# 常用气象格式的数据读取及可视化

hzwjy 2012-11-1

本文介绍了气象研究(非业务使用)中的常用的数据格式，读写操作和数据可视化方法。

# 目录

# 1. 常用编程语言和工具

气象研究中常用的编程语言包括 C 语言、FORTRAN 语言，其中 FORTRAN 为"FORmula TRANslator"的缩写，意为"公式翻译器"，在数值计算、工程计算等领域有着广泛的应用，使用 FORTRAN 编写程序也是气象研究最基本的技能之一。

此外还会用到 IDL、MatLab、NCL、GrADS 等工具进行数据分析和可视化。IDL 和 MatLab 可用于数值计算和数据可视化，直接使用其内置的函数可以很方便地完成用户所需的各种计算，与 FORTRAN 相比，其码也更简洁。NCL 与 GrADS 也可以完成数据分析以及数据的可视化，而且它们都是专为气象领域开发的软件，对气象数据的支持较好，绘图也更为方便，经过简单的设置就可以读取数据并绘制各种图像。在平时的使用中，可以以其中的一种工具为主，再稍微了解另外的几种工具，配合使用达到效率的最大化。IDL 和 MatLab 都是商业软件，可以使用 GDL (GNU Data Language)和 SciLab 等作为替代。

除了上面提到的编程语言，也可以使用 Python (SciPy)计算机语言进行数据分析和绘图，也可以尝试 GMT (Generic Mapping Tools)、ncview、Panoply 等工具。

下面的实际操作中主要以 FORTRAN (使用 gfortran 编译器)和 NCL (NCL 6.1.0)来实现，本文仅对数据处理和绘图进行简单的介绍，相关语法请参考其它资料。

# 2. 常用的数据格式

## 2.1. 普通二进制格式

由于计算机存储文件都采用二进制形式，所以从广义上讲，所有的存储在设备中的文件都是二进制文件，包括 ASCII 及扩展 ASCII 字符中编写的数据或程序指令的文件。这里用"二进制格式"指除文本数据外的文件格式，而"普通"则是为了区分下面将提到的标准科学数据格式。这种文件的数据的存放序列与其在内存中的存放非常相似，所以在输入输出时几乎不需作转化。由于去掉了格式控制，与有格式文件相比，在使用数据信息时所做的处理更简洁迅速；同样也是这个原因使得无格式文件中即使存放着数字，也不能用文本编辑软件打开并看到。FORTRAN 的无格式文件便是这种普通二进制格式。

比如将 1~10 的整形数据存入一个直接存储的无格式文件中：

```
PROGRAM TEMPLATE

  IMPLICIT NONE

  INTEGER       ::  i

  OPEN(31, file="test.dat", access="direct", form="unformatted", recl=4)
  DO i = 1, 10
    WRITE(31, rec=i) i
  END DO

  CLOSE(31)
```

```
END PROGRAM TEMPLATE
```

使用文本编辑器打开只能看到无意义的乱码，使用十六进制编辑器(比如 WinHEX)打开后可以看到如下内容：

| Offset | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00000000 | 01 | 00 | 00 | 00 | 02 | 00 | 00 | 00 | 03 | 00 | 00 | 00 | 04 | 00 | 00 | 00 |
| 00000010 | 05 | 00 | 00 | 00 | 06 | 00 | 00 | 00 | 07 | 00 | 00 | 00 | 08 | 00 | 00 | 00 |
| 00000020 | 09 | 00 | 00 | 00 | 0A | 00 | 00 | 00 | | | | | | | | |

使用 NCL 可以很方便地读取这些数据，因为我们知道这里总共有 10 个整数，所以可以一次性将 10 个数据都读取出来：

```
; NCL 脚本
 begin

  fiDat = "./test.dat"
  data = fbindirread(fiDat, -1, (/10/), "integer")

  print(data)

end
```

如果采用顺序存储的方法存储相同的数据：

```
PROGRAM TEMPLATE

  IMPLICIT NONE

  INTEGER       ::  i

  OPEN(31, file="test.dat", access="sequential", form="unformatted")
  DO i = 1, 10
    WRITE(31) i
  END DO

  CLOSE(31)

END PROGRAM TEMPLATE
```

使用十六进制编辑器打开可以看到如下内容：

```
Offset   0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
00000000 04 00 00 00 01 00 00 00 04 00 00 00 04 00 00 00
00000010 02 00 00 00 04 00 00 00 04 00 00 00 03 00 00 00
00000020 04 00 00 00 04 00 00 00 04 00 00 00 04 00 00 00
00000030 04 00 00 00 05 00 00 00 04 00 00 00 04 00 00 00
00000040 06 00 00 00 04 00 00 00 04 00 00 00 07 00 00 00
00000050 04 00 00 00 04 00 00 00 08 00 00 00 04 00 00 00
00000060 04 00 00 00 09 00 00 00 04 00 00 00 04 00 00 00
00000070 0A 00 00 00 04 00 00 00
```

在每一个记录前后都有 4-byte 大小的数据(可以通过设置编译选项更改长度)，用于表示记录的开始和结束。这也是为什么同样的数据，采用顺序存储时文件比直接存储时大。

在使用普通二进制数据时还可能会遇到"端序(Endianness)"的问题，在计算机科学领域中，端序是指存放多字节数据的字节(byte)的顺序，典型的情况是整数在内存中的存放方式和网络传输的传输顺序。Endianness 有时候也可以用指位序(bit)。平时接触到的大部分二进制数据都是小端(little-endian)，大端(big-endian)多出现在数值模式的输入和输出中。

```
PROGRAM TEMPLATE

  IMPLICIT NONE

  INTEGER   :: i

  i = 305419896 ! 16 进制的 0x12345678
  OPEN(40, FILE = 'test1.dat', form = 'unformatted', access='direct', recl=4)
  WRITE(40, REC=1) i

END PROGRAM TEMPLATE
```

```
Offset   0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
00000000 78 56 34 12
```

这便是小端二进制数据的文件内容。

通过更改 FORTRAN 编译器设置，可以让程序输出大端数据：

```
$ gfortran -fconvert=big-endian foo.f90
```

再次运行程序后，输出数据的内容为：

```
Offset   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00000000 12 34 56 78
```

使用 NCL 读写二进制数据时，可以通过函数"*setfileoption*"设置文件的端序。

NCL 读写二进制数据的函数汇总：

```
Direct Access: data = fbindirread(path,rec,dim,type)
Sequential Access: data = fbinrecread(path,rec,dim,type)
Cray (C block IO write):   data = cbinread(path,dim,type)
```

```
Cray Sequential:    data = craybinrecread(path,rec,dim,type)
```

## 2.2. 文本数据

数据内容的记录是以 ASCII 字符的方式存在的，每一条记录是以 ASCII 码中的回车符 CR(0D)加换行符 LF(0A)来结束的，可以用文本编辑软件打开格式文件并直接看懂其内容。

比如一个文本文件的内容是：

```
12345
57890
```

用文本编辑器打开即可看到相应的内容，如果用十六进制编辑器打开该文件，可以看到字符的 ASCII 代码：

| Offset | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00000000 | 31 | 32 | 33 | 34 | 35 | 0D | 0A | 35 | 37 | 38 | 39 | 30 | 0D | 0A | | | 12345 57890 |

图中"31"对应字符"1"，"32"对应字符"2"，"0D"和"0A"对应回车和换行符。

使用 FORTRAN 格式化输入输出方式即可读写文本数据：

```
PROGRAM TEMPLATE

  IMPLICIT NONE

  INTEGER       ::  i

  OPEN(51, file="test.txt", form="formatted")

  WRITE(51, '(5i3)') (i,i=1,5)
  WRITE(51, '(5i3)') (i,i=6,10)

  CLOSE(51)

END PROGRAM TEMPLATE
```

输出文件可以用文本编辑器打开，用十六进制编辑器打开可以看到如下内容：

| Offset | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00000000 | 20 | 20 | 31 | 20 | 20 | 32 | 20 | 20 | 33 | 20 | 20 | 34 | 20 | 20 | 35 | 0A | 1 2 3 4 5 |
| 00000016 | 20 | 20 | 36 | 20 | 20 | 37 | 20 | 20 | 38 | 20 | 20 | 39 | 20 | 31 | 30 | 0A | 6 7 8 9 10 |

使用下面的 NCL 代码读取刚才生成的 test.txt 文件：

```
; NCL 脚本

load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/contributed.ncl"

begin

  fiASC = "./test.txt"
  nrow = numAsciiRow(fiASC)
  ncol = numAsciiCol(fiASC)
```

```
  data = asciiread(fiASC, (/nrow, ncol/), "integer")

  print(data)

end
```

## 2.3. NetCDF

NetCDF (Network Common Data Form)是由美国大学大气研究协会(UCAR)下的 Unidata 项目科学家针对科学数据的特点，提出的一种面向数组型数据、适于网络共享的数据描述和编码标准。已被国内外许多行业和组织采用，目前广泛应用于大气科学、水文、海洋学、环境模拟、地球物理等诸多领域。NetCDF 数据具有自描述性(普通二进制数据需要有相关文档介绍数据格式，否则无法正确读出数据)，数据与硬件平台无关(不用考虑数据的端序)。

目前采用 NetCDF 格式的资料主要为再分析资料、卫星资料、数值模式资料等。

每个 NetCDF 文件具备如下所示的结构，其中包含维数、变量、属性和数据 4 个子域，属性又分为适用于整个文件的全局属性和适用于特定变量的局部属性、

- dimensions(维)：一个维可以用来代表一个真实的物理空间、例如时间、纬度、经度或者高度。一个 NetCDF 的维有一个名字和长度，维的长度必须是一个正整数。
- variables(变量)：在 NetCDF 数据集中，变量是用来存放数据块的。NetCDF 数据集里的变量一般都是数组变量。一个变量代表着具有相同的数据类型的数组的值。每个变量都有一个名字、一个数据类型和数组形状。
- attributes(属性)：NetCDF 的属性是用来对数据进行辅助说明，存放关于数据的数据，例如变量的单位。
- data(数据)：NetCDF 支持的数据类型是 char、byte、short、int、float 或者 real、double。

```
dimensions:
        longitude = 240 ;
        latitude = 121 ;
        levelist = 37 ;
        time = UNLIMITED ; // (48 currently)
variables:
        float longitude(longitude) ;
                longitude:units = "degrees_east" ;
                longitude:long_name = "longitude" ;
        float latitude(latitude) ;
                latitude:units = "degrees_north" ;
                latitude:long_name = "latitude" ;
        int levelist(levelist) ;
                levelist:units = "millibars" ;
                levelist:long_name = "pressure_level" ;
        int time(time) ;
                time:units = "hours since 1900-01-01 00:00:0.0" ;
                time:long_name = "time" ;
                time:avg_period = "0000-01-00 00:00:00" ;
        short t(time, levelist, latitude, longitude) ;
                t:scale_factor = 0.00213156961327994 ;
                t:add_offset = 252.846295249234 ;
                t:_FillValue = -32767s ;
```

```
                t:missing_value = -32767s ;
                t:units = "K" ;
                t:long_name = "Temperature" ;

// global attributes:
                :Conventions = "CF-1.0" ;
                :history = "2011-03-18 02:08:23 GMT by mars2netcdf-0.92" ;
}
```

在实际处理时，需要多留心 short 类型的变量。为了减少存储数据所需的空间，往往会对实际物理量进行处理，将变量由 int、float 等转换为 short 型。由 short 变量得到原物理量时就需要用到"scale_factor"和"add_offset"属性，转换公式需要查看数据的说明文档。一般情况下有两种方法：

```
物理量 = 存储值 * scale_factor + add_offset
物理量 = (存储值 + add_offset) * scale_factor
```

NetCDF 库提供了创建和读取 NetCDF 文件的接口程序，也包括一些有用的工具，ncdump 程序可以用来查看 NetCDF 文件内容以及数据，在处理 NetCDF 数据时很有用：

```
$ ncdump  #查看程序的用法
```

```
$ ncdump -h surface-0.5d.nc  #查看文件内容
```

```
$ ncdump -v lon surface-0.5d.nc  # 查看变量 lon 的数值
```

使用 NCL 读取 NetCDF 数据很简单。使用其附带的"***ncl_filedump***"程序可以很方便地查看 NetCDF 文件的内容和数据(不仅可用于 NetCDF 格式，下面将要提到的 HDF 格式、GRIB 格式也可以用这个方法)。

```
$ ncl_filedump  # 查看程序的用法
```

```
$ ncl_filedump surface-0.5d.nc  # 查看文件内容
```

```
$ ncl_filedump -v lon surface-0.5d.nc  # 查看变量 lon
```

编写 NCL 脚本可以读取变量，并用做数据分析和绘图。

```
; NCL 脚本
begin

  finame = "./surface-0.5d.nc"
  fi = addfile(finame, "r")

  print(fi) ; 用于查看文件内容
; "ncl_filedump"程序本质上就是调用了 addfile 和 print 函数

  data = fi->landmask ; 读取变量

end
```

## 2.4. HDF、HDF-EOS、HDF5 和 HDF-EOS5

HDF (Hierarchical Data Format)数据格式是美国伊利诺伊大学国家超级计算应用中心

(NCSA, National Central for Super computing Applications)于 1987 年研制开发的一种软件和函数库，是一种具有自描述性、可扩展性、自我组织性的数据存储格式。

1993 年美国国家航空航天局(NASA)把 HDF 格式作为存储和发布 EOS(Earth Observation System)数据的标准格式。在 HDF 标准基础上，开发了另一种 HDF 格式即 HDF-EOS(或称为 HDF-EOS2)，专门用于处理 EOS 产品，使用标准 HDF 数据类型定义了点、条带、栅格 3 种特殊数据类型，并引入了元数据(Metadata)。HDF-EOS 是 HDF 的扩展，它主要扩充了两项功能：一是提供了一种系统宽搜索服务方式，它能在没有读文件本身的情况下搜索文件内容；二是提供了有效的存储地理定位数据，将科学数据与地理点捆绑在一起。

HDF5 则是为了充分利用当今计算机系统的能力和特点，克服 HDF4 的不足，为了满足科学数据存储不断增加而设计的文件格式和库文件。HDF5 支持管理超过 2GB 大小的文件，还支持并行 I/O。类似的，也有一个基于 HDF5 标准开发的 HDF-EOS5 标准。HDF4 和 HDF5 标准互不兼容。

HDF、HDF-EOS、HDF5 和 HDF-EOS5 数据多用于卫星资料的存储和发布。

目前 NCL 支持的 HDF 格式的数据类型包括 Scientific Data Sets (SDS)、GRID 和 SWATH，读取这些数据的方法与读取 NetCDF 格式数据的方法类似，这里不再赘述。如果需要读取 Vdata Set 等数据类型，可以借助 IDL 等工具。

## 2.5. GRIB 和 GRIB2

GRIB (GRIdded Binary)和 GRIB2 (General Regularly-distributed Information in Binary Form)是与计算机无关的压缩的二进制编码，主要用来存放数值天气分析和预报产品资料，由世界气象组织基本系统委员会(The World Meteorological Organization Commission for Basic System，WMO/CBS)制订修改。

目前数值模式产品、再分析资料等多使用 GRIB 及 GRIB2 格式存储。

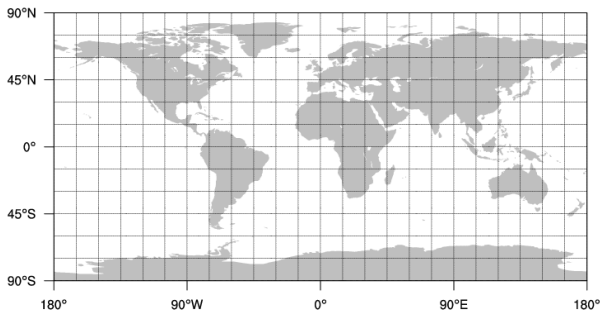
# 3. 数据可视化

## 3.1. 变量与经纬度

气象资料往往与地理坐标有密切的关系。地理坐标信息、变量的排序方式不同，绘图时用到的方法也不同。常用的数据一般有如下几种：

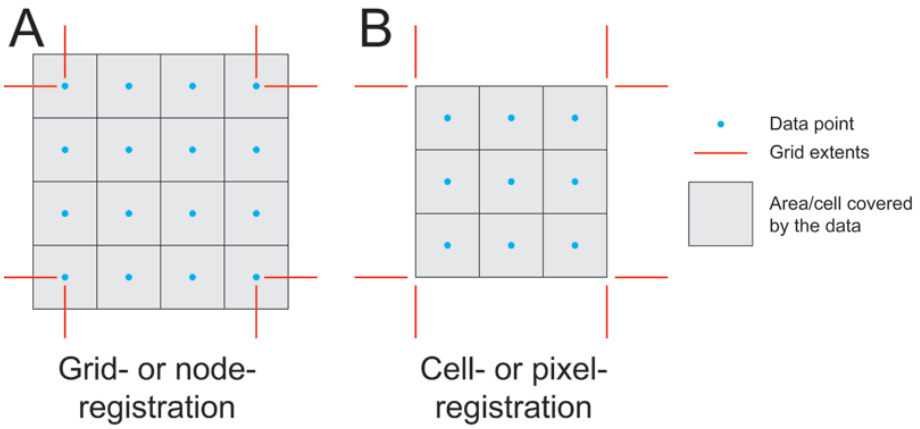
A. 变量水平方向为 2D 数据，地理坐标为 1D 数据

数据采用网格存储，为了找到某个点的经纬度，只需要知道这个点在网格中是行列位置即可。常用的全球数值模式产品、全球范围的分析场产品、麦卡托投影(Mercator projection)的区域模式产品、L3 级的卫星资料等都是这种类型。




使用这类资料时，需要注意：

- 经度是从 0°开始还是从西半球开始
- 格点采用的是普通经纬度网格还是高斯网格
- 纬度的维数大小是奇数还是偶数，这与网格经纬度有关，如下图所示



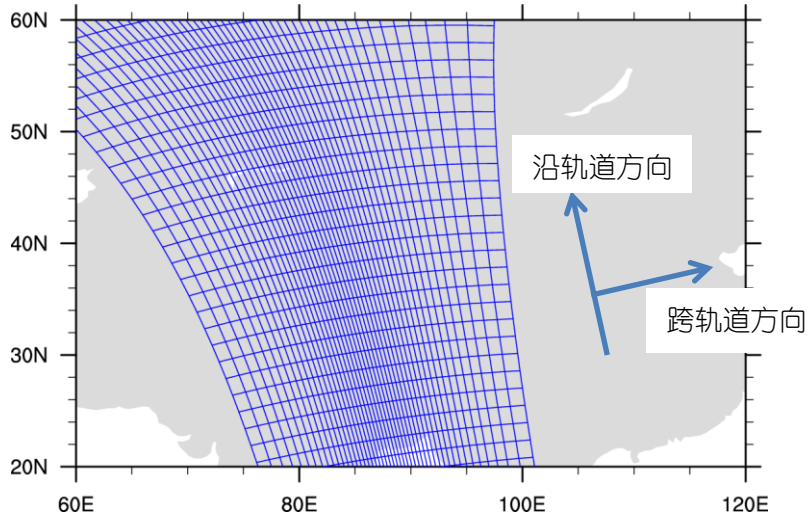

一般情况下，全球范围的模式产品纬度是奇数，比如 1°x1°的数据(360x181 格点)对应的经纬度范围是 0°至 359°，-90°至 90°；L3 的卫星数据纬度是偶数(360x180 格点)，经度是 0.5°至 359.5°，纬度是-89.5°至 89.5°。

使用 NCL 绘制这类数据时，需要用下面的方法附加上经纬度信息：

```
; NCL 脚本，这里的 pre 变量格点数为 181x360 (NCL 的数组表达方式与 C 类似)

  lon = ispan(0, 359, 1) ; 使用一个 1D 数组指定经度
  lon@units = "degrees_east"
  lat = ispan(-90, 90, 1) ; 使用一个 1D 数组指定纬度
  lat@units = "degrees_north"

  pre!0 = "lat" ; 给数组维度命名
  pre!1 = "lon"
  pre&lat = lat ; 给数组维度附加坐标信息
  pre&lon = lon
```

B. 变量水平方向为 2D 数据，地理坐标为 2D 数据

这种变量与坐标的对应方式多见于 L1 和 L2 卫星产品，以及使用兰伯托投影(Lamberto projection)的模式产品中。下面给出的是卫星探测扫描线的示意图：




使用 NCL 处理这类数据时，可以用下面的方法给数据附加上经纬度信息：

```
; NCL 脚本
  fi = addfile("./foo.hdf", "r")

  lat = fi->Latitude
  lon = fi->Longitude

  data = fi->Pressure
  data@lat2d = lat
  data@lon2d = lon
```

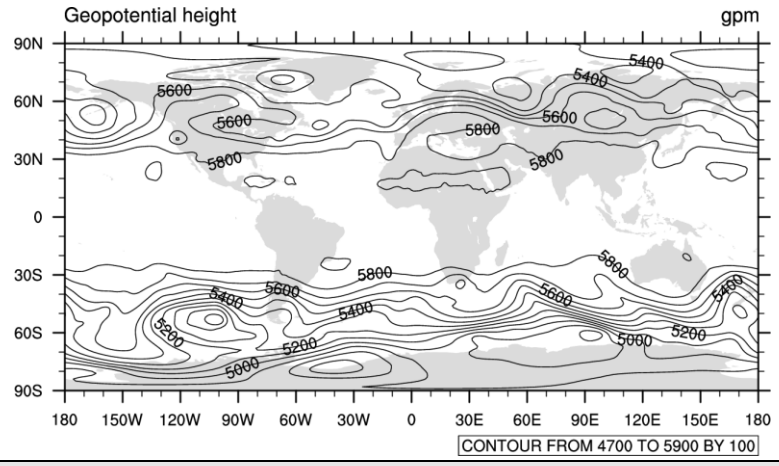
C. 变量水平方向为 1D 数据，地理坐标为 1D 数据

这种对应方式多见于站点数据，上面提到的 B 类数据将 2D 数据变成 1D 数据后，也可以像这种数据一样进行处理。

使用 NCL 绘制时，需要用下面的方法给数据附加经纬度信息：

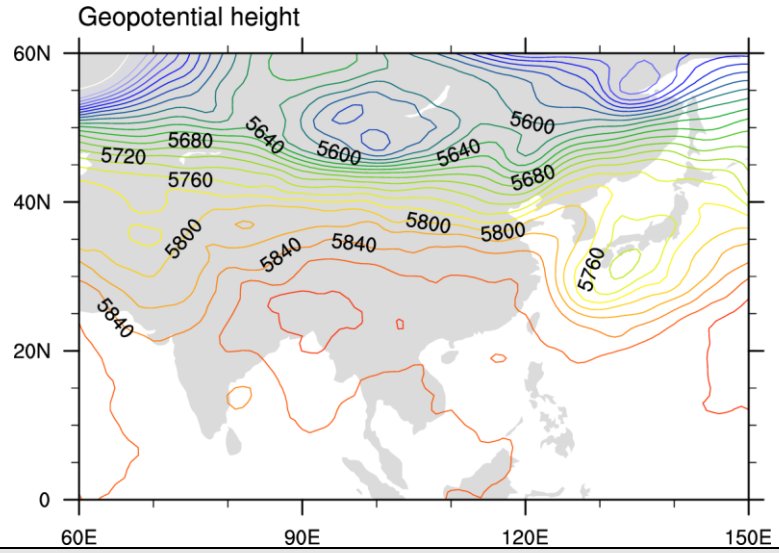
```
  data = fi->LandSeaMask

  wks = gsn_open_wks("ps", get_script_prefix_name())

  res = True

  res@...

  res@sfXArray = lon
  res@sfYArray = lat

  plot = gsn_csm_contour_map(wks, data, res)
```

# 3.2. 绘制等值线图



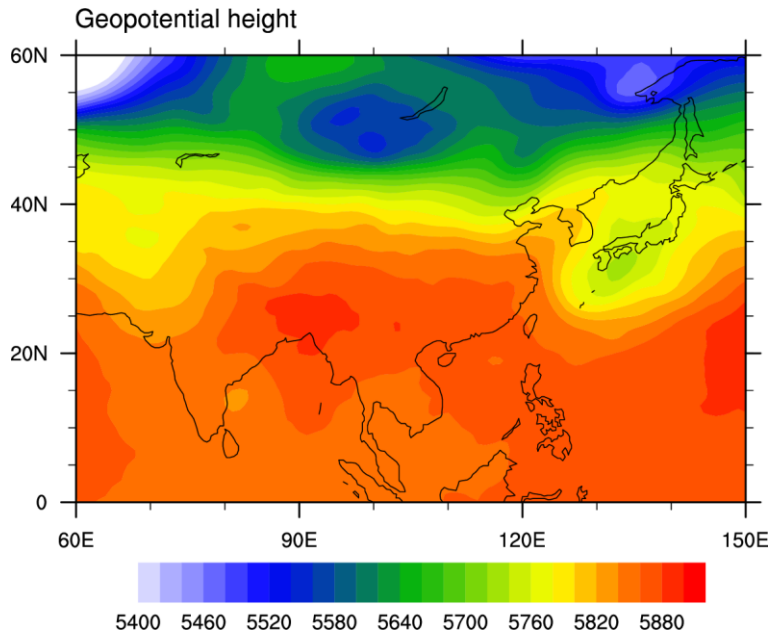

```
  load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
  load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_csm.ncl"

begin

  fi = addfile("./fnl_20050529_12_00.grb", "r")

  ; print(fi) ; 用于查看文件内容，在知道文件内容后可以注释该行

  hgt = fi->HGT_3_ISBL ; 位势高度
  ; 数据的经纬度信息已经附加在变量的属性中，可以用 printVarSummary(hgt)查看

  wks = gsn_open_wks("ps", get_script_prefix_name())
  ; 输出 ps 格式图像，文件名由该脚本的文件名决定
  gsn_define_colormap(wks,"WhBlGrYeRe") ; 设置色表
  ; 6.1.0 以后的 NCL 可以用另外的方法控制颜色

  res = True
; 绘制 500hPa 高度的位势高度
  plot = gsn_csm_contour_map(wks, hgt({500},:,:), res)

end
```

更改绘图设置：




```
  load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
  load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_csm.ncl"

begin

  fi = addfile("./fnl_20050529_12_00.grb", "r")

  ; print(fi) ; 用于查看文件内容，在知道文件内容后可以注释该行

  hgt = fi->HGT_3_ISBL

  wks = gsn_open_wks("ps", get_script_prefix_name())
  gsn_define_colormap(wks,"WhBlGrYeRe") ; 设置色表

  res = True

  res@cnMonoLineColor = False ; 彩色等值线
  res@cnInfoLabelOn = False ; 去掉右下角的说明
  res@gsnSpreadColors = True ; 6.1.0 后默认即为 True
  ; 设置地图范围
  res@mpMinLatF = 0
  res@mpMaxLatF = 60
  res@mpMinLonF = 60
  res@mpMaxLonF = 150
  ; 设置等值线起止、间隔
  res@cnLevelSelectionMode = "ManualLevels"
  res@cnMinLevelValF = 5400
  res@cnMaxLevelValF = 5900
  res@cnLevelSpacingF = 20

  ; 去掉右上角的文字
  res@gsnRightString = ""
  plot = gsn_csm_contour_map(wks, hgt({500},:,:), res)

end
```

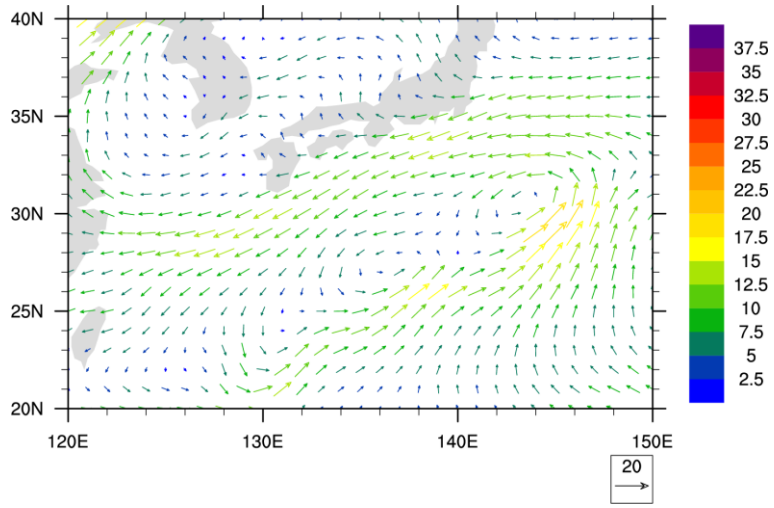
## 3.3. 绘制填色图




```
  load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
  load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_csm.ncl"

begin

  fi = addfile("./fnl_20050529_12_00.grb", "r")

  ; print(fi) ; 用于查看文件内容，在知道文件内容后可以注释该行

  hgt = fi->HGT_3_ISBL

  wks = gsn_open_wks("ps", get_script_prefix_name())
  gsn_define_colormap(wks,"WhBlGrYeRe") ; 设置色表

  res = True

  res@cnFillOn = True ; 填色
  res@cnLinesOn = False ; 去掉等值线
  res@cnLineLabelsOn = False ; 去掉等值线的数字
  res@cnInfoLabelOn = False ; 去掉右下角说明

  res@gsnSpreadColors = True ; 6.1.0 后默认即为 True

  ; 设置地图范围
  res@mpMinLatF = 0
  res@mpMaxLatF = 60
  res@mpMinLonF = 60
  res@mpMaxLonF = 150

  ; 设置等值线起止、间隔
  res@cnLevelSelectionMode = "ManualLevels"
  res@cnMinLevelValF = 5400
  res@cnMaxLevelValF = 5900
```

```
  res@cnLevelSpacingF = 20

  ; 设置 Label Bar
  res@lbBoxLinesOn = False ; Label bar 上不标记黑线

  ; 去掉右上角的文字
  res@gsnRightString = ""
  plot = gsn_csm_contour_map(wks, hgt({500},:,:), res)

end
```

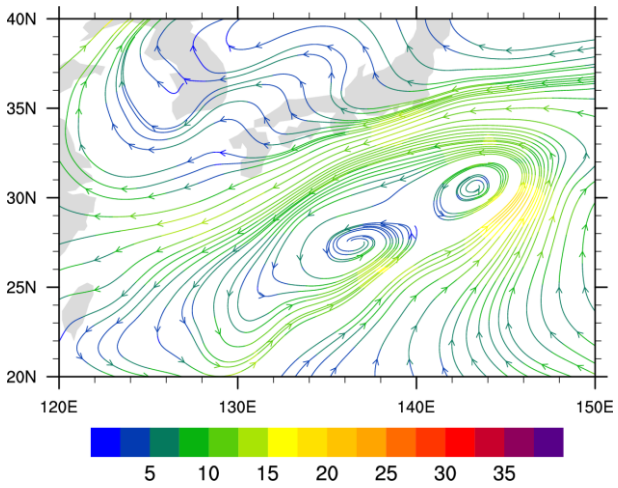
# 3.4. 绘制矢量箭头




```
  load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
  load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_csm.ncl"

begin

  fi = addfile("./fnl_20050529_12_00.grb", "r")

  u = fi->U_GRD_3_ISBL
  v = fi->V_GRD_3_ISBL

  wks = gsn_open_wks("ps", get_script_prefix_name())
  gsn_define_colormap(wks,"BlGrYeOrReVi200") ; 设置色表

  res = True

  res@vcMonoLineArrowColor = False ; 彩色箭头
  res@vcRefLengthF = 0.04 ; 单位长度箭头的长度
  res@vcRefMagnitudeF = 20 ; 单位长度箭头的数值
  res@vcRefAnnoArrowUseVecColor = False ; 右下角的图例中的箭头保持原来的颜色

  res@gsnSpreadColors = True ; 6.1.0 后默认即为 True

  ; 设置地图范围
  res@mpMinLatF = 20
  res@mpMaxLatF = 40
  res@mpMinLonF = 120
  res@mpMaxLonF = 150

  ; 设置 Label Bar
  res@lbOrientation = "Vertical" ; 竖直放置 Laber bar
  res@lbBoxLinesOn = False ; Label bar 上不标记黑线

  res@gsnRightString = "" ; 去掉右上角的文字
  res@gsnLeftString = "" ; 去掉左上角的文字

  plot = gsn_csm_vector_map(wks, u({850},:,:), v({850},:,:), res)
```

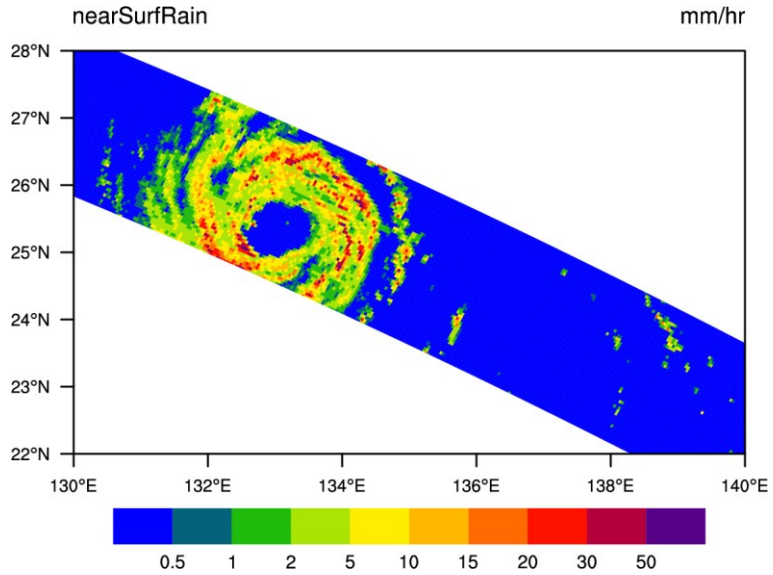
```
end
```

## 3.5. 绘制流线




```
  load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
  load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_csm.ncl"

begin

  fi = addfile("./fnl_20050529_12_00.grb", "r")

  u = fi->U_GRD_3_ISBL
  v = fi->V_GRD_3_ISBL

  wks = gsn_open_wks("ps", get_script_prefix_name())
  gsn_define_colormap(wks,"BlGrYeOrReVi200") ; 设置色表

  res = True

  res@stMonoLineColor = False
  res@stArrowLengthF = 0.008 ; 流线箭头的大小
  ; 设置地图范围
  res@mpMinLatF = 20
  res@mpMaxLatF = 40
  res@mpMinLonF = 120
  res@mpMaxLonF = 150
  ; 设置 Label Bar
  res@lbBoxLinesOn = False ; Label bar 上不标记黑线

  res@gsnRightString = "" ; 去掉右上角的文字
  res@gsnLeftString = "" ; 去掉左上角的文字

  plot = gsn_csm_streamline_map(wks, u({850},:,:), v({850},:,:), res)

end
```

# 3.6. 绘制卫星数据




```
  load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
  load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_csm.ncl"

begin

  fi = addfile("./2A25.20020903.27378.7.HDF", "r")
  lat = fi->Latitude
  lon = fi->Longitude

  rain = fi->nearSurfRain
  rain@_FillValue = -9999.

  rain@lat2d = lat
  rain@lon2d = lon

  wks = gsn_open_wks("ps", get_script_prefix_name())
  gsn_define_colormap(wks, "BlGrYeOrReVi200")

  res = True

  res@cnFillOn = True
  res@cnLinesOn = False
  res@cnLineLabelsOn = False
  res@cnInfoLabelOn = False

  res@cnFillMode = "CellFill" ; 可以加快绘图速度

  res@mpMinLatF = 20
  res@mpMaxLatF = 30
  res@mpMinLonF = 130
  res@mpMaxLonF = 140

  res@pmTickMarkDisplayMode = "Always" ; 默认是每隔 5°出现一次坐标数字
```

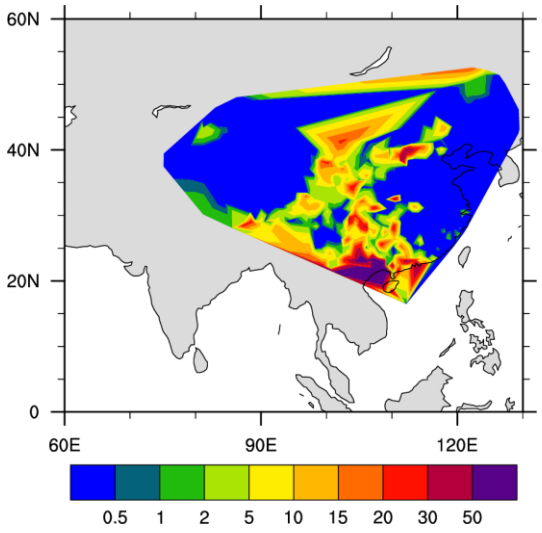
```
  res@gsnSpreadColors = True

  ; 设置等值线数值
  res@cnLevelSelectionMode = "ExplicitLevels"
  res@cnLevels = (/0.5, 1, 2, 5, 10, 15, 20, 30, 50/)

  ; 设置 Label Bar
  res@lbBoxLinesOn = False ; Label bar 上不标记黑线

  plot = gsn_csm_contour_map(wks, rain, res)

end
```

# 3.7. 绘制站点数据

NCL 绘制站点数据时，会采用三角网络(triangular mesh)的方法。比如有个文本文件存储的站点数据：

```
   43.42  130.15    0.00
   39.23  111.09   21.90
   38.57  113.31    0.00
   44.34  120.54    0.00
   42.11  116.28    5.00
   38.29  106.13    0.80
   43.59  119.24    0.00
…省略…
```

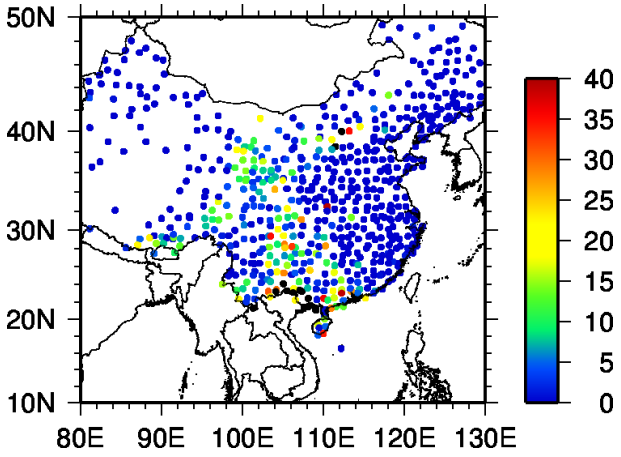
绘图结果如下，惨不忍睹。




```
  load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
  load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_csm.ncl"

begin

  fidat = "./rainOut.txt"
  nrow = numAsciiRow(fidat)

  data = asciiread(fidat, (/nrow, 3/), "float")
  lat = data(:,0)
  lon = data(:,1)
  rain = data(:,2)

  wks = gsn_open_wks("ps", get_script_prefix_name())
  gsn_define_colormap(wks, "BlGrYeOrReVi200")

  res = True

  res@cnFillOn = True
  res@cnLinesOn = False
  res@cnLineLabelsOn = False
  res@cnInfoLabelOn = False

  res@cnLevelSelectionMode = "ExplicitLevels"
  res@cnLevels = (/0.5, 1, 2, 5, 10, 15, 20, 30, 50/)
```

```
  res@trGridType = "TriangularMesh"
  res@gsnSpreadColors = True

  res@mpMinLatF = 0
  res@mpMaxLatF = 60
  res@mpMinLonF = 60
  res@mpMaxLonF = 130

  res@sfXArray = lon
  res@sfYArray = lat

  plot = gsn_csm_contour_map(wks, rain, res)

end
```

NCL 绘制这类数据比较难看，虽然可以用其他的方法代替，比如先画好地图，然后以 marker 形式将数据用彩色点标注在地图上，官网例子 station_2.ncl 就是采用的这样的方法(http://www.ncl.ucar.edu/Applications/station.shtml)，不过这种绘图速度很慢。遇到这种数据，可以借助 GMT 进行绘图。GMT 提供了若干 Linux 命令行工具，通过设置命令参数进行绘图的设置。这些参数对初学者来说与乱码无异。但也值得花点时间了解 GMT，用来辅助其它工具进行绘图。比如使用下面 Bash 脚本绘制站点数据特别合适：




```
#! /bin/bash
#

psfile=Rain.ps

# PLOT_DEGREE_FORMAT 设置地图坐标标记
gmtset PLOT_DEGREE_FORMAT F
gmtset DEGREE_SYMBOL none
gmtset BASEMAP_TYPE plain

# 字体大小设置
gmtset ANNOT_FONT_SIZE_PRIMARY 12

makecpt -Cseis -I -T0/40/1 -Z > Rain.cpt

# 绘制底图（空白）
```

```
psbasemap -R80/130/10/50 -Jm0.05i -B10f2WSne -X1i -Y6.5i -P -K > $psfile
# 绘制降水
psxy rainOut.txt -: -R -Jm -CRain.cpt -Sc0.04i -O -K >> $psfile
# 绘制海岸线
pscoast -R -Jm -A0/0/1 -Di -N1 -W1/0.2p -O -K >> $psfile
# 绘制色标
psscale -D2.75i/1i/2i/0.2i -CRain.cpt -B5 -O -K >> $psfile
# 这里添加的文字是按照地图坐标的位置添加的
echo "130.5 58.5 18 0 0 TL (a)" | pstext -R -Jm -O -K >> $psfile

# 清理临时文件
rm -f .gmt*
```

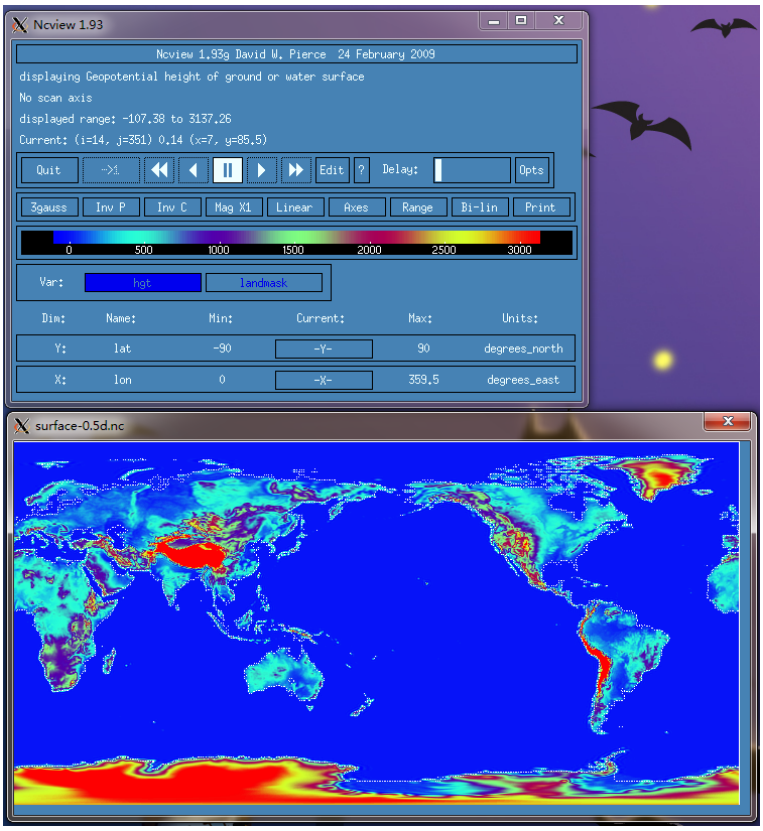
# 3.8. 看图工具

这里介绍的工具主要用于浏览数据，而不是用来绘制出版物所用的图像。

**ncview**

http://meteora.ucsd.edu/~pierce/ncview_home_page.html

这个工具主要用于 NetCDF 数据的可视化，尤其适用于格点数据。目前最新版本为 2.1.1 (Released 1 August, 2011)，支持 NetCDF4 格式。如果平时较少接触 NetCDF4 格式，可以下载 1.9.3 版本(ftp://cirrus.ucsd.edu/pub/ncview/ncview-1.93g.tar.gz)，这个版本的 ncview 编译也更简单。



# 附录：

## NCL 的安装

NCL 的安装可以参考 http://www.ncl.ucar.edu/Download/install.shtml ，大致步骤为：

a. 新建目录，解压缩 NCL 软件包

```
$ mkdir /home/usr/NCL/
$ cd /home/usr/NCL/
$ tar zxf ncl_ncarg-6.1.0.linux_redhat_x86_64_nodap_gcc444.tar.gz
```

b. 设置环境变量

```
$ export NCARG_ROOT=/home/usr/NCL/
$ export PATH=$PATH:$NCARG_ROOT/bin
```

正确安装后即可运行程序"ncl"，然后可以将环境变量的设置写入.bashrc 文件。运行时，只需要在命令行中输入：

```
$ ncl foo.ncl  # foo.ncl 为脚本文件名
```

## NetCDF 库的安装

NetCDF 库源文件可以在 http://www.unidata.ucar.edu/downloads/netcdf/index.jsp 下载。Linux 下的 NetCDF 库的编译和安装步骤为：

a. 解压缩

```
$ tar zxf netcdf-4.0.1.tar.gz
```

b. 设置编译选项

```
$ ./configure --prefix=/home/usr/netcdf
```

由于软件设置的默认安装路径需要管理员权限，这里通过"--prefix"参数修改安装路径

c. 编译和安装

```
$ make install
```

d. 设置环境变量

```
$ export PATH=$PATH:/home/usr/netcdf/bin
```

这样就完成了安装，可以运行 ncdump 程序。如果一切正确，可以将这个设置内容添加到.bashrc 文件中。在编译完 NetCDF 库后，就可以使用 C 语言和 FORTRAN 语言调用相应的函数来读取和创建 NetCDF 文件的数据了。相关操作比较复杂，这里不做介绍。

## GMT 的编译

GMT 的编译需要预先编译 NetCDF 库，按照前面的步骤编译完成 NetCDF 并设置好环境变量后，可以按照下面的方法对 GMT 进行编译：

a. 解压缩

b. 设置编译选项，指定 NetCDF 库的路径

```
$ ./configure --prefix=/home/usr/GMT \
    NETCDF_INC=/home/usr/netcdf/include \
    NETCDF_LIB=/home/usr/netcdf/lib
```

如果安装 NetCDF 时使用超级用户直接安装在默认路径中，可以不用额外设定 NetCDF 库的位置

c. 编译安装

```
$ make install-all
```

d. 设置环境变量

```
$ export PATH=$PATH:/home/usr/GMT/bin
```

正确完成后，可以运行"psxy"、"psxyz"等 GMT 工具，为了方便以后使用，可以把环境变量的设置添加到.bashrc 中。

下载分流链接：
http://pan.baidu.com/share/link?shareid=121761&uk=1845531131

# 参考资料

NetCDF 网站：http://www.unidata.ucar.edu/software/netcdf/
NCL 网站：http://www.ncl.ucar.edu/index.shtml
GMT 网站：http://gmt.soest.hawaii.edu/
GrADS 网站：http://grads.iges.org/grads/grads.html
GDL 网站：http://gnudatalanguage.sourceforge.net/
SciLab 网站：www.scilab.org
SciPY 网站：www.scipy.org
ncview 网站：http://meteora.ucsd.edu/~pierce/ncview_home_page.html
张林，高玉春，杨金红，杨洪平. 基于VC++平台的相控阵天气雷达NetCDF数据读取与产品显示[J]. 气象科技 , 2010,(02)
刘媛媛，应显勋，赵芳. GRIB2 介绍及解码初探[J]. 气象科技 , 2006,(S1)