tmpprs","ugrdprs","vgrdprs","tcdcprs","vissfc","cape255_
cords=[32.75,43.75,17.75,31]
sres="0p25"
pred_hrs=12
tres_high=True


#setting up the data
pulldate,lat_str,lon_str,lat,lon,pdtxt=prepare_in(cords,sres,pred_hrs)
#pulling the data
gfs,res=download_data(pulldate=pulldate,tres_high=tres_high)

```

The second cell prints the code names and long name format for all the downloaded parameters. This allows the user to better understand the data.

```python
#%%
for p in parameters:
    lname=get_long_name(gfs, p)
    print(p+":"+lname)

```

The third cell lets the user load a specific parameter along with the mbl height. Only par and mb are alterable

```python
#%% Select a parameter

#selecting a parameter
par='tcdcprs'
mb=200

#getting the parameter's original name from the database
lname=get_long_name(gfs, par)
print('Parameter selected: '+par+'( '+lname+')')

#separating the specific parameter,selecting mb value and filtering missing values
rt=get_data(gfs,res,par,mb,fltr=True)

```

The fourth cell plots the data onto a figure.

```
#%%
#plotting
plt.imshow(rt,cmap='jet',origin='lower')
cb=plt.colorbar()
cb.ax.set_ylabel(lname, rotation=270,labelpad=15)
# Set the x-axis labels
plt.xticks(np.arange(len(lon)), lon)

# Set the y-axis labels
plt.yticks(np.arange(len(lat)), lat)
# Reduce the number of ticks
ax = plt.gca()
ax.xaxis.set_major_locator(ticker.MaxNLocator(integer=True))
ax.yaxis.set_major_locator(ticker.MaxNLocator(integer=True))

plt.title('datetime: '+pulldate)
```

*Note:

If any user wants to explore the functions further, it can be done through get_gfs_kim_func.py as all functions contain docstring and comments.

For the automated set up we used Task Scheduler. The task was named “gfs data download”. It runs every 12 hours at 00:00:10 and 12:00:10. It’s available by running Task Scheduler (administration mode), on the left select “Task Scheduler Library” to see the list of scheduled tasks. By double clicking on the task and navigating to Triggers tab, we can select the trigger and edit it. In the actions tab: Program/Script is: C:\Users\LAB-A\anaconda3\python.exe.

For Add Arguments section: "C:\GFS data\get_gfs_kim_schedule.py". This can be replaced with any other script in the future if the user wishes to.