Start Using GrADS

A. Chakraborty `and` R. S. Nanjundiah

Distributed in

A Short Term Course on

Physics of Atmosphere and Oceans

(1st to 12th July 2002)

Centre for Atmospheric and Oceanic Sciences
Indian Institute of Science
Bangalore 560012
INDIA

Contents

1 What is this?

In this documentation we give a quick introduction to the 'Grid Analysis and Display System (GrADS)' software package. This package is very useful for atmospheric data analysis. This is a freeware and can be downloaded from

`http://grads.iges.org/grads/downloads.html`

An extensive documentation of this software is available in

`http://grads.iges.org/grads/gadoc/`

The purpose of the current documentation is to make a new user familiar with GrADS providing examples of useful commands and functions.

2 Atmospheric variables and data file

Temperature, water vapor amount, wind speed and wind direction are some examples of atmospheric variables. These variables have values at every point of the earth's atmosphere. For analysis point of view, the earth's atmosphere is divided into a number grid boxes in horizontal, and many levels in the vertical. Hence, a three dimensional structure is formed consisting of several tiny-boxes. Three dimensional continuous atmospheric fields are discretised assigning a single value to each of these boxes. When these boxes are regularly spaced in the horizontal and vertical, we say that the variable is *gridded*. This variable may be stored in a data file for analysis. One or more variables stored in a data file have to have a description of how the data is stored because every data value for a single variable can have at most four dimension stamps. Description of the data is generally given in a separate file call 'data descriptor' file. Even, this description may be embedded in the data file itself, which is termed as self descriptor file.

3 How GrADS views a data file

GrADS views a data file to be of five dimensional. The dimensions are longitude, latitude, vertical levels, time, and different variables. GrADS assumes that in the data file longitude varies fastest, then latitude, then vertical levels, then variables, and then time dimension. A data descriptor file contains position and time informations for all variables stored in a data file.

4 Starting of GrADS

This software can be invoked using the command *grads* on the unix command prompt. A graphics window is opened whose orientation can be specified using options -l for landscape (default) or -p for portrait on the command prompt. Otherwise, it asks for the orientation before opening. The prompt appears:

```
ga− >
```

which waits for the user command.

5 Open a data descriptor file

If the data descriptor file name is *model.ctl* then we open it using the command

```
ga− > open model.ctl
```

Here, the extension .ctl can be omitted, because this is the default file extension for GrADS data descriptor file.

6 The first plot

If we know the name of any variable (say *tmp*) in the descriptor file then this variable can be plotted using command

```
ga− > d tmp
```

If *tmp* varies in longitude and latitude in the descriptor file, then a contour plot appears on the graphics window. This plot is for the first time stamp and first level as described in the descriptor file.

7 Make it little colorful

The type of graphics output can be specified using the command

```
ga− > set gxout GraphicType
```

There are several options available for the *GraphicType*. In the present case, to fill the plot area with colors, use graphic type *shaded*. But before we make a shaded plot we need to clear the previous output. This is done by using

```
ga− > c [or clear]
```

Now the command

```
ga− > set gxout shaded
```

will change the output type to shaded. And plot the variable again using

```
ga− > d tmp
```

Instead of lines of different values, the window is filled with different colors for different ranges of the data value. To know which color corresponds to which range use

```
ga− > run cbar
```

A color bar appears describing the range of each color used in the figure.

8 Zoom in to a region

If the input data is global, both the previous plots display the variable all over the globe. But what if we are interested only to a particular region, say the Indian region? Any region can be set using commands

```
ga− > set lon 60 100
ga− > set lat 0 30
```

These set the longitude from 60E to 100E and latitude from the Equator to 30N. So, to plot only for this region,

```
ga− > c
ga− > d tmp
ga− > run cbar
```

Notice the orientation of the color bar for different combinations of longitude and latitude ranges. Sometimes the color bar is vertical and sometimes it is horizontal. We can specify its position, orientation and length scale using a different command

```
ga− > run cbarn sf vert xmid ymid
```

where,
sf $\rightarrow$ scale the whole color bar 1.0 = original 0.5 half the size, etc.
vert $\rightarrow$ 0 FORCES a horizontal bar = 1 a vertical color bar
xmid$\rightarrow$ the x position on the page of the center the color bar
ymid $\rightarrow$ the y position on the page of the center the color bar
Therefore,

```
ga− > run cbarn 1.5 1 5.5 2.0
```

draws a vertical color bar of length 1.5 of the previous length (obtained using run cbar) with its middle at the coordinate (5.5,2.0).

9 Change the level and time

For a variable with multiple layers (in the vertical) and with many time points the default level and time will be the first as specified in the descriptor file. We set the level and time using

ga− > `set lev` *level*
ga− > `set time` *TimeDate*

For different pressure level data (in hPa), use

```
ga- > set lev 850
```

to set the level to 850 hPa, and for monthly data use

```
ga- > set time jul
```

to set the time to July of the year when the data time starts in the descriptor file. Now do,

```
ga- > c
ga- > d tmp
ga- > run cbar
```

to get a plot of the variable *tmp* in the same region set above at 850 hPa and for the month of July.

10 How do I know the current dimension environment?

The dimension environment currently working for the internal GrADS calculations and graphics output can be known using

```
ga- > q dims
```

which shows the default file and dimensions. The default file is a file from which default dimensions are set. This, by default, is the first file open and can be changed using

ga− > `set dfile` *FileNumber*

where *FileNumber* is the sequence number to the file opened.

11 Open and display from multiple files

Many times different data variables are stored in different files. One simple example is the two components of the horizontal wind vector $\vec{V}$. If the zonal component (x direction, will be called u) is stored in one file described by *ugrd.ctl* and the meridional component (y direction, will be termed as v) is stored in another file described by *vgrd.ctl* then we open both the files using

```
ga-> reinit
ga-> open ugrd.ctl
ga-> open vgrd.ctl
```

and display the zonal component (say, variable name is u) using

```
ga-> d u
```

and the meridional component (say, variable name is v) using

```
ga-> d v.2
```

The *.2* after v in the second case tells GrADS that this variable will come from the second file opened. To display the wind vector use

```
ga-> c
ga-> set gxout vector
ga-> d u;v.2
```

Is the plot too hazy, with vectors overlapping one on the other? We can skip vectors in x and/or y directions using the function *skip(var,nx,ny)*, where *var* is the variable, integers *nx* and *ny* specifies that every nx^{th} and ny^{th} data should be plotted in the x (longitude) and y (latitude) directions. So,

```
ga-> c
ga-> d skip(u,5,4); v.2
```

plots every 5^{th} and every 4^{th} vectors in the x and y directions respectively.

12 Do animation in time and level

To see how 850 hPa wind changes with season, do animation using

```
ga-> c
ga-> set lon 0 360
```

```
ga− > set lat -90 90
ga− > set lev 850
ga− > set time jan dec
ga− > d skip(u,3,3);v.2
```

which animates wind vectors for the 12 months. To see the animation with levels instead of in time, use

```
ga− > c
ga− > set time jul
ga− > set loopdim z
ga− > set z 1 17
ga− > d skip(u,3,3);v.2
```

The command ‘set loopdim z’ sets the looping dimension to z (levels). By default looping dimension is t (time), which can be set back using

```
ga− > set loopdim t
```

We can set the looping dimension to longitude (x) or latitude (y) as well.

13 Vertical profile of temperature at a point

Many times it is required to know the vertical structure of a field along a latitude or longitude or even at a single point. This can be obtained as follows. We reinitialize GrADS and open the file *all.prs* in which all the required variables are described.

```
ga− > reinit
ga− > open all.prs
```

Now set the specific dimensions,

```
ga− > c
ga− > set lon 85
ga− > set lat 20
ga− > set lev 1000 10
ga− > set time jul
```

and use command

```
ga− > d tmp
```

to get the vertical profile of temperature at 85E, 20N for the month of July.

14 Walker and Hadley circulations

Get a longitude-pressure cross section at a latitude using

```
ga− > c
ga− > set lon -180 180
ga− > set lat 20
ga− > set z 1 17
ga− > set time jul
ga− > d skip(u,3,1);-vvel*1.0e3;
```

where we have assumed that vvel is the pressure velocity, which is positive downwards. The Walker circulation can be seen in this plot where we get ascending motion in the Indian region and descending motion along 120E–160E. We multiply *vvel* by $1.0e3$ to make it of the same order of *u*. The negative sign is to make it positive upwards.

Now get a latitude-pressure cross section using commands

```
ga− > c
ga− > set lon 85
ga− > set lat -90 90
ga− > d skip(v,1,1);-vvel*1.0e3
```

Ascending motion along 30N and descending motion in the southern hemisphere can be noticed, which is a characteristic Hadley circulation.

15 Make Average

All the above mentioned examples use the data as it is, without doing any mathematical operation on it. GrADS has many functions to manipulate the data as per the requirement. The function ave() is one of them. To plot the June-July-August-September average temperature at the 500 hPa level we use this function as follows

```
ga− > c
ga− > set gxout shaded
ga− > set lon 0 360
ga− > set lat -90 90
ga− > set lev 500
ga− > set time jun
ga− > d ave(tmp,time=jun,time=sep)
ga− > run cbar
```

The first argument to ave() is the variable, second argument is the starting time and third argument is the ending time for averaging.
A zonally averaged temperature profile for the month of July can be obtained using

```
ga− > c
ga− > set lon 0
ga− > set lat -90 90
ga− > set lev 1000 10
ga− > set time jul
ga− > d ave(tmp,x=1,x=144)
ga− > run cbar
```

What is `x=1,x=144` in the `ave()`? x denotes the grid number in east-west direction. Here it is assumed that the descriptor file starts from 0.0E and has 144 grid points in east-west, with a resolution of 2.5°. Therefore the `ave()` here will average temperature zonally. We can as well get the same average using *lon* as the dimension variable instead of *x*. Let's try it,

```
ga− > c
ga− > d ave(tmp,lon=0,lon=360)     !  this is wrong for zonal average
ga− > run cbar
```

As commented, the above example will not give the exact result as the previous one. This can be easily checked making the difference

```
ga− > c
ga− > d ave(tmp,x=1,x=144) - ave(tmp,lon=0,lon=360)
ga− > run cbar
```

Why is the difference? Because, 0 and 360 are the same locations, and therefore by doing `d ave(tmp,lon=0,lon=360)` we are including one grid more in the longitudinal direction. So, the correct average using *lon* as the dimension will be

```
ga− > c
ga− > d ave(tmp,lon=0,lon=357.5)
ga− > run cbar
```

because, the end grid location of 2.5° resolution data which starts at 0.0E will be 357.5E. This average give the same result as `d ave(tmp,x=1,x=144)`, which can be checked plotting difference.

Meridional means are obtained using commands something like

```
ga− > c
ga− > set gxout shaded
ga− > set lon 0 360
ga− > set lat 0
```

```
ga− > set lev 1000 10
ga− > set time jul
ga− > d ave(tmp,y=1,y=73)
ga− > run cbar
```

where again we use south-north grid index for averaging. Now, to average only to the latitude belt 15N to 25N use

```
ga− > c
ga− > d ave(tmp,lat=15,lat=25)
ga− > run cbar
```

The `ave()` does area weighted average in the latitudinal direction. To do average without weight use `mean()`

```
ga− > c
ga− > d mean(tmp,lat=15,lat=25)
ga− > run cbar
```

Area average is obtained using `aave()`

```
ga− > c
ga− > set x 1
ga− > set y 1
ga− > set lev 850
ga− > set time jul
ga− > d aave(tmp,x=1,x=144,y=1,y=73)
```

which again makes an area-weighted global average of temperature at 850 hPa for the month of July. The result is printed on the screen instead of plotting on the graphic window, because the output is a single number.

To get a time series of average temperature in the Indian region at 850 hPa from January to December use

```
ga− > c
ga− > set x 1
ga− > set y 1
ga− > set lev 850
ga− > set time jan dec
ga− > d tloop(aave(tmp,lon=70,lon=90,lat=10,lat=30))
```

The highest temperature may be noticed in the month of May, just before the starting of summer monsoon in this region.

16 What is undefined value?

If value of any variable at one or more grid points is not known then in the data file in place of that grid we put a number which is well outside the range of the variable. This value is called *undefined* value in GrADS and is mentioned in the descriptor file as *undef*. While processing the data, if GrADS finds any value equal to this *undef* then that data point is not considered for any calculations or plotting. This feature can be used to exclude certain data points in GrADS. One good example of this is land-ocean mask. Say, in a file we have a variable having value 1.0 over the land and value 0.0 over the ocean, and this file is described by file *mask.ctl*. Now we open this file as the second file

```
ga− > c
ga− > open mask.ctl
ga− > set lon 0 360
ga− > set lat -90 90
ga− > set t 1
ga− > d mask.2(t=1)
ga− > run cbar
```

What appears on the graphic window is a shaded plot having value 1.0 over the land and 0.0 over the ocean. While plotting we used *(t=1)* after the variable name because in general time dimension of the first and the second files are not be same. Command `d mask.2(t=1)` says that plot the second variable for the first time of the second file. The *maskout*() function takes two variables or expressions as arguments. If the second variable at any grid is negative then the value of the first variable for that grid is put as *undef*. The return of the function is the modified field of the first variable. For example, if we do

```
ga− > c
ga− > d maskout(tmp,mask.2(t=1)-1)
ga− > run cbar
```

then a shaded plot is obtained only over the land. The expression `mask.2(t=1)-1` makes the values of the second argument over land 0.0 and over ocean −1.0, and the maskout function puts the values of *tmp* undefined over the ocean. Values of *tmp* are unchanged over the land. Now we plot the time series of 850 hPa temperature only over the land of the Indian region using the procedure described above:

```
ga− > c
ga− > set x 1
ga− > set y 1
ga− > set lev 850
ga− > set time jan dec
```

```
ga− > d tloop(aave(maskout(tmp,mask.2(t=1)-1),lon=70,lon=90,lat=10,
lat=30))
```

17 Put text on the plot

The plot can be made self descriptive adding axes labels and title. Commands

```
ga− > set strsiz 0.18 0.17                        !  sets string size
ga− > set string 2 c 6 0  !  sets color, justification, thickness and
rotation
ga− > draw string 5.5 8.25 Time series of 850hPa Temperature over
Indian region
```

draws the title of the plot with its center at 5.5 and 8.25 of the page coordinate. Now we draw the y-axis label:

```
ga− > set strsiz 0.16 0.14
ga− > set string 1 c 5 90
ga− > draw string 0.5 4.25 Temperature (K)
```

18 Save graphic output to a file

The graphic output can be saved to a file using

```
ga− > enable print temp850.gm
ga− > print
ga− > disable print
ga− >!gxeps -c temp850.gm
```

At first the graphic information is printed to a meta file named *temp850.gm*. The command *gxeps* is a shell command, which converts the meta file to encapsulated postscript file named *temp850.eps*. Any shell command can be invoked from GrADS command prompt preceding it by an exclamation sign (!). If the option `-c` is omitted then color information is not stored in the postscript file.

19 Write everything to a script file

Any GrADS commands can be written to a script file and the file can be `run` from the command prompt to invoke all the commands contained in the file. We put the previous series of commands to a file named *temp.gs*:

```
* A comment line in a script file starts with an '*' in the 1st
column
* This is how the file temp.gs will look like:
'reinit'
```

```
'open all.prs'
'open mask.ctl'
'set x 1'
'set y 1'
'set lev 850'
'set time jan dec'
'd
tloop(aave(maskout(tmp,mask.2(t=1)-1),lon=70,lon=90,lat=10,lat=30))'
'set strsiz 0.18 0.17'
'set string 2 c 6 0'
'draw string 5.5 8.25 Time series of 850hPa Temperature over Indian
region'
'set strsiz 0.16 0.14'
'set string 1 c 5 90'
'draw string 0.5 4.25 Temperature (K)'
```

Now run the file from GrADS command prompt:

```
ga- > run temp.gs
```

or simply

```
ga- > temp[.gs]
```

to get the same output as obtained before.