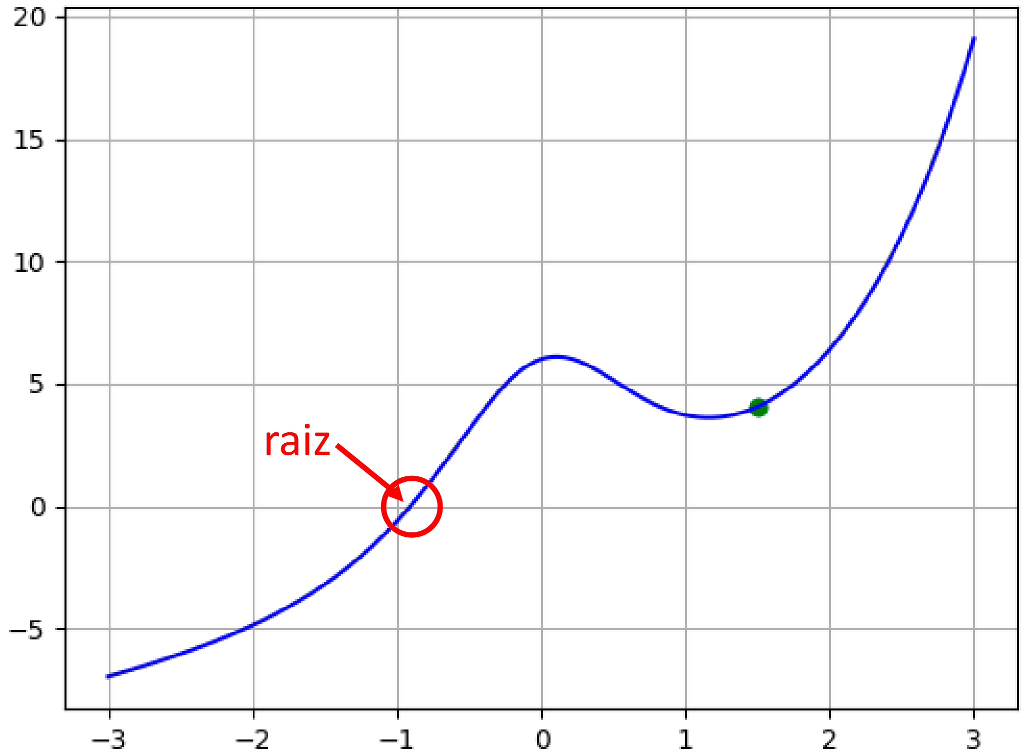
Aula 7

Determinação de raízes pelo método de Newton (parte 2).

Dados em 2D. Gráficos 2D. Cartografia.

Método de Newton



Vejamos o seguinte exemplo:

$$f(x) = x + e^x + \frac{10}{1 + x^2} - 5$$

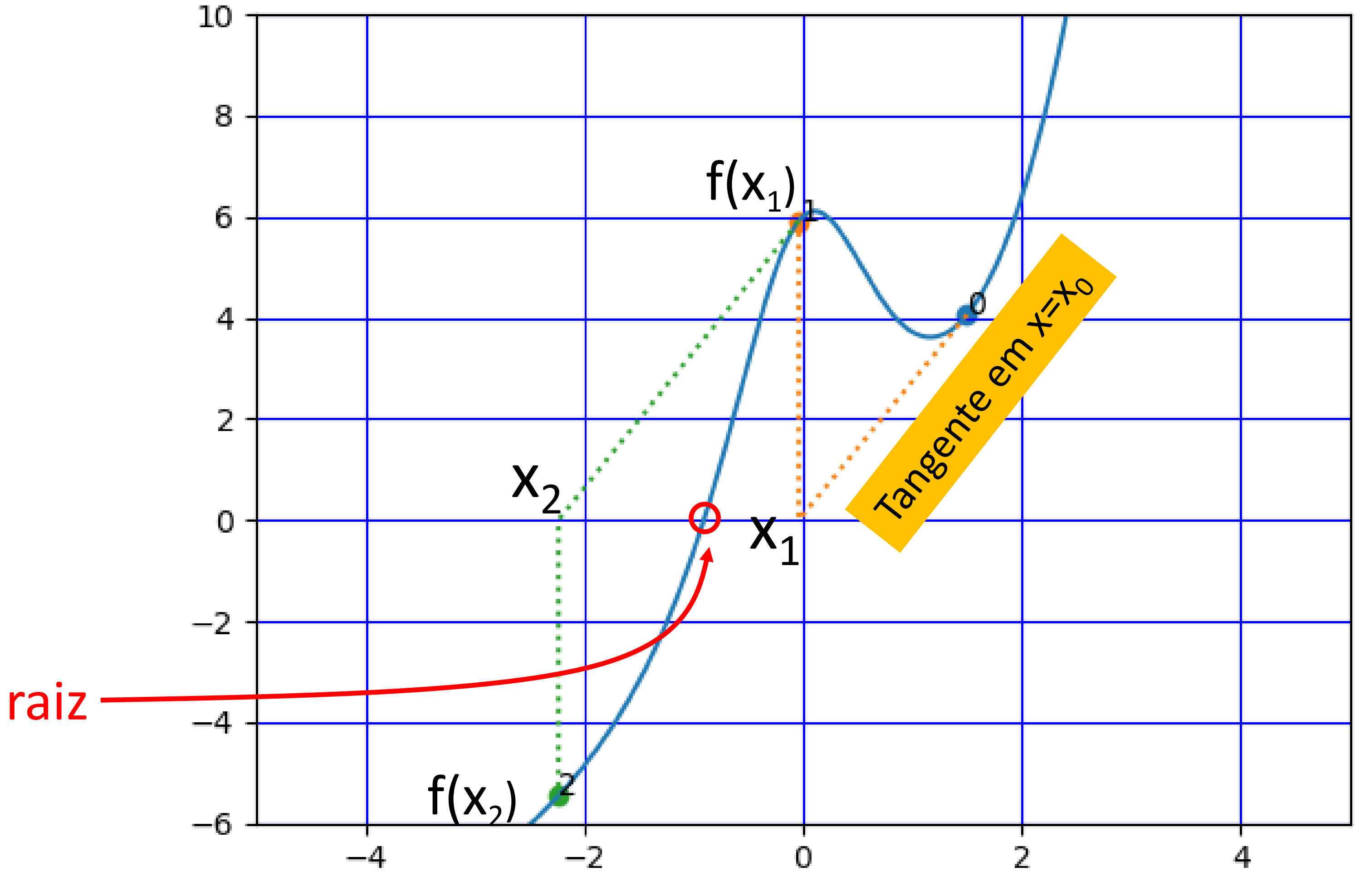
Partindo do dado inicial: x[0]=1.5

Código essencial

```
import numpy as np
def f(x):
    y=x+np.exp(x)+10./(1.+x**2)-5
    return y
def fprime(x): #df/dx
    y=-20.0*x/(x**2 + 1.0)**2 + np.exp(x) + 1
    return y
x=1.5;
maxIter=20; kit=0
while kit<maxIter:
    kit=kit+1
    x=x-f(x)/fprime(x)
```

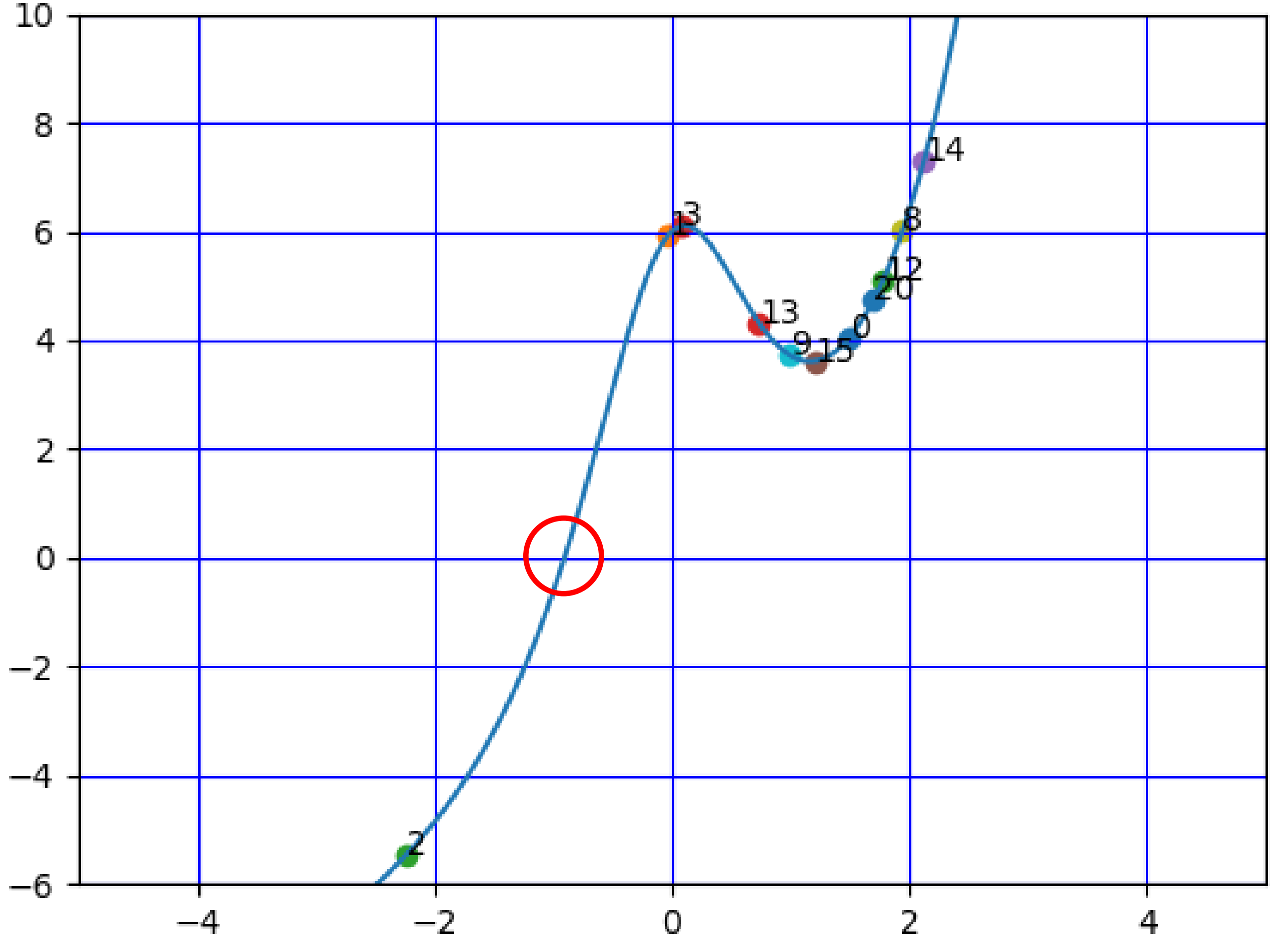
Interpretação geométrica: 2 iterações

$$f(x) = x + e^x + \frac{10}{1 + x^2} - 5$$



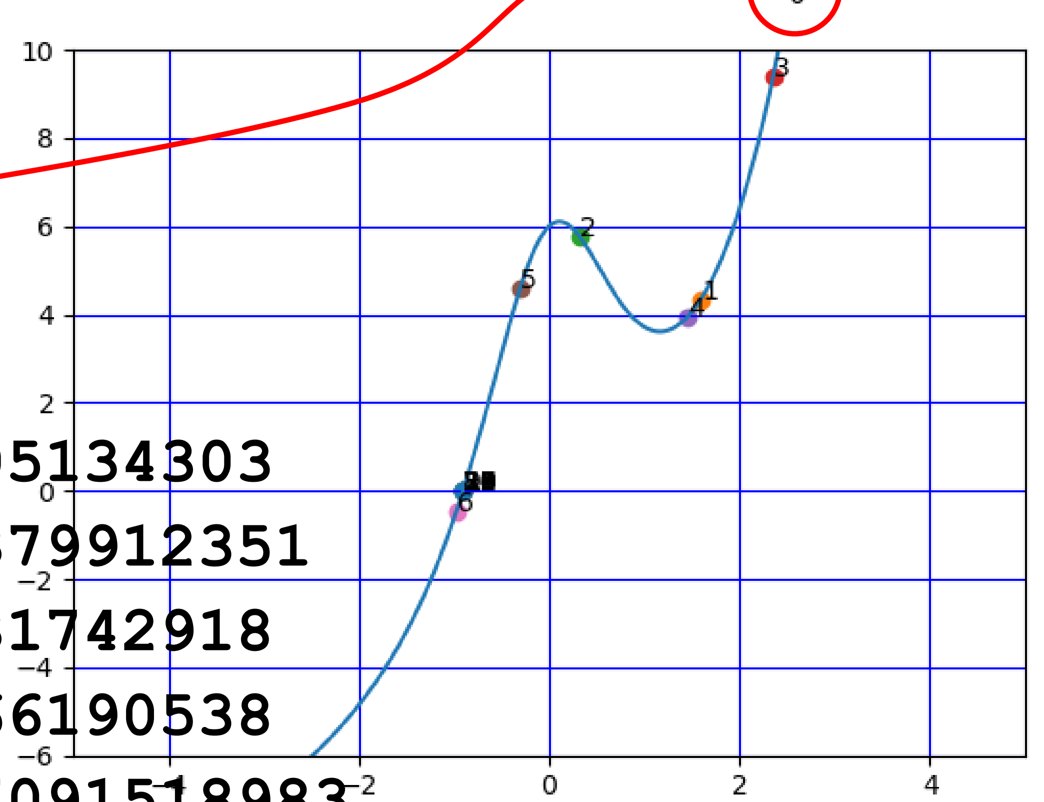
A convergência não é garantida !

xSTART=1.5
maxITER=20



Newton

xSTART=2.5



x, f(x)

1.59561096182 4.34705134303

0.314570243215 5.78379912351

2.35130850143 9.38231742918

1.44881562435 3.93366190538

-0.306478668229 4.57091518983

-0.972969770835 -0.458033808979

-0.902643768769 0.0132054216333

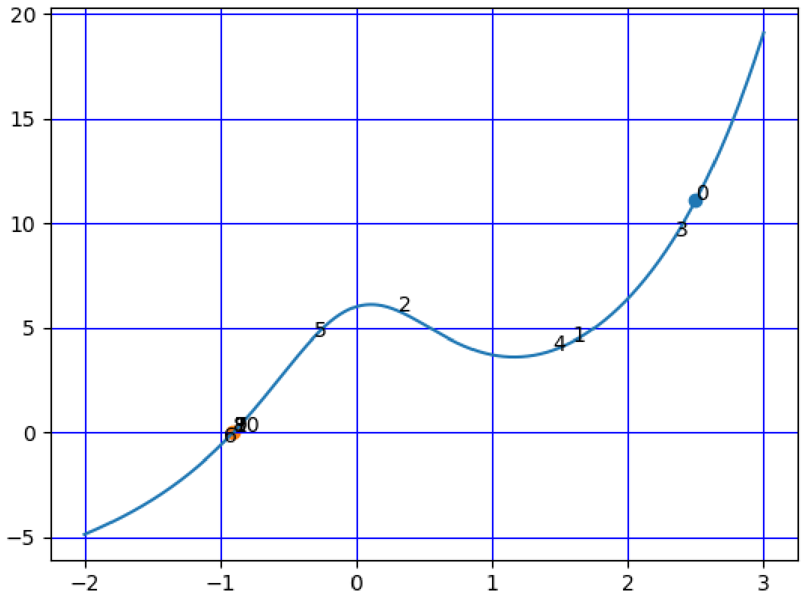
-0.904561191218 9.63369817875e-06

-0.904562592072 5.14610576374e-12

-0.904562592072 8.881784197e-16

Neste caso converge

“Convergência”




```
import numpy as np
def f(x):
    y=x+np.exp(x)+10./(1.+x**2)-5
    return y
def fprime(x):
    y=-20.0*x/(x**2 + 1.0)**2 + np.exp(x) + 1
    return y
x=2.5;maxIter=20; kit=0; tol=1.e-6; move=1000
plt.scatter(x,f(x));plt.text(x,f(x),kit)
while kit<maxIter and np.abs(move)>tol:
     move=-f(x)/fprime(x)
     x=x+move
     kit=kit+1
     print('%3i %10.6f %10.6e' % (kit,x,f(x)))
     plt.text(x,f(x),kit)
plt.scatter(x,f(x))
```

```
  1   1.595611 4.347051e+00
  2   0.314570 5.783799e+00
  3   2.351309 9.382317e+00
  4   1.448816 3.933662e+00
  5  -0.306479 4.570915e+00
  6  -0.972970 -4.580338e-01
  7  -0.902644 1.320542e-02
  8  -0.904561 9.633698e-06
  9  -0.904563 5.146106e-12
 10  -0.904563 8.881784e-16
```

Newton *vs* Bisseção $f(x) = x + e^x + \frac{10}{1+x^2} - 5$

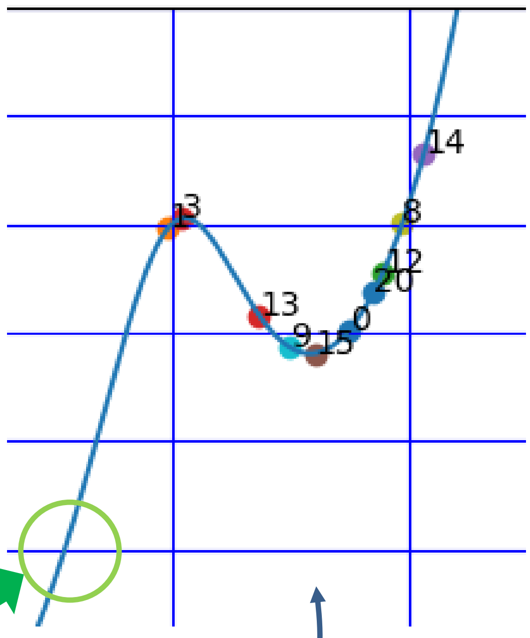
```
iTer raiz              "erro"
Newton: (maxErro<1e-6)
10 -0.904562592072 -7.48305528268e-13
Bisseçcao:
20 -0.9045600891113281 3.814697265625e-06

Newton: (maxErro<1e-12)
10 -0.904562592072 -7.48305528268e-13
Bisseçcao:
42 -0.9045625920725797 9.094947017729282e-13
```

Comentários

O método de Newton, quando converge, é muito mais eficiente que a bissecção.

É aplicável em problemas multidimensionais.

fsolve usa uma versão do método de Newton…

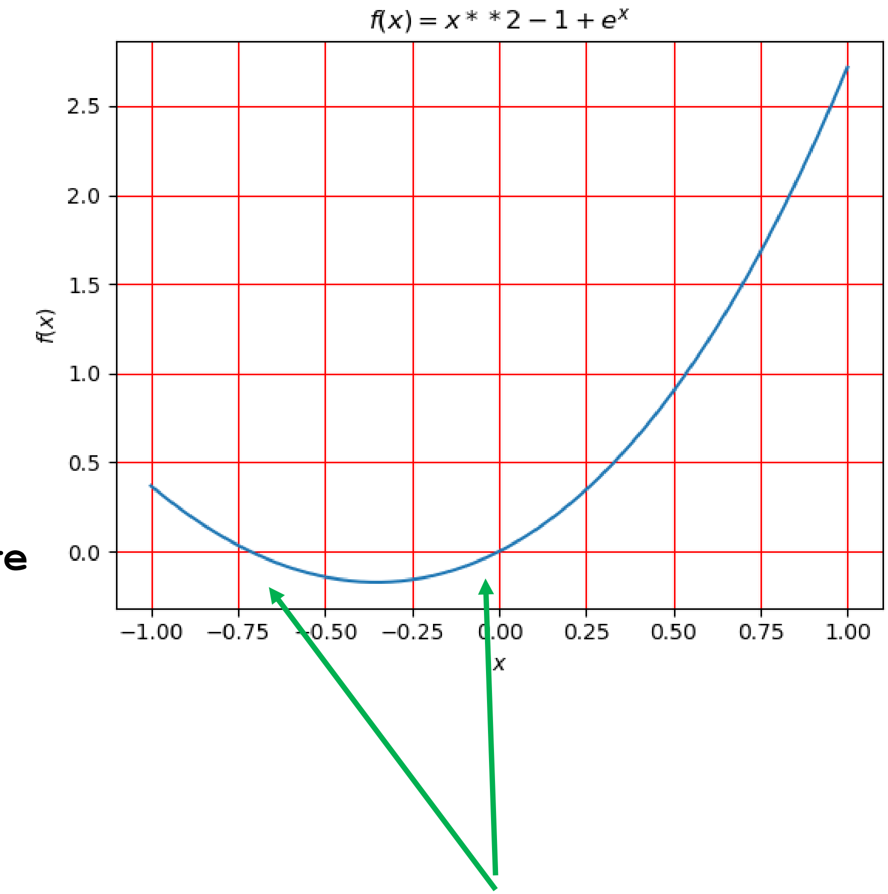


```
def f(x):
    y=x+np.exp(x)+10./(1.+x**2)-5
    return y
from scipy.optimize import fsolve
r=fsolve(f,2.5)
print('fsolve:',r)
>>fsolve: [-0.90456259] OK!
r=fsolve(f,1.5)
print('fsolve:’,r)
>> :\ProgramData\Anaconda3\lib\site-packages\scipy\optimize\minpack.py:161: RuntimeWarning: The iteration is not making good progress, as measured by the
  improvement from the last ten iterations.
```

fsolve,solve

$$f(x) = x^2 - 1 + e^x$$

```
def f(x):
    y=x**2-1+np.exp(x)
    return y
import numpy as np
from sympy import Symbol,exp,solve
xR=fsolve(f,0.25)
print(xR)
>>[  8.50262600e-18] #≈ 0
xR=fsolve(f,-1)
print(xR)
>>[-0.71455638]
x=Symbol('x')
Teq=solve(x**2-1+exp(x),x)
>>No algorithms are implemented to solve equation
x**2 + exp(x) - 1
```



Raizes $\in [-1,1]$

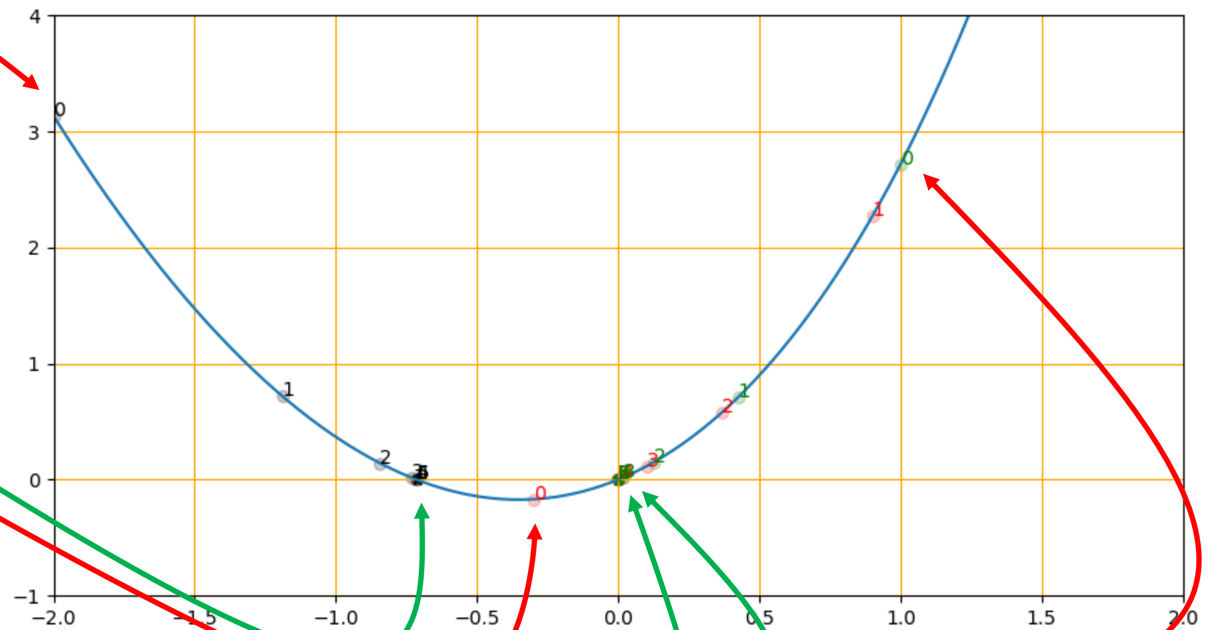
$$f(x) = x^2 - 1 + e^x$$

```
import matplotlib.pyplot as plt;import numpy as np
def f(x):
    y=x**2-1+np.exp(x);    return y
def fprime(x):
    y=2*x+np.exp(x);    return y
cores=['black','red','green']; x0=[-2,-0.3,1];maxIter=200;
for kc in range(len(cores)):
    x=x0[kc]; cor=cores[kc];kit=0; tol=1.e-6; move=1000
    print('%3i %10.6f %10.6e' % (kit,x,f(x)))
    while kit<maxIter and np.abs(move)>tol:
        move=-f(x)/fprime(x)
        x=x+move
        kit=kit+1
        plt.scatter(x,f(x),color=cor,alpha=0.2)
```

```
print('%3i %10.6f %10.6e' % (kit,x,f(x)))
```

$$f(x) = x^2 - 1 + e^x$$

```
0  -2.000000 3.135335e+00
1  -1.188717 7.176607e-01
2  -0.842494 1.404308e-01
3  -0.730539 1.533625e-02
4  -0.714880 3.045408e-04
5  -0.714557 1.305004e-07
6  -0.714556 2.409184e-14
0  -0.300000 -1.691818e-01
1   0.901420 2.275655e+00
2   0.367972 5.802046e-01
3   0.101914 1.176745e-01
4   0.012162 1.238461e-02
5   0.000215 2.147108e-04
6   0.000000 6.906540e-08
7   0.000000 7.105427e-15
0   1.000000 2.718282e+00
1   0.423883 7.075599e-01
2   0.126045 1.502200e-01
3   0.017694 1.816427e-02
4   0.000448 4.479309e-04
5   0.000000 3.001863e-07
6   0.000000 1.350031e-13
```

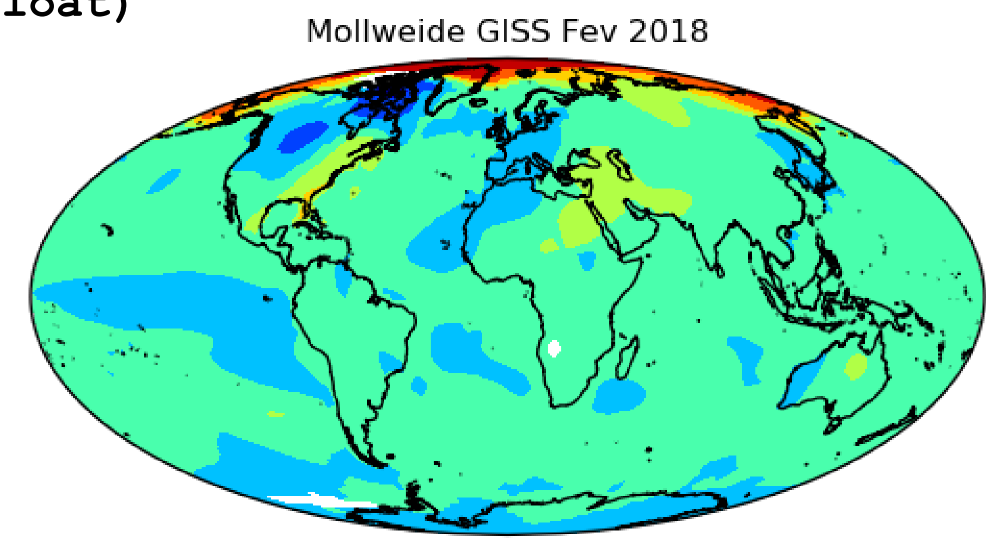


Dados 2D

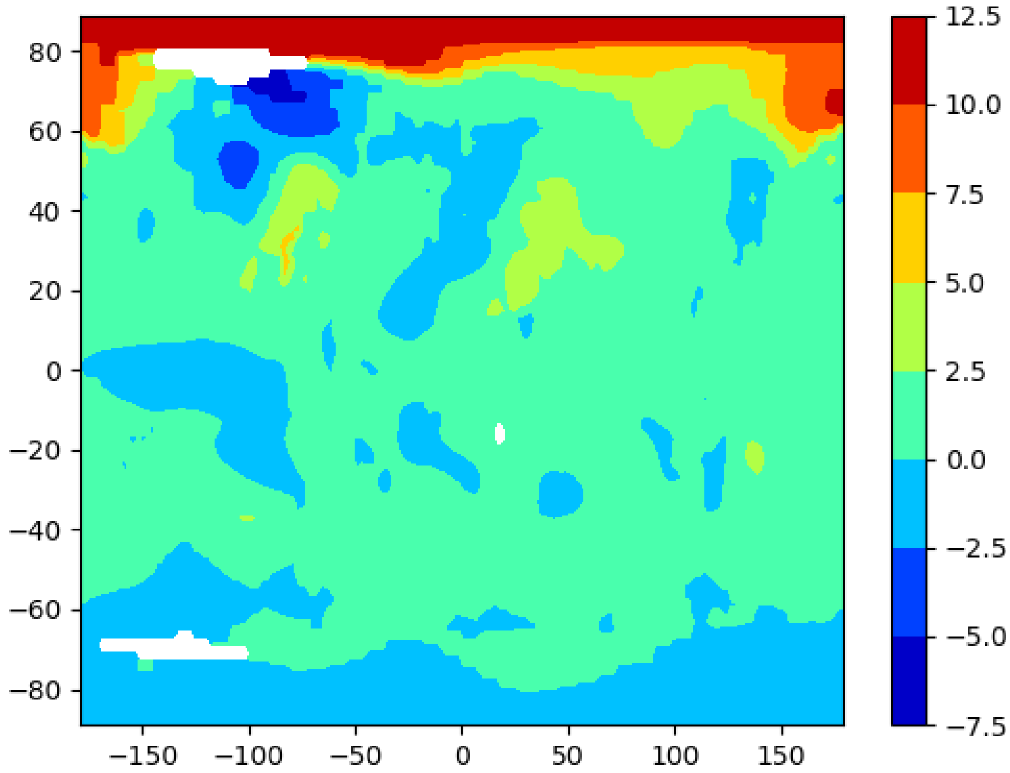
'GISS_T_Ano_Feb2018.dat'

```
1     1  -179.00   -89.00        -1.6145
2     1  -177.00   -89.00        -1.6145
3     1  -175.00   -89.00        -1.6145
4     1  -173.00   -89.00        -1.6145
```

```
import numpy as np
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
plt.close('all')
GISS=np.loadtxt('GISS_T_Ano_Feb2018.dat')
nx=180;ny=90
lon=np.zeros((ny,nx),dtype=float)
lat=np.copy(lon)
Ta=np.copy(lon)
for k in range(len(GISS)):
    ix=int(GISS[k,0])-1
    iy=int(GISS[k,1])-1
    lon[iy,ix]=GISS[k,2]
    lat[iy,ix]=GISS[k,3]
    Ta[iy,ix]=GISS[k,4]
plt.close('all')
```



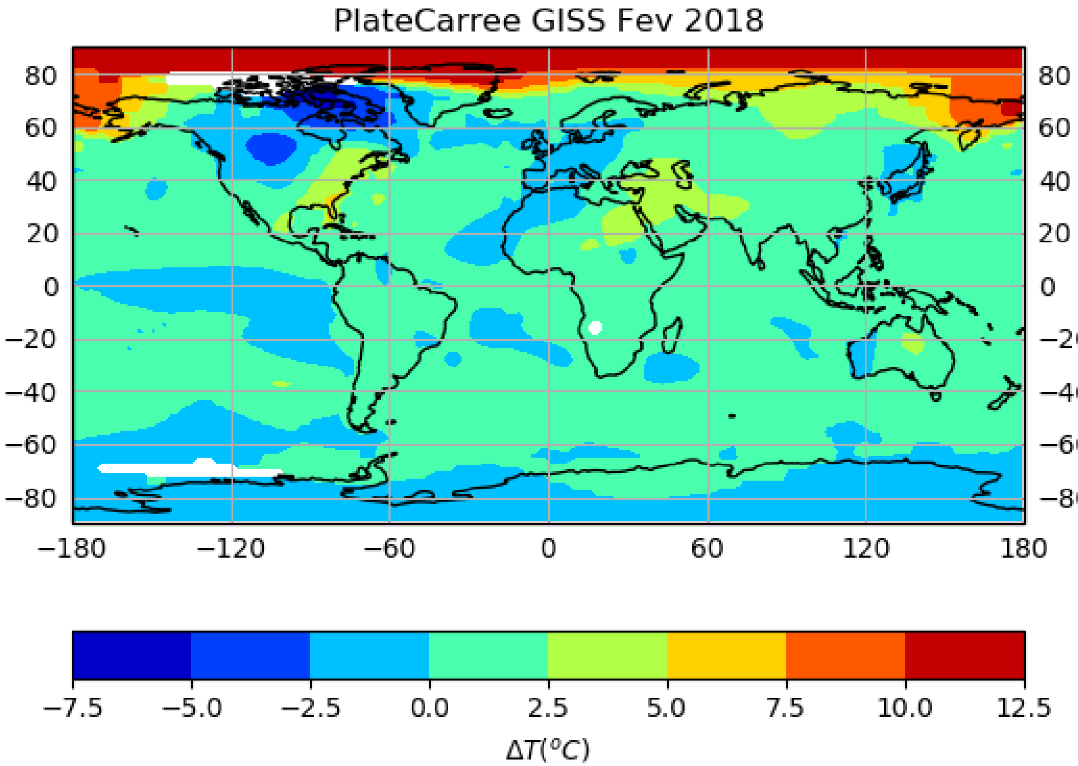
```
plt.figure()
plt.contourf(lon,lat,Ta,cmap='jet')
```

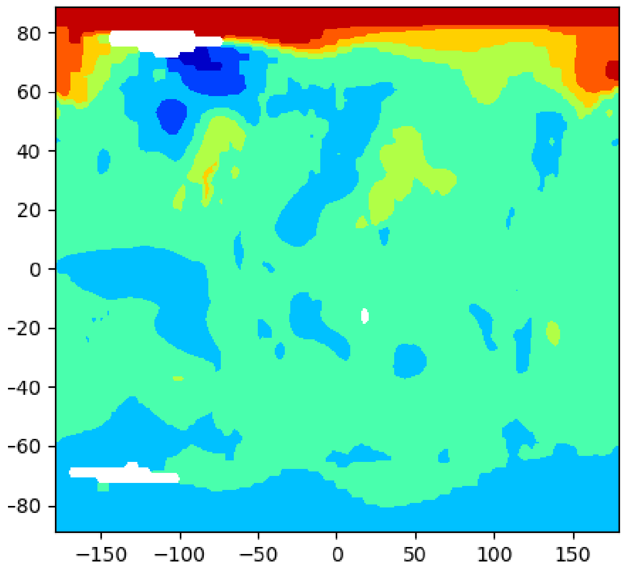


PlateCarree

```
plt.figure()
projection =ccrs.PlateCarree(central_longitude=0)
ax = plt.axes(projection=projection)
ax.set_global()
ax.coastlines()
data_crs=ccrs.PlateCarree()
map=ax.contourf(lon,lat,Ta,cmap='jet',\
    transform=data_crs )
plt.colorbar(map,orientation='horizontal',\
    label=r'$\Delta T (^{o}C)$')
gl=ax.gridlines(draw_labels=True)
gl.xlabels_top=False
plt.title('PlateCarree GISS Fev 2018')
```

```
projection =ccrs.PlateCarree(central_longitude=0)
```

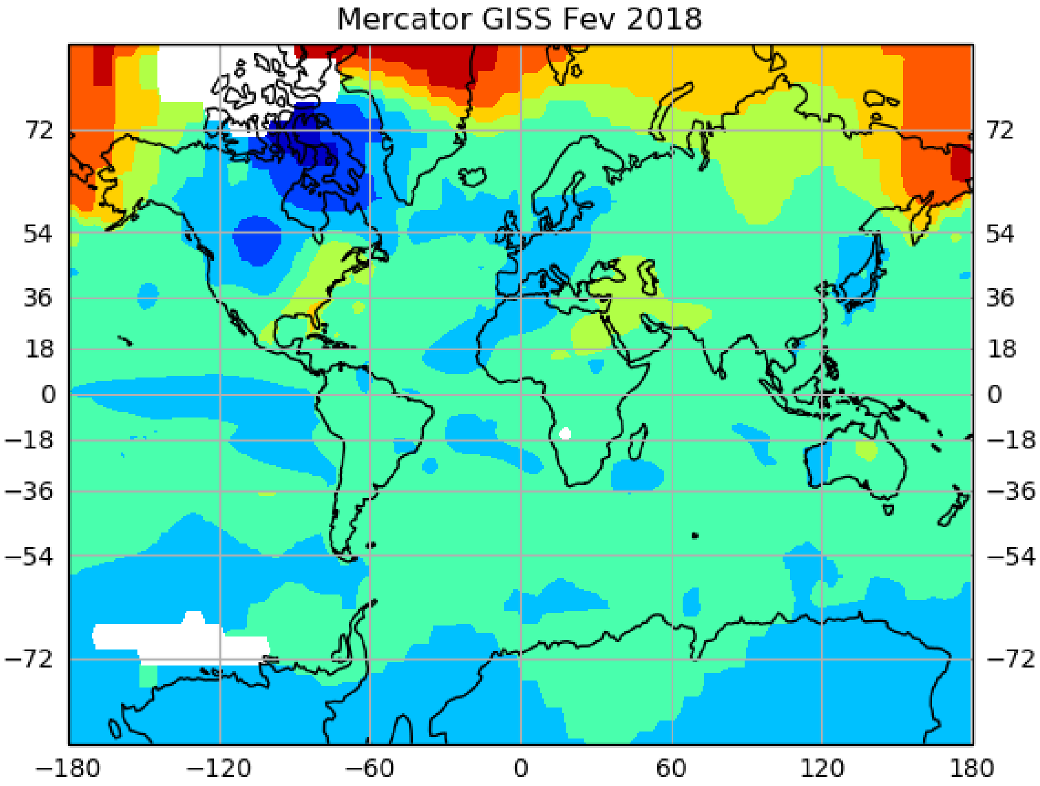


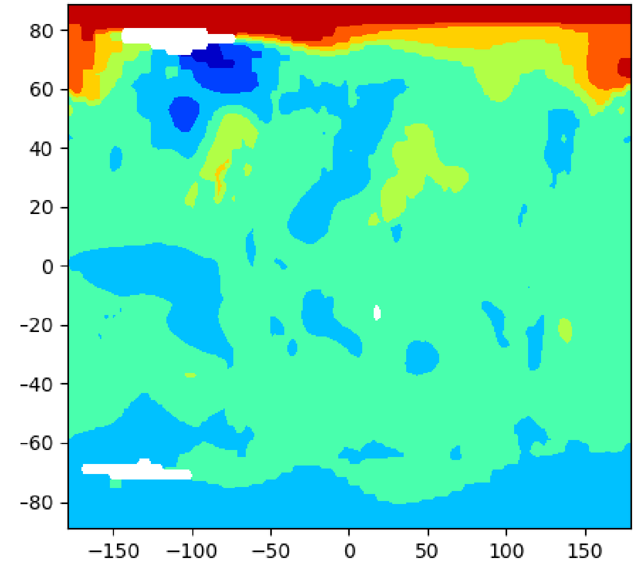


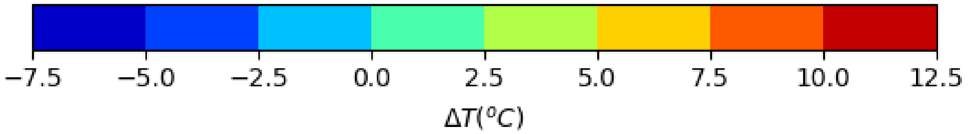
Mapa global Mercator

```
plt.figure()
projection =ccrs.Mercator(central_longitude=0,\
      min_latitude=-80,max_latitude=80)
ax=plt.axes(projection=projection)
ax.set_global()
ax.coastlines()
data_crs=ccrs.PlateCarree()
map=ax.contourf(lon,lat,Ta,cmap='jet',\
      transform=data_crs )
plt.colorbar(map,orientation='horizontal',\
      label=r'$\Delta T (^{o}C)$')
gl=ax.gridlines(draw_labels=True)
gl.xlabels_top=False
plt.title('Mercator GISS Fev 2018')
```

```
projection =ccrs.Mercator(central_longitude=0,\
     max_latitude=80,min_latitude=-80,globe=None)
```

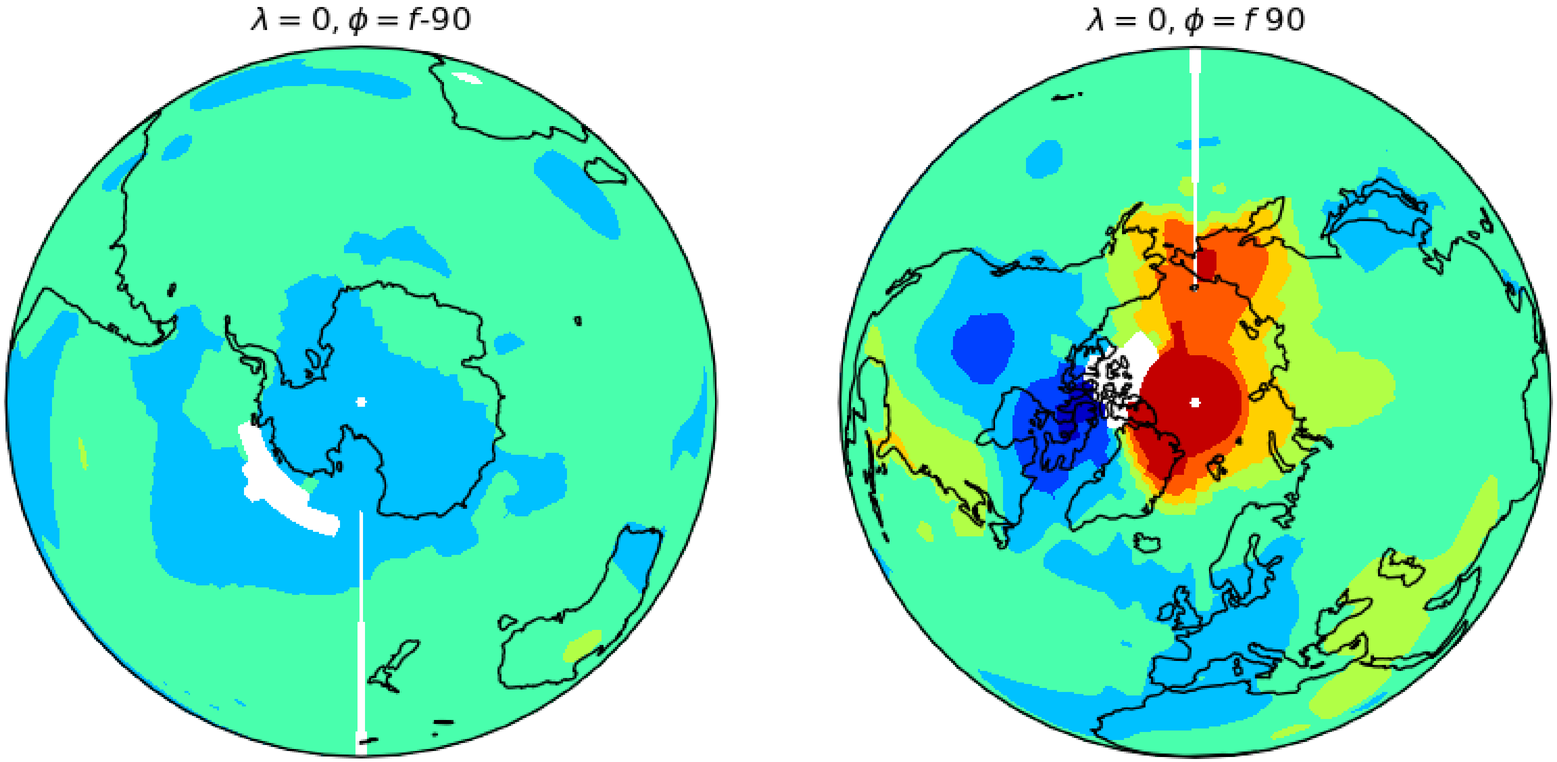






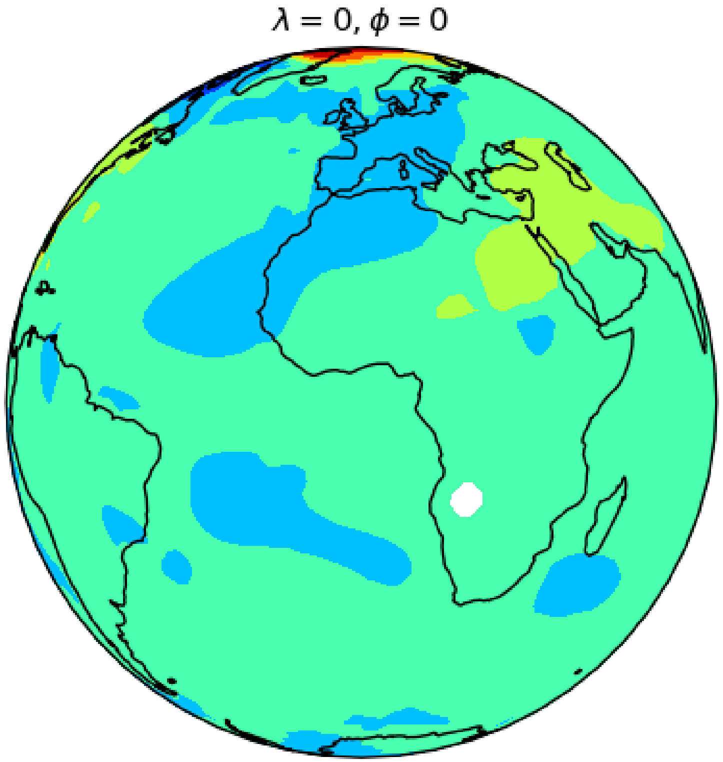
Ortographic

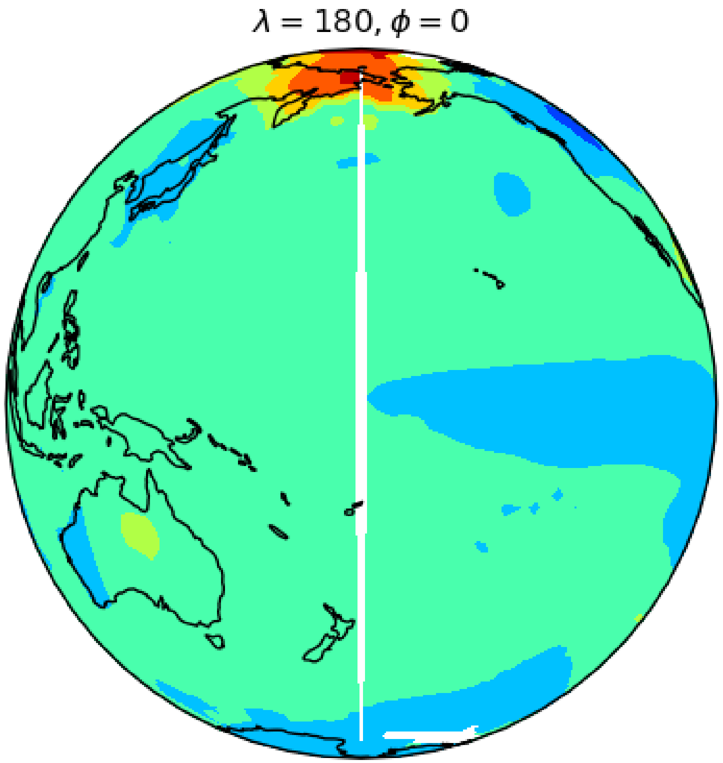
```
k=0
for clat in[-90,90]:
    k=k+1
    plt.figure()
    projection =ccrs.Orthographic(central_longitude=0,\
        central_latitude=clat,globe=None)
    ax = plt.axes(projection=projection)
    ax.set_global()
    ax.coastlines()
    data_crs=ccrs.PlateCarree()
    ax.contourf(lon,lat,Ta,cmap='jet',\
        transform=data_crs)
    plt.title(r"$\lambda=0,\phi=f$" + "%3.0f" % (clat))
    plt.show()
    plt.savefig('GISS_Ortho'+str(k)+'.png')
```


λ = 0, φ = f-90
λ = 0, φ = f 90

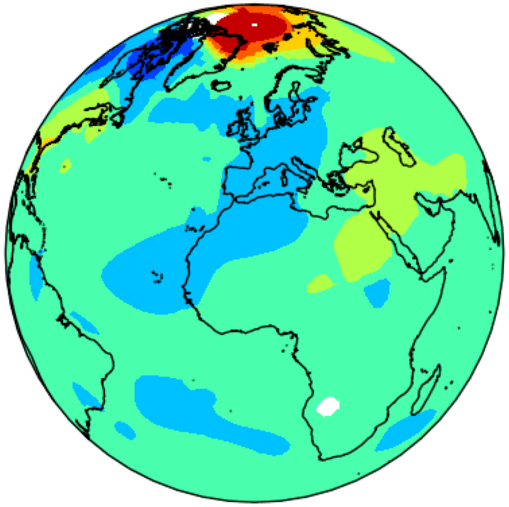
Ortographic

```
for clon in[0,180]:
    k=k+1
    plt.figure()
    projection =ccrs.Orthographic(central_longitude=\
        clon,central_latitude=0,globe=None)
    ax = plt.axes(projection=projection)
    ax.set_global()
    ax.coastlines()
    data_crs=ccrs.PlateCarree()
    ax.contourf(lon,lat,Ta,cmap='jet',\
        transform=data_crs )
    plt.title(r"$\lambda=%3.0f,\phi=0$" % (clon))
    plt.show()
    plt.savefig('GISS_Ortho'+str(k)+'.png')
```


$\lambda = 0, \phi = 0$


$\lambda = 180, \phi = 0$

```
…
import imageio; import os
frames=[]
for long in range(360,0,-10):
    plt.figure()
    projection=ccrs.Orthographic(\
       central_longitude=long,central_latitude=23.5)
    ax=plt.axes(projection=projection)
    ax.set_global()
    ax.coastlines(resolution='50m')
    data_crs=ccrs.PlateCarree()
    map=ax.contourf(lon,lat,Ta,cmap='jet’,\
       transform=data_crs)
    plt.show()
    frame='GISS_M'+str(long)+'.png'
    frames.append(frame)
    plt.savefig(frame)
    plt.clf();plt.close()
images=[]
for frame in frames:
    images.append(imageio.imread(frame))
    os.remove(frame)
imageio.mimsave('GISS.gif', images,duration=0.5)
```



'GISS.gif'