

4.3 绘制二维图形

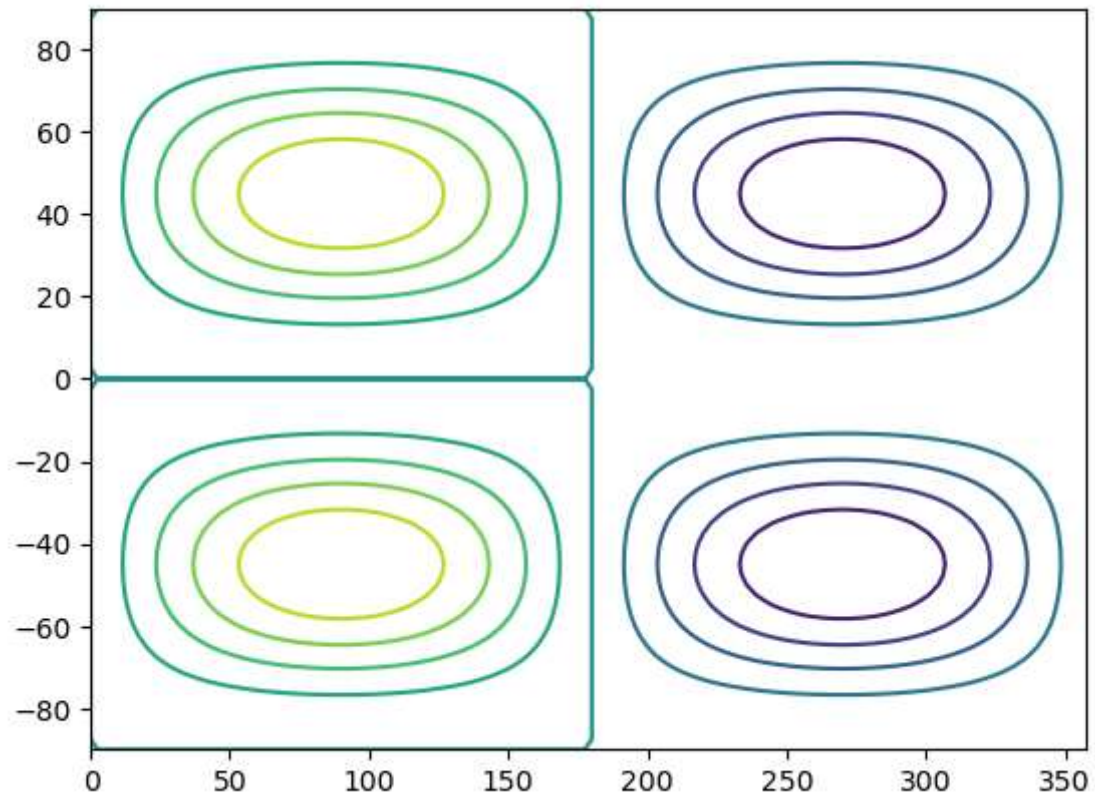
Markdown by 郭子悦

本节主要介绍二维图形的绘制，有用于表达数据空间分布的等值线、等值线填色图，还有用于表达流体运动大小和方向的矢量图。

a) 等值线

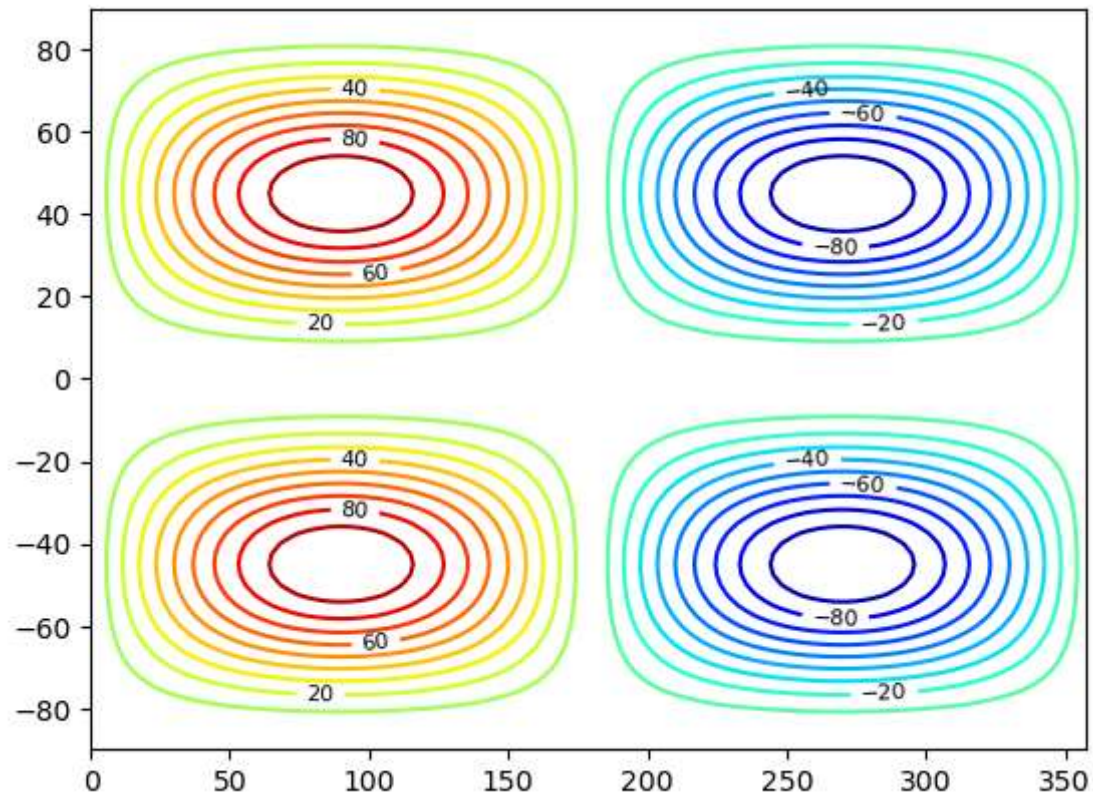
- 等值线用contour()函数绘制
 - 前2个参数分别是x和y轴的坐标
 - 第3个参数是(x, y)所在格点处数值的大小
 - 参数levels可以是一个整数，表明画几根等值线，也可以是一维序列表示要画出的等值线
 - 参数colors设置等值线的颜色为单一颜色，也可以是与levels相同个数的不同颜色（单色时负值设为虚线）
 - 参数cmap使用已有的或自定义的颜色表
 - 参数linewidths为线宽
- 使用clabel()函数为等值线标值
 - levels给出需标值等值线
 - colors标值的颜色
 - fontsize标值的大小

```
In [4]: import matplotlib.pyplot as plt
from netCDF4 import Dataset
f = Dataset('wave.grib2.nc')
dat = f.variables['gh'][:]
lon = f.variables['longitude'][:]
lat = f.variables['latitude'][:]
fig = plt.figure()
ax = fig.add_subplot(1,1,1)
ax.contour(lon,lat,dat[0],levels=11)
plt.show()
```



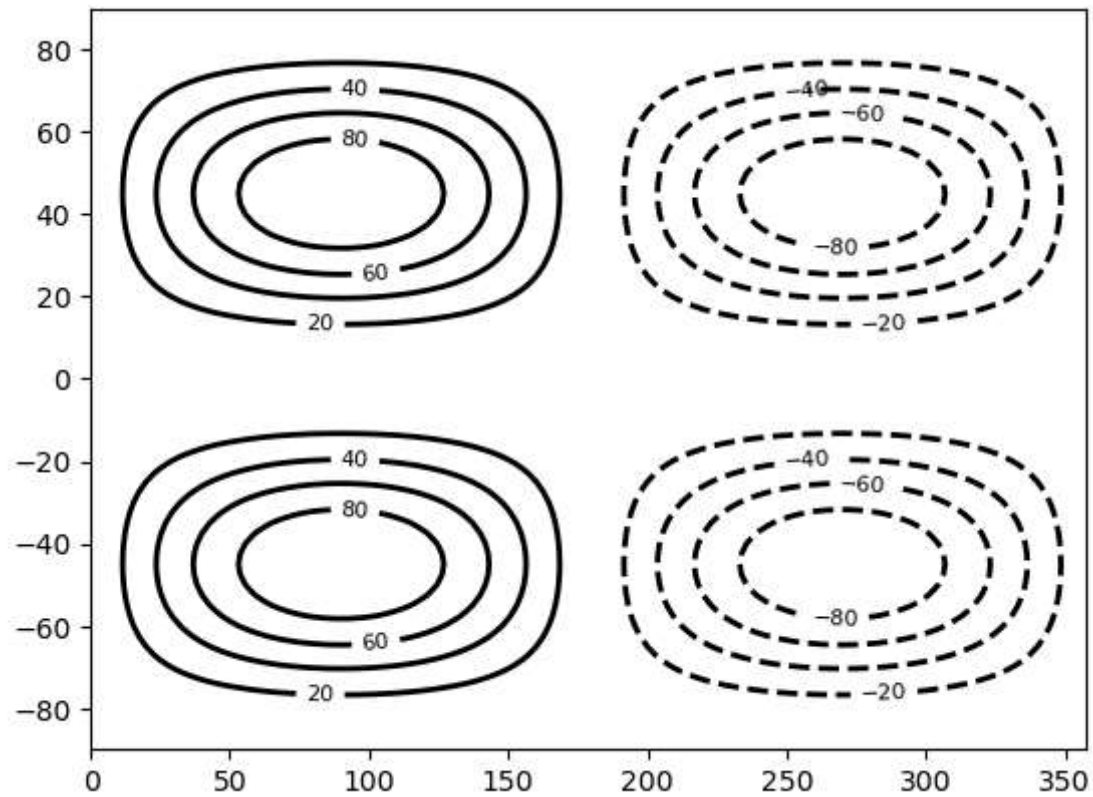
```
In [5]: fig = plt.figure()
ax = fig.add_subplot(1,1,1)
levels = list(range(-100,0,10))+list(range(10,101,10))
print(levels)
h = ax.contour(lon,lat,dlat[0],levels=levels,cmap='jet')
levels = list(range(-100,0,20))+list(range(20,101,20))
ax.clabel(h,levels=levels,fontsize=8,colors='k')
plt.show()
```

```
[-100, -90, -80, -70, -60, -50, -40, -30, -20, -10, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```



```
In [6]: fig = plt.figure()
ax = fig.add_subplot(1,1,1)
levels = list(range(-100,0,20))+list(range(20,101,20))
print(levels)
h = ax.contour(lon,lat,dlat[0],levels=levels,colors='k',linewidths=2)
levels = list(range(-100,0,20))+list(range(20,101,20))
ax.clabel(h,levels=levels,fontsize=8,colors='k')
plt.show()
```

```
[-100, -80, -60, -40, -20, 20, 40, 60, 80, 100]
```



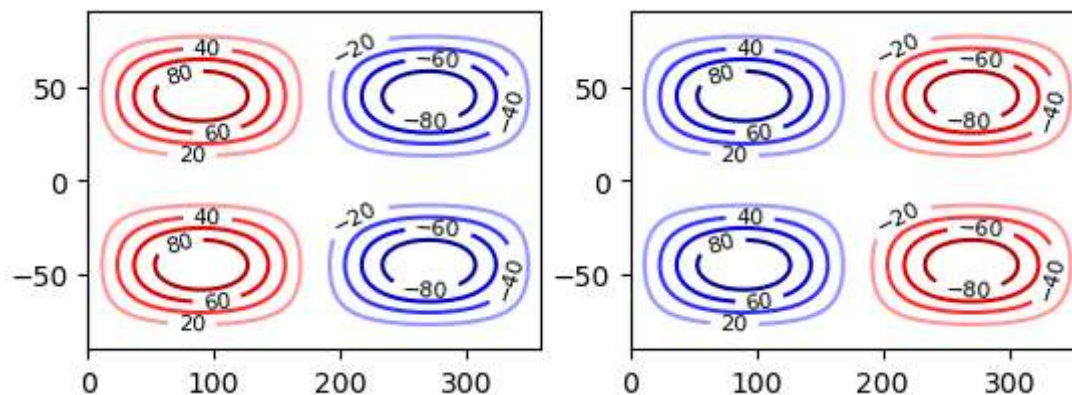
b) 颜色表

- 在contour中使用到cmap选项，matplotlib为其提供了丰富的颜色表，可用于所有带cmap参数的函数
- 颜色表为字符串，带"_r"的颜色表字符串表示取反
- 此外，还可以自定义颜色表

```
In [7]: from matplotlib.cm import datad  
print(datad.keys())
```

```
dict_keys(['Blues', 'BrBG', 'BuGn', 'BuPu', 'CMRmap', 'GnBu', 'Greens', 'Greys', 'OrRd', 'Oranges', 'PRGn', 'PiYG', 'PuBu', 'PuBuGn', 'PuOr', 'PuRd', 'Purples', 'RdBu', 'RdGy', 'RdPu', 'RdYlBu', 'RdYlGn', 'Reds', 'Spectral', 'Wistia', 'YlGn', 'YlGnBu', 'YlOrBr', 'YlOrRd', 'afmhot', 'autumn', 'binary', 'bone', 'brg', 'bwr', 'cool', 'coolwarm', 'copper', 'cubehelix', 'flag', 'gist_earth', 'gist_gray', 'gist_heat', 'gist_ncar', 'gist_rainbow', 'gist_stern', 'gist_yarg', 'gnuplot', 'gnuplot2', 'gray', 'hot', 'hsv', 'jet', 'nipy_spectral', 'ocean', 'pink', 'prism', 'rainbow', 'seismic', 'spring', 'summer', 'terrain', 'winter', 'Accent', 'Dark2', 'Paired', 'Pastel1', 'Pastel2', 'Set1', 'Set2', 'Set3', 'tab10', 'tab20', 'tab20b', 'tab20c'])
```

```
In [8]: fig = plt.figure()
        for i,s in enumerate(['','_r']):
            ax = fig.add_subplot(2,2,i+1)
            levels = list(range(-100,0,20))+list(range(20,101,20))
            h = ax.contour(lon,lat,dat[0],levels=levels,cmap='seismic'+s)
            levels = list(range(-100,0,20))+list(range(20,101,20))
            ax.clabel(h,levels=levels,fontsize=8,colors='k')
        plt.show()
```



- 自定义颜色表使用LinearSegmentedColormap函数
 - 第1个参数为新的colormap的名字
 - 第2个参数为颜色列表
 - 第3个参数为线性插出的颜色个数

```
In [9]: from matplotlib.colors import LinearSegmentedColormap

        self_define_colors = ['green','lime','w','pink','red']
        n_bin = 11
```

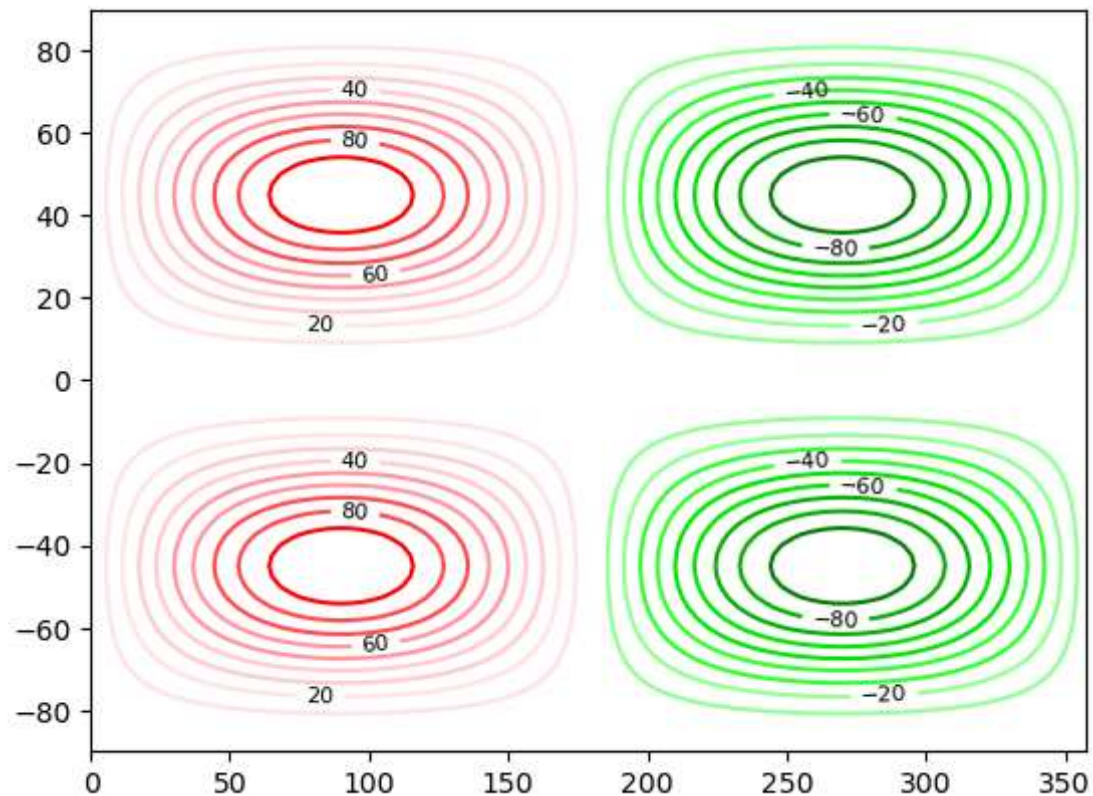


```

cmap = LinearSegmentedColormap.from_list('sdc',self_define_colors,N=n_bin)

fig = plt.figure()
ax = fig.add_subplot(1,1,1)
levels = list(range(-100,0,10))+list(range(10,101,10))
h = ax.contour(lon,lat,dat[0],levels=levels,cmap=cmap)
levels = list(range(-100,0,20))+list(range(20,101,20))
ax.clabel(h,levels=levels,fontsize=8,colors='k')
plt.show()

```



c) 二维填色图

利用伪彩色表示数值大小，可以使用

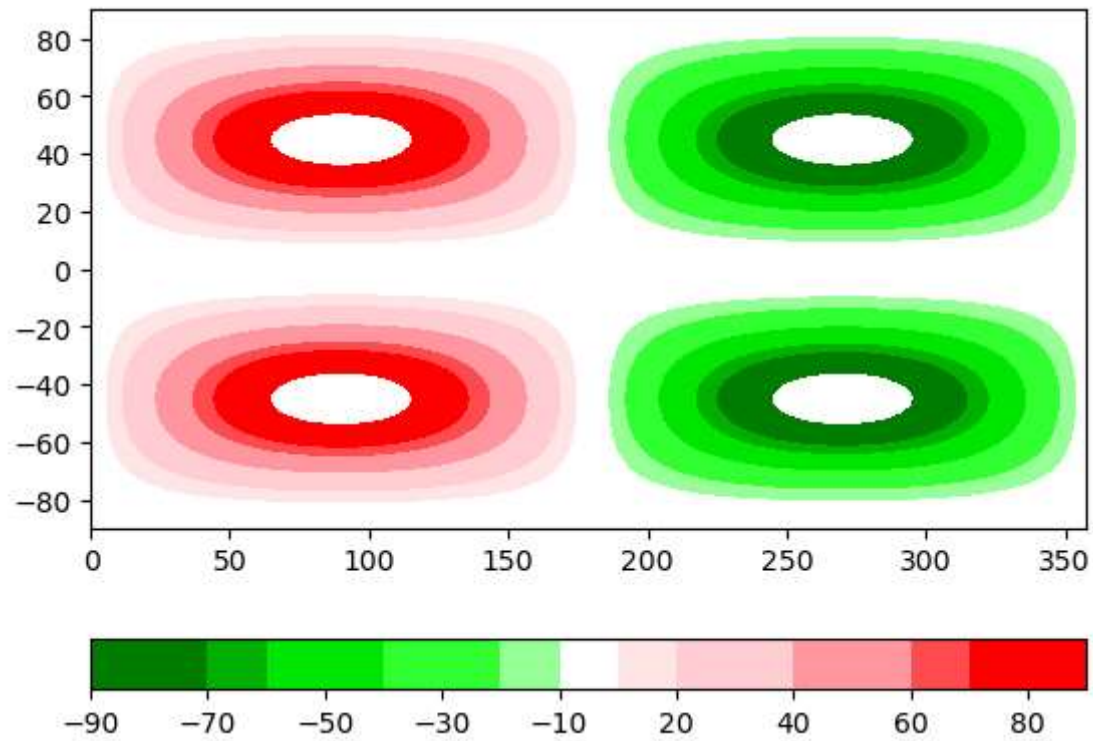
- contourf
- pcolor

- pcolormesh (较pcolor快)
- plt.colorbar 画色标
 - colorbar第1个参数为handles
 - cax用于多图共享colorbar
 - orientation有horizontal和vertical两个选项

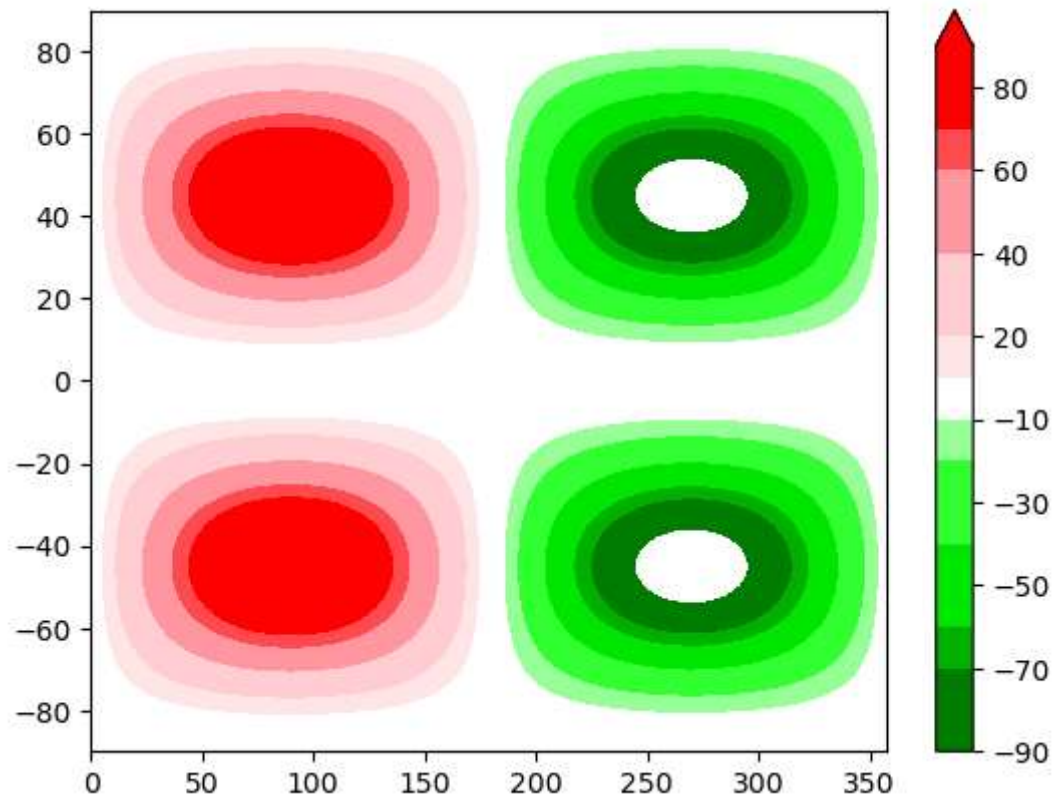
(1) contourf

- 前3个参数与contour一致
- levels, cmap亦同
- 使用到extend参数, 用于确定溢出的数据怎么画

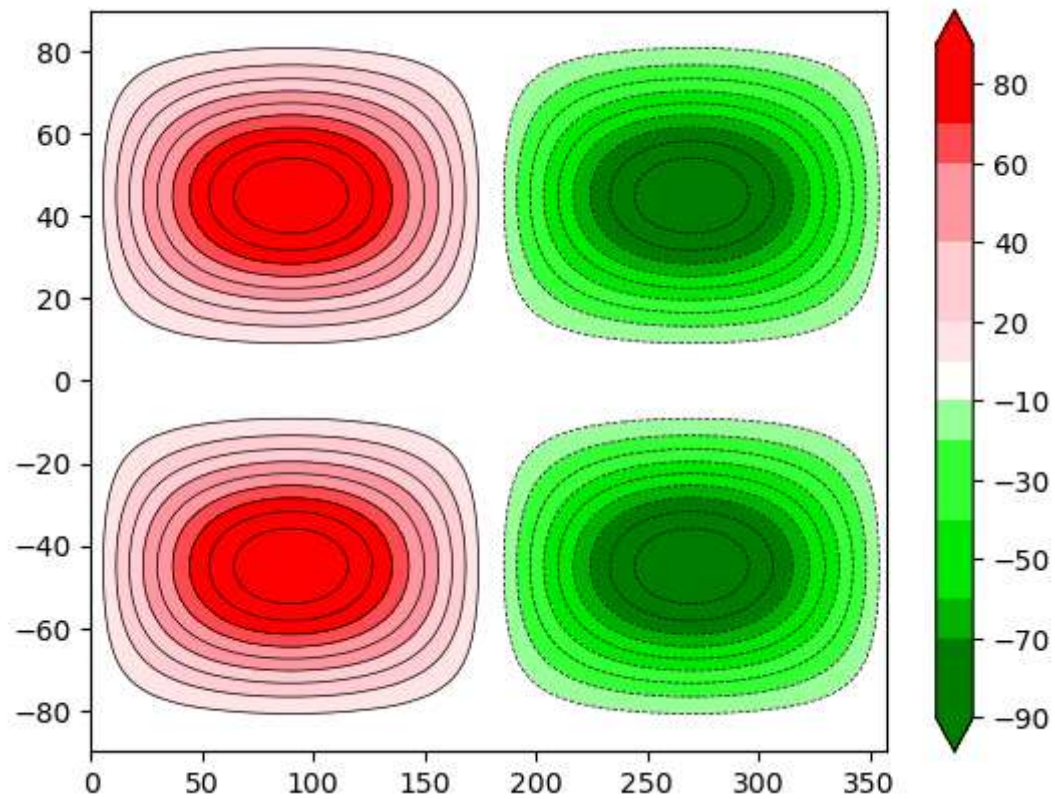
```
In [10]: fig = plt.figure()
ax = fig.add_subplot(1,1,1)
levels = list(range(-90,0,10))+list(range(10,91,10))
h = ax.contourf(lon,lat,dat[0],levels=levels,cmap=cmap)
plt.colorbar(h,orientation='horizontal')
plt.show()
```



```
In [11]: fig = plt.figure()
ax = fig.add_subplot(1,1,1)
levels = list(range(-90,0,10))+list(range(10,91,10))
h = ax.contourf(lon,lat,d1[0],levels=levels,cmap=cmap,
                extend='max')
plt.colorbar(h,orientation='vertical')
plt.show()
```

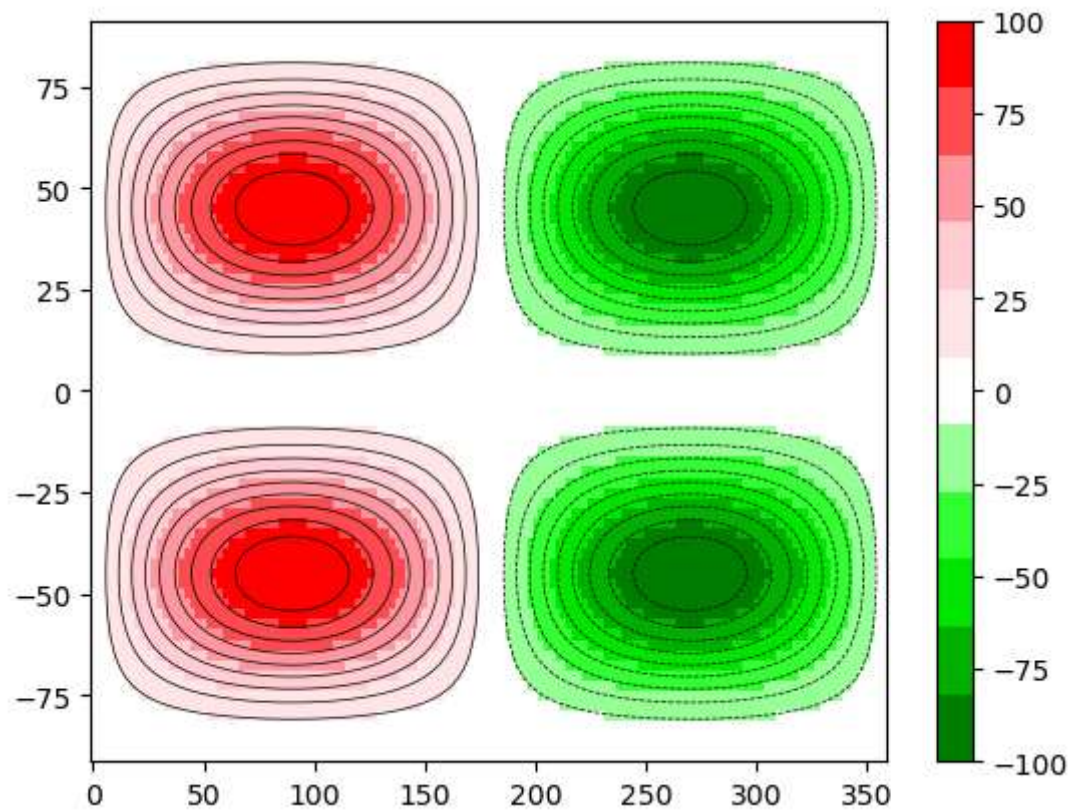
```
In [12]: fig = plt.figure()
ax = fig.add_subplot(1,1,1)
levels = list(range(-90,0,10))+list(range(10,91,10))
h = ax.contourf(lon,lat,dat[0],levels=levels,cmap=cmap,
                extend='both')
ax.contour(lon,lat,dat[0],levels=levels,colors='k',linewidths=0.5)
plt.colorbar(h,orientation='vertical')
plt.show()
```



(2) `pcolor`或`pcolormesh`

- 选项与`contourf`相似，但不使用`levels`和`extend`
- 数值的绘制范围由`vmin`和`vmax`来确定
- 填色的个数由`colormap`的颜色个数来确定

```
In [13]: fig = plt.figure()
ax = fig.add_subplot(1,1,1)
h = ax.pcolormesh(lon,lat,dat[0],cmap=cmap,vmin=-100,vmax=100)
levels = list(range(-90,0,10))+list(range(10,91,10))
ax.contour(lon,lat,dat[0],levels=levels,colors='k',linewidths=0.5)
plt.colorbar(h,orientation='vertical')
plt.show()
```



d) 矢量图

- 使用quiver绘制矢量图
- quiverkey绘制矢量参考

计算地转风绘制矢量图，使用：

- 中央差
 - $f(x) = \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x}$
- 前差
 - $f(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x}$
- 后差

$$\blacksquare f(x) = \frac{f(x) - f(x - \Delta x)}{\Delta x}$$

计算差分

```
In [14]: import numpy as np

def der(dat,dx=2.5*np.pi/180,axis=0):
    datp = np.zeros(dat.shape)
    if axis==0:
        datp[1:-1] = (dat[2:] - dat[:-2])/dx/2
        datp[0] = (dat[1] - dat[0])/dx
        datp[-1] = (dat[-1]-dat[-2])/dx
    else:
        datp[:,1:-1] = (dat[:,2:] - dat[:, :-2])/dx/2
        datp[:,0] = (dat[:,1] - dat[:,0])/dx
        datp[:, -1] = (dat[:, -1]-dat[:, -2])/dx
    return datp
```

- 球坐标系下地转风关系

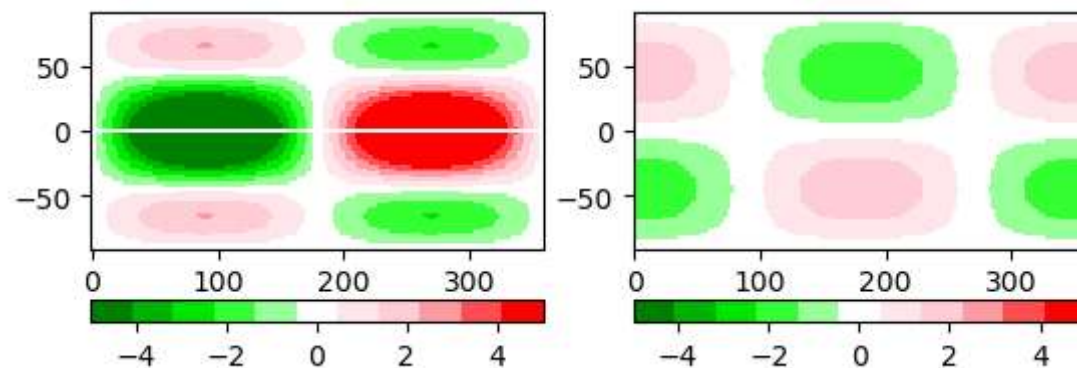
$$\blacksquare u = -\frac{g}{f} \frac{\partial h}{a \partial \varphi}$$

$$\blacksquare v = \frac{g}{f} \frac{\partial h}{a \cos \varphi \partial \lambda}$$

```
In [15]: def geostrophic_wind(lon,lat,gh):
    g = 9.8
    a = 6.378137e6
    omega = 2*np.pi/24/3600
    x, y = np.meshgrid(lon,lat)
    y = y/180*np.pi
    f = 2*omega*np.sin(y)
    msk = np.abs(f)<1e-10
    f[msk] = np.nan
    u = -1*g*der(gh,axis=0)/(f*a)
    v = g*der(gh,axis=1)/(f*a*np.cos(y))
    return u,v
```

```
In [20]: print(lat)
lat = lat[::-1]
dat = dat[:,::-1,:]
u, v = geostrophic_wind(lon,lat,dat[0])
fig = plt.figure()
for i, x in enumerate([u,v]):
    ax = fig.add_subplot(2,2,i+1)
    h = ax.pcolormesh(lon,lat,x,cmap=cmap,vmin=-5,vmax=5)
    plt.colorbar(h,orientation='horizontal')
plt.show()
```

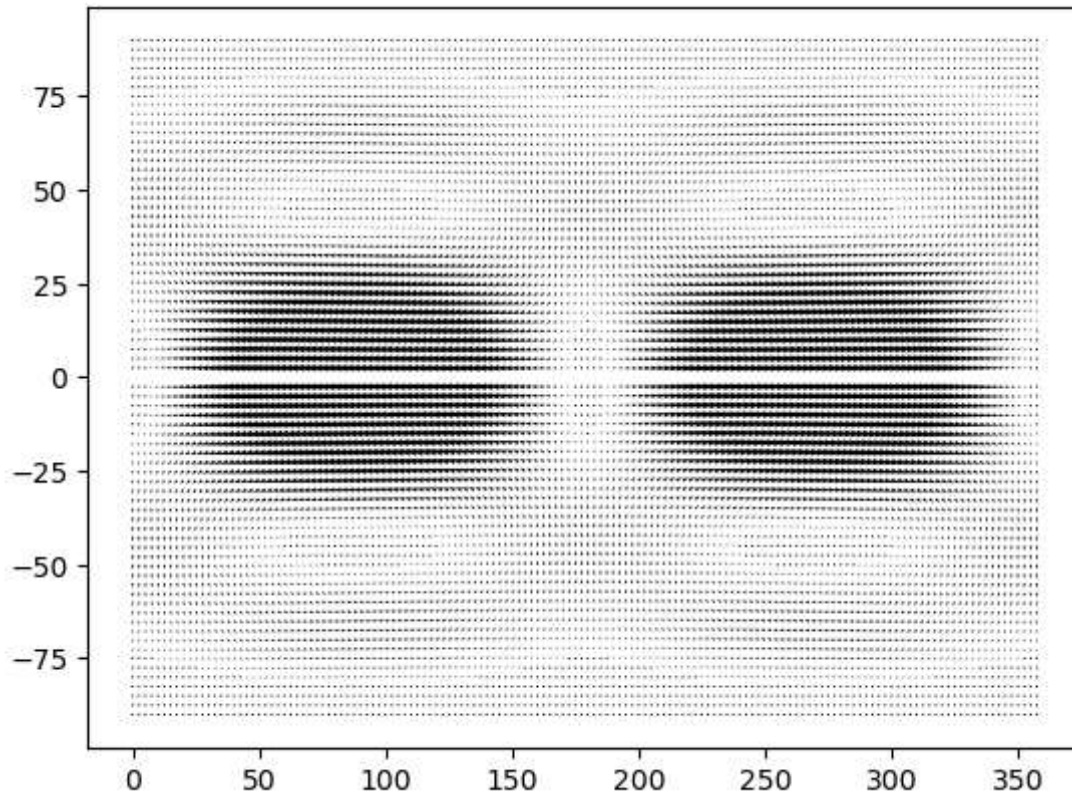
```
[ 90.  87.5  85.   82.5  80.   77.5  75.   72.5  70.   67.5  65.   62.5
 60.  57.5  55.   52.5  50.   47.5  45.   42.5  40.   37.5  35.   32.5
 30.  27.5  25.   22.5  20.   17.5  15.   12.5  10.    7.5   5.    2.5
  0.   -2.5  -5.   -7.5 -10.  -12.5 -15.  -17.5 -20.  -22.5 -25.  -27.5
-30.  -32.5 -35.  -37.5 -40.  -42.5 -45.  -47.5 -50.  -52.5 -55.  -57.5
-60.  -62.5 -65.  -67.5 -70.  -72.5 -75.  -77.5 -80.  -82.5 -85.  -87.5
-90. ]
```



- `quiver()`函数绘制矢量
 - 前4个参数分别为x和y轴的坐标，矢量在x和y方向的值
 - `scale`用来设置矢量大小，值越大矢量越小，反之则反
 - `units`用来设置矢量大小的参考，一般用width已够用
 - `pivot`设置矢量相对格点的位置，选mid即可
 - `color`为矢量颜色，若提供第5个参数可设置cmap用颜色表示第5个参数的大小
 - `width`表示矢量的宽度

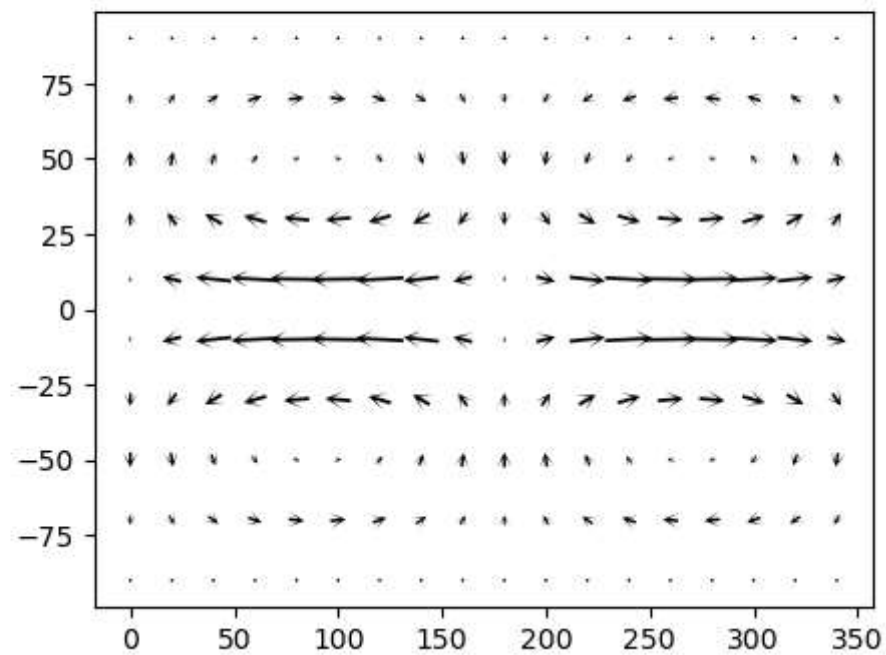
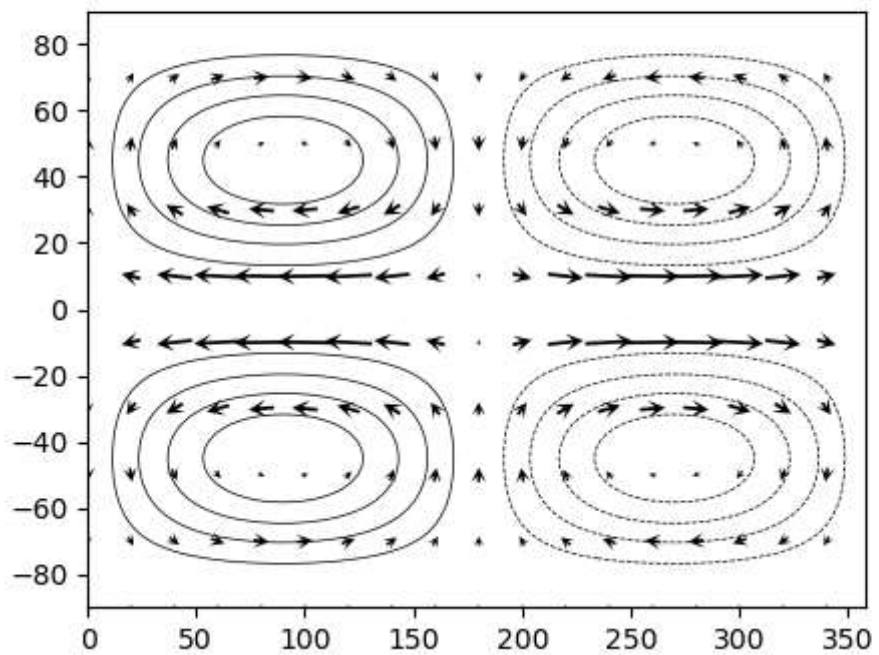
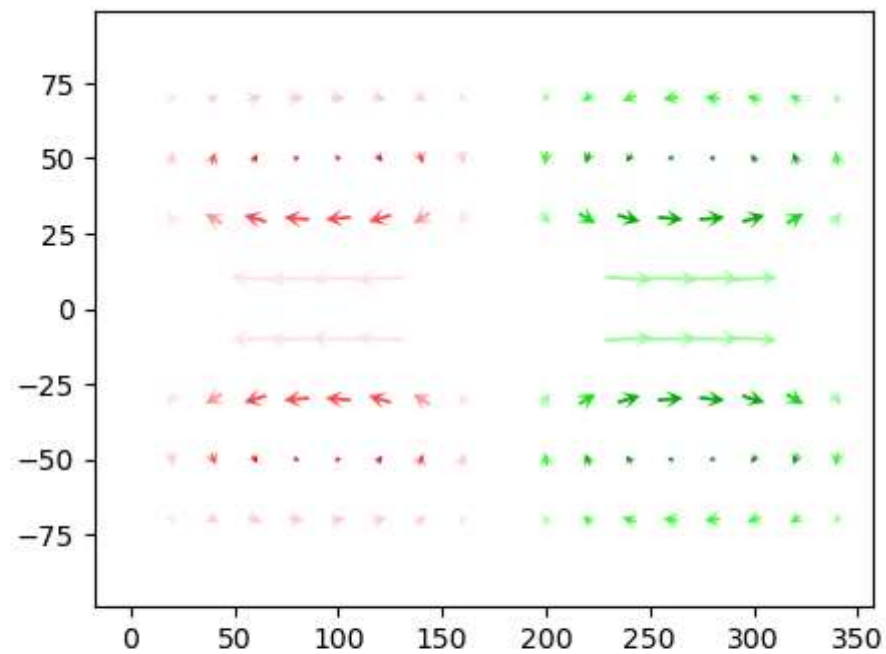
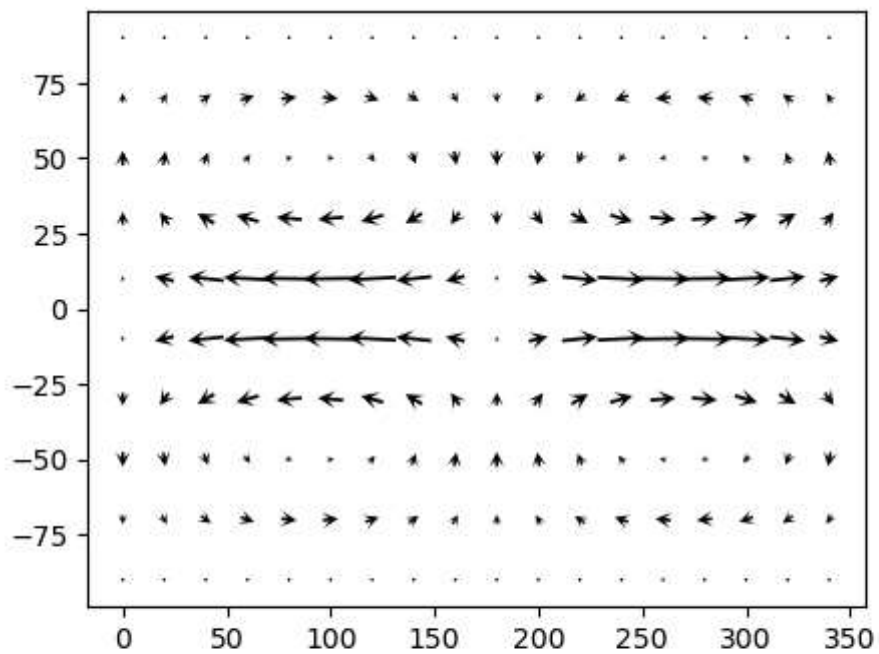
- headwidth矢量头大小
- headlength矢量头长度
- headaxislength矢量内侧的长度

```
In [21]: fig = plt.figure()  
ax = fig.add_subplot(1,1,1)  
ax.quiver(lon,lat,u,v)  
plt.show()
```



- 矢量太密需间隔取值或对数值升尺度
- 进一步设置使图美观


```
In [22]: fig = plt.figure(figsize=(11,8.5))
d = 8
ax = fig.add_subplot(2,2,1)
ax.quiver(lon[::d],lat[::d],u[::d,::d],v[::d,::d],
          scale=110,units='width',pivot='mid',color='k',
          headwidth=5,headlength=5,headaxislength=3)
ax = fig.add_subplot(2,2,2)
ax.quiver(lon[::d],lat[::d],u[::d,::d],v[::d,::d],dat[0,::d,::d],
          scale=110,units='width',pivot='mid',cmap=cmap,
          headwidth=5,headlength=5,headaxislength=3)
ax = fig.add_subplot(2,2,3)
ax.quiver(lon[::d],lat[::d],u[::d,::d],v[::d,::d],
          scale=110,units='width',pivot='mid',color='k',
          headwidth=5,headlength=5,headaxislength=3)
levels = list(range(-100,0,20))+list(range(20,100,20))
ax.contour(lon,lat,dat[0],levels=levels,colors='k',linewidths=0.5)
ax = fig.add_subplot(2,2,4)
ax.quiver(lon[::d],lat[::d],u[::d,::d],v[::d,::d],
          scale=110,units='width',pivot='mid',color='k',
          headwidth=5,headlength=5,headaxislength=2)
plt.show()
```



- `quiverkey()`绘制矢量参考
 - 第1个参数为`quiver()`函数返回的句柄
 - 参数2和3是参考矢量要画的位置，与`coordinates`的参数设置有关
 - 第4个参数为参考矢量的数值
 - `label`是描述性文字，一般说明参考矢量的大小
 - `color`为矢量的颜色
 - `labelcolor`是说明文字的颜色
 - `labelpos`是说明文字相对于矢量的位置有N、S、W、E
 - `labelsep`是说明离矢量的距离
 - `fontproperties`是说明文字的进一步设置
 - 若想使参考矢量位于所有图层最上方需要使用`zorder`
 - 使用`fill()`函数为参考矢量提供一个底

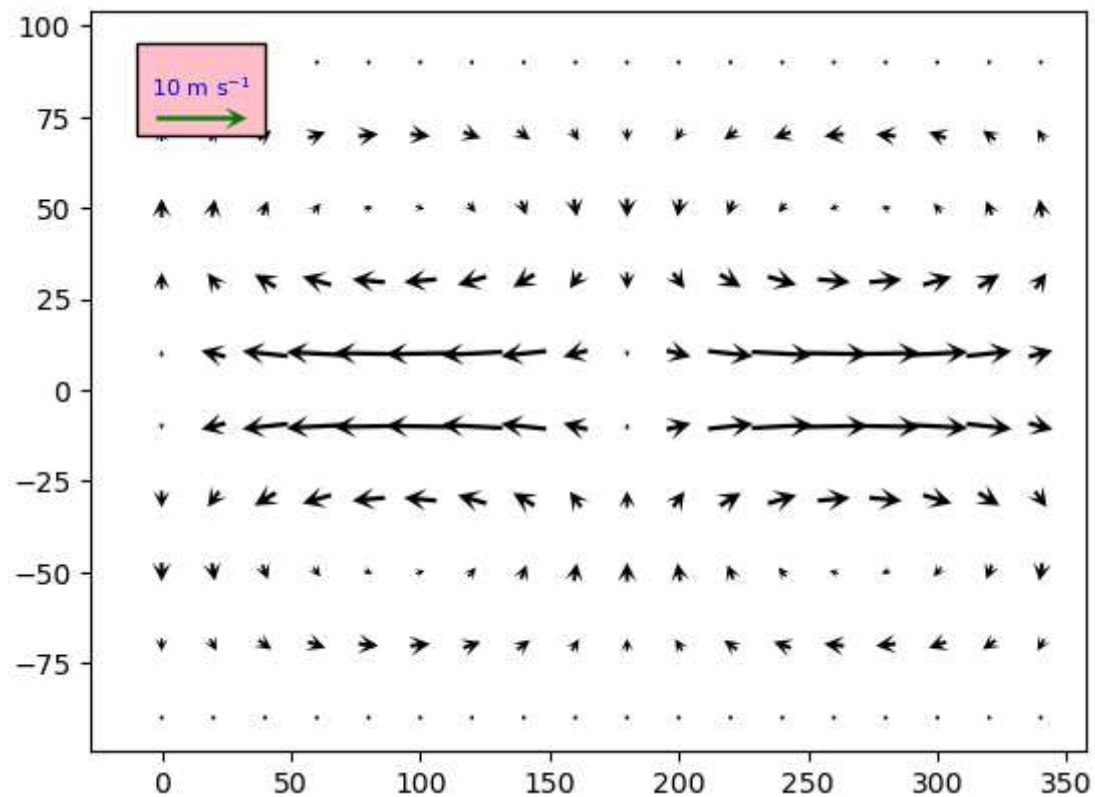
```
In [23]: fig = plt.figure()
d = 8
ax = fig.add_subplot(1,1,1)
q = ax.quiver(lon[::d],lat[::d],u[::d,::d],v[::d,::d],
              scale=110,units='width',pivot='mid',color='k',
              headwidth=5,headlength=5,headaxislength=3,
              zorder=1)

loc = (15,75)

ax.fill([loc[0]-25,loc[0]+25,loc[0]+25,loc[0]-25,loc[0]-25],
        [loc[1]-5,loc[1]-5,loc[1]+20,loc[1]+20,loc[1]-5],
        facecolor='pink',edgecolor='k',zorder=1.1)

ax.quiverkey(q,loc[0],loc[1],10,label='10 m s'+r'$^{-1}$',coordinates='data',
             color='green',labelcolor='blue',labelpos='N',labelsep=0.08,
             fontproperties={'size':8},
             zorder=1.5)

plt.show()
```



e) 垂直气压坐标

气象上y轴有时是由气压表示的高度变化，并且个别情况下还需要考虑到地形。下面以NCEP-DOE Reanalysis 2为例绘制2025年1月1日位势高度沿85°E在北半球的剖面。

- 需使用invert_yaxis()和set_yscale()函数
- 以及将x和y轴进行设置

```
In [24]: f = Dataset('pres.sfc.gauss.2025.nc')
pres = f.variables['pres'][0]
lonp = f.variables['lon'][:]
latp = f.variables['lat'][:]

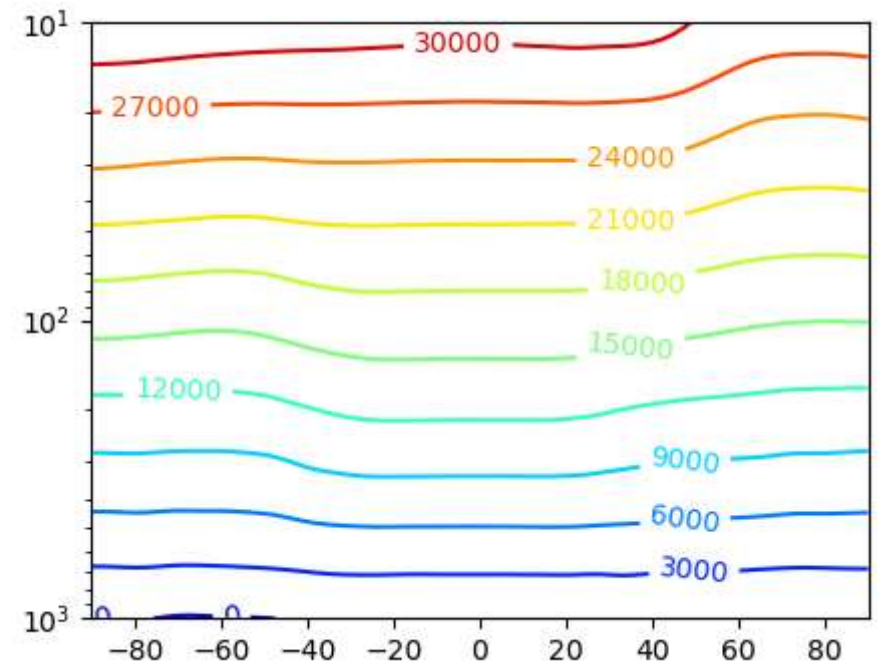
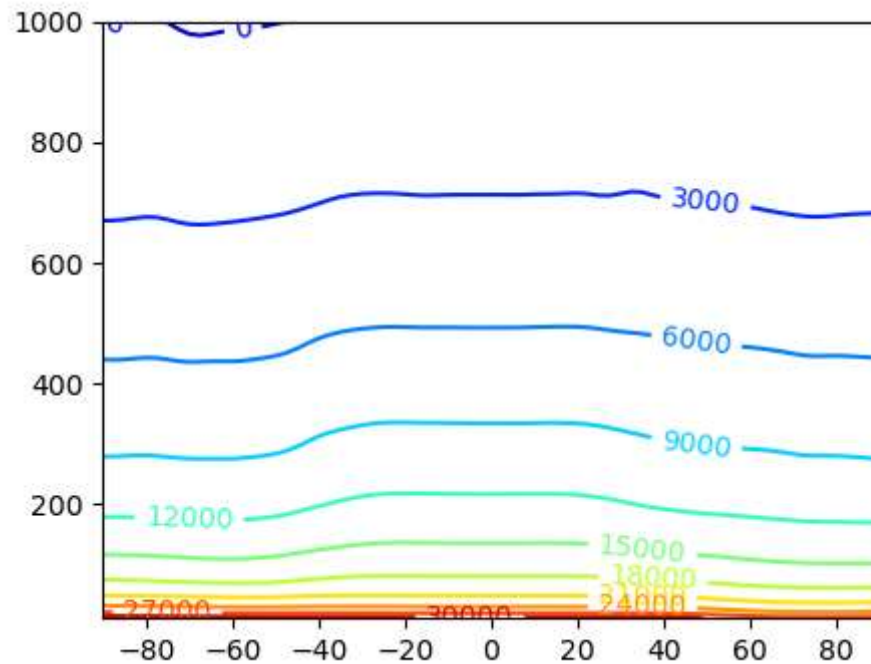
f = Dataset('hgt.2025.nc')
hgt = f.variables['hgt'][0]
```

```
lon = f.variables['lon'][:]
lat = f.variables['lat'][:]
levels = f.variables['level'][:]

msk = lon==84.375 #取了相近的纬度，亦可求平均
pres = pres[:,msk]/100
msk = lon==85
hgt = hgt[:,msk][:,0]

fig = plt.figure(figsize=(11,8.5))
ax = fig.add_subplot(2,2,1)
h = ax.contour(lat,levels,hgt,levels=11,cmap='jet')
ax.clabel(h)

ax = fig.add_subplot(2,2,2)
h = ax.contour(lat,levels,hgt,levels=11,cmap='jet')
ax.clabel(h)
ax.set_yscale('log')
ax.invert_yaxis()
plt.show()
```



```
In [25]: fig = plt.figure(figsize=(11,8.5))
ax = fig.add_subplot(1,2,1)
h = ax.contour(lat,levels,hgt,levels=11,cmap='jet')
ax.clabel(h)
ax.set_yscale('log')
ax.invert_yaxis()
ax.set_yticks(levels)
ax.set_yticklabels([str(int(x))+ ' hPa' for x in levels],fontsize=10)
ax.set_xticks(np.arange(-90,91,15))
ax.set_xticklabels([str(int(x))+r'$^{\circ}$'+ 'E' for x in np.arange(-90,91,15)])
ax.set_xlim([0,90])

ax = fig.add_subplot(1,2,2)
h = ax.contour(lat,levels,hgt,levels=11,cmap='jet')
ax.clabel(h)
ax.set_yscale('log')
ax.invert_yaxis()
ax.set_yticks(levels)
ax.set_yticklabels([str(int(x))+ ' hPa' for x in levels],fontsize=10)
```



```
ax.set_xticks(np.arange(-90,91,15))
ax.set_xticklabels([str(int(x))+r'$^o$'+ 'E' for x in np.arange(-90,91,15)])
ax.set_xlim([0,90])
ax.set_ylim([1000,10])

pres = pres[:,0].tolist()
latp = latp.tolist()
latp = latp[:1]+latp+latp[-1:]
pres = [1000]+pres+[1000]

ax.fill(latp,pres,facecolor='gray')
plt.show()
```

