In [1]:
```
#ATSC 528 Assignment #1: Function Fitting
#Author: Taylor Dolan
#Due Date: 9/30/2022

#Created By    : Jared W. Marquis
#Creation Date : 01 August 2022
#Course        : ATSC 528 - Atmospheric Data Analysis
#Assignment    : #01 - Function Fitting

#Purpose:
#Script to take sparse upper air observations and analyze them on a polar stereographic map projection using fu
#Using a 2 dimensional polynomial function fitting technique, and using a RoI of 10 cm and 20cm
#Plotting the 500 Geopotential Heights for the ROI of 10cm and 20cm, as well as the # of obs used in the analys
#Outputting the analysis as text files to read easier
#Answering deeper analysis questions at the end of the script
```

In [2]:
```
#Imports
import numpy as np
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
import cartopy.feature as cfeature
import csv
import pandas as pd
```

In [3]:
```
#Read in observations
PATH_obs = '/Users/taylordolan/Documents/GitHub/ATSC528_2022/01-Function_Fitting/' #define the path

fileObject = open(PATH_obs + "RAOBs_201903131200.txt", "r") #open the text file
csvreader = csv.reader(fileObject) #read it
rows = []
for row in csvreader:
        rows.append(row) #This makes the text file look nice
```

In [4]:
```
#Convert text file observations to a dateframe for easier calculations (i dont like text files)
list_name = rows
df = pd.DataFrame(list_name, columns = ['Station ID', 'lat', 'lon', '500mb Height', '500mb Wind Dir', '500mb Wi
obs_lon = df['lon'].astype(float)           #longitudes, these are in degrees, dont use in equations
obs_lat = df['lat'].astype(float)           #latitudes, these are in degrees, dont use in equations
obs_ht = df['500mb Height'].astype(float)   #height values as floats
obs_ht = np.array(obs_ht)                   #height values converted to an array
```

In [5]:
```
#Convert lat and lon values to radians
obs_lon_radians = (obs_lon * (np.pi/180))
obs_lat_radians = (obs_lat * (np.pi/180))
```

In [6]:
```
#Set up analysis map with a 22x28 rectangular grid of points
x = 18.9          #xo (NW of your map, (0,0))
y = -6.3          #yo (NW of your map (0,0))
delta_x = 1.27    #delta x, this is the spacing of the points
delta_y = 1.27    #delta y, this is the spacing of the points
```

In [7]:
```
#Convert observations to x,y
x_values = x + np.arange(22)*delta_x              #23, points on the map (analysis points)
y_values = y + np.arange(28)* delta_y             #28, points on the map (analysis points)
grid_x, grid_y = np.meshgrid(x_values,y_values)   #this makes the grid
```

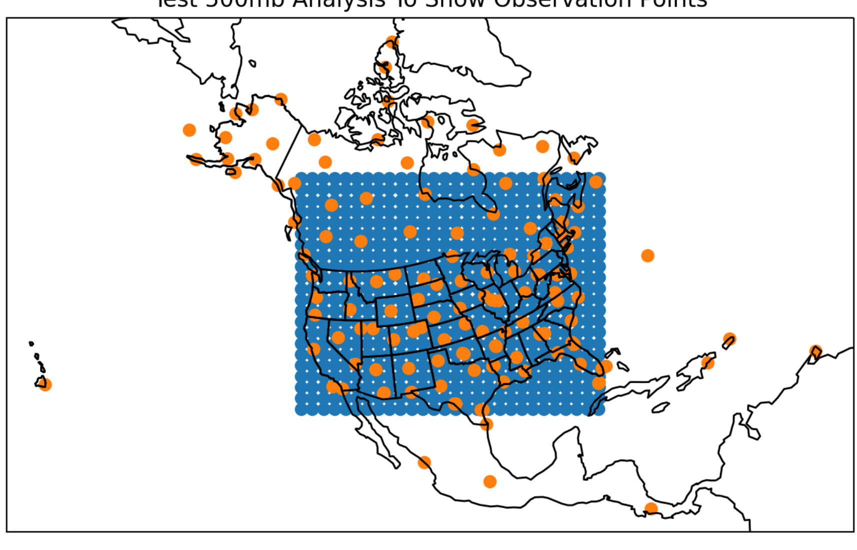
In [8]:
```
#Calculate and Define Values
map_proj = 1/(15e6) #map projection
rho = 6371 * 1e5 #radius of Earth
lambda_o = -115
phi_o = 60*(np.pi/180) #in radians
Roi_array = np.array([10,20,]) #radius of influence of 10 and 20

#Longitude (from grid to longitude, used for plotting)
new_lon = np.arctan(grid_y/grid_x)*(180/np.pi) + lambda_o

#Latitude (from grid to latitude,used for plotting
new_lat = (180/np.pi)*((np.pi/2)-(2*np.arctan(np.sqrt((grid_x/map_proj)**2+(grid_y/map_proj)**2)/(rho*(1+np.sin
```

In [9]:
```
#Convert lambda knot to radians
lambda_o_radians = (lambda_o * (np.pi/180))
```

In [10]:
```
#Convert the latitude and longitude from the text file to x and y

#Equations
#sigma (image scale factor) = (1+sin(phi_knot)/(1+sin(phi))) where phi is a latitude
#r (the radius of any lat circle on the image plane) = rho*sigma*cos(phi)
#x (coordinate) = r*cos(lambda)
#y (coordinate) = r*sin(lambda)
#lambda is the deviation (lambda(knot) - longitude of point) of lon from the standard lon, lambda (knot)

sigma_obs = (1+np.sin(phi_o))/(1+np.sin(obs_lat_radians)) #shows an array of sigma values, which is the scale f
r_obs = (rho*sigma_obs*(np.cos(obs_lat_radians)))         #shows an array of r values, the radius of any lat ci
lambda_obs = ((obs_lon_radians - lambda_o_radians))       #shows an array of the lambda values, deviation of lo
x_obs = (r_obs * (np.cos(lambda_obs))) * map_proj         #x coordinate for observations, longitudes
x_obs = np.array(x_obs)                                   #x coordinate for observations in an array (use for l
y_obs = (r_obs * (np.sin(lambda_obs))) *map_proj          #y coordinate for observations, latitudes
y_obs = np.array(y_obs)                                   #y coordinate for observations in an array (use for l

#Longitude (from grid to longitude, used for plotting)
obs_lon = np.arctan(y_obs/x_obs)*(180/np.pi) + lambda_o

#Latitude (from grid to latitude,used for plotting
obs_lat = (180/np.pi)*((np.pi/2)-(2*np.arctan(np.sqrt((x_obs/map_proj)**2+(y_obs/map_proj)**2)/(rho*(1+np.sin(p
```

In [11]:
```
#Plot 500mb analyses with lat/lon to make sure it is working
#Blue dots are the grid that was created, orange dots are where there are upper air observations

proj = ccrs.Stereographic(central_longitude=-115,central_latitude=90,true_scale_latitude=60)
fig = plt.figure(figsize=(8,8),dpi=200)
ax1 = fig.add_subplot(111,projection=proj)
ax1.add_feature(cfeature.STATES)
ax1.add_feature(cfeature.COASTLINE)
ax1.scatter(new_lon.ravel(), new_lat.ravel(), transform = ccrs.PlateCarree())
ax1.scatter(obs_lon, obs_lat, transform = ccrs.PlateCarree())
plt.title('Test 500mb Analysis To Show Observation Points')
plt.show()
```

Test 500mb Analysis To Show Observation Points

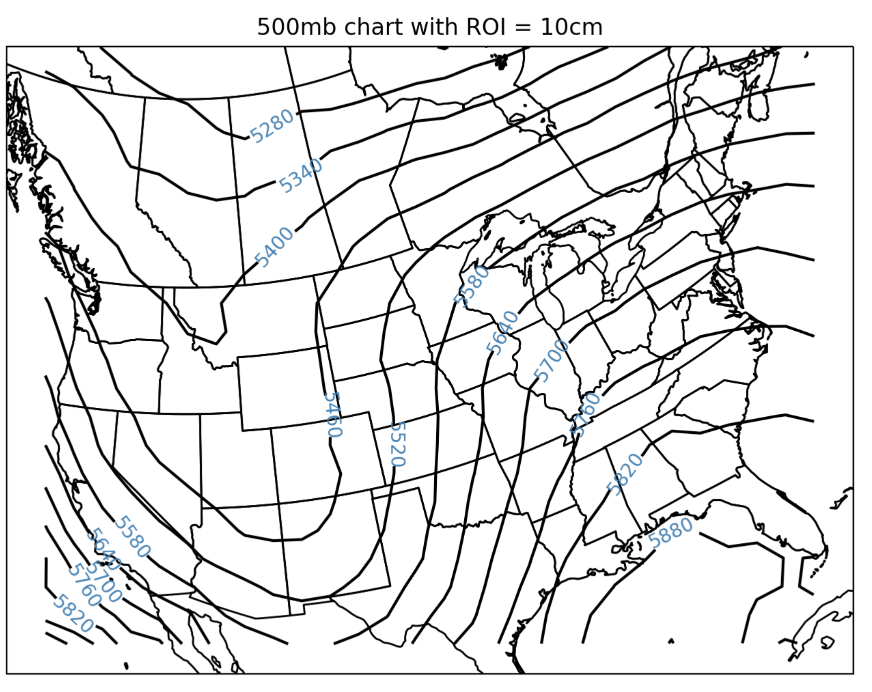


In [12]:
```
#Perform 500mb geopotential height analyses using a second order 2-d polynomial with two radii of influence (10

#Create empty matrices to store the data (22x28 matrix (technically 28x22), and include the RoI of 10 and 20 cm
x_columns = 28
y_rows = 22
obs_matrix = np.empty((x_columns,y_rows,len(Roi_array)))      #this stores the observations in the matrix
analysis_matrix = np.empty((x_columns,y_rows,len(Roi_array))) #this stores the analysis values in the matrix

#Looping through the data
for i in range(len(Roi_array)): #the length of the Roi array (which is 2)
    roi = Roi_array[i] #Defining the values
    for j in range(len(grid_x)): #going through the length of the x grid
        for k in range(len(grid_x[0])): #going through the values of the x grid, and it keeps going through the
            x_dist = grid_x[j,k] - x_obs #finding the x distance
            y_dist = grid_y[j,k] - y_obs #finding the y distance
            radius = ((x_dist)**2 + (y_dist)**2)**(1/2) #this is the distance formula
            impt_vals = np.where(radius <= roi)[0] #these are are the obs in the radius of influence

            obs_matrix[j,k,i] = len(impt_vals) #fill in the obs matrix with the shapes of lengths and impt valu

            #Identifying the xk, yk, and fo values for the 6x6 matrix (these come from that big matrix equation
            xk = x_dist[impt_vals]
            yk = y_dist[impt_vals]
            fo = obs_ht[impt_vals]

            #Creating the 6x6 analysis (R) matrix and the 1x6 (O) column to plug in the values (need to find C)
            #Dont use np.empty since it'll cause a mess
            R_matrix = np.zeros((6,6))
            O_column = np.zeros(6)

            #Fill in the R matrix and the O column
            for l in range(len(yk)):
                array_1 = np.matrix([1, xk[l], yk[l], xk[l]**2, yk[l]**2, xk[l]*yk[l]]) #creating horizonal lon
                array_1_transposed = array_1.T #creating tall long matrix
                array_2 = array_1_transposed * array_1 #storing the multiplication here
                R_matrix += np.array(array_2) #filling in the R_matrix
                O_column += np.array([fo[l], xk[l]*fo[l], yk[l]*fo[l], xk[l]**2*fo[l], yk[l]**2*fo[l], xk[l]*yk

            #Get the averages inside the matrices/column
            R_avg = R_matrix/obs_matrix[j,k,i] #fill it in, one should be in the top left, it is!!!
            O_avg = O_column/obs_matrix[j,k,i] #fill it in

            #Create the c column
            c = np.linalg.inv(np.matrix(R_avg))*np.matrix(O_avg).T #need it to be transposed, seen in the notes

            #Fill in analysis matrix
            analysis_matrix[j,k,i] = c[0,0]
```

In [13]:
```
#Plots below
```

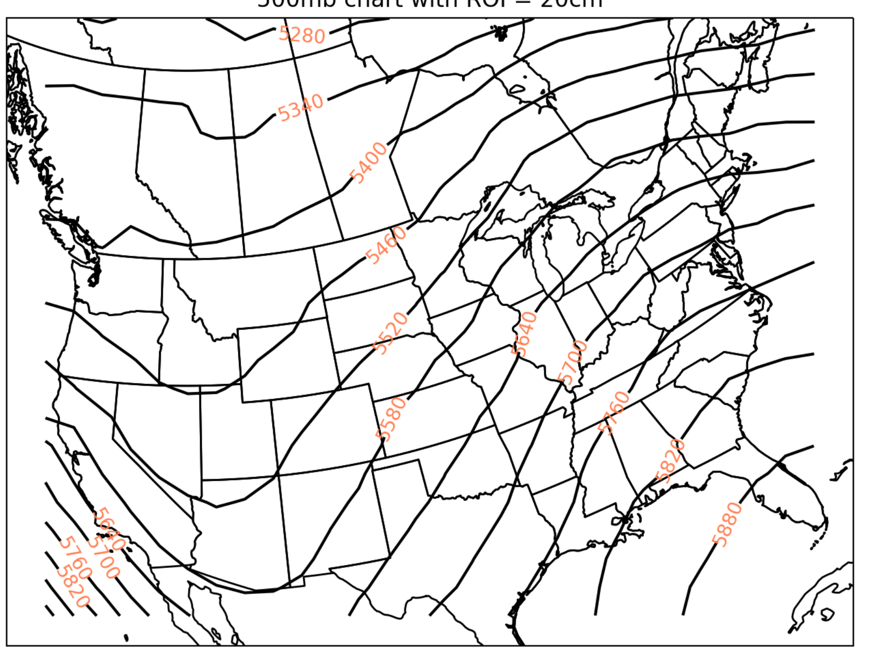
In [14]:
```
#500 mb Contour Chart, ROI = 10cm
proj = ccrs.Stereographic(central_longitude=-115,central_latitude=90,true_scale_latitude=60)
fig = plt.figure(figsize=(8,8),dpi=200)
ax1 = fig.add_subplot(111,projection=proj)
ax1.add_feature(cfeature.STATES)
ax1.add_feature(cfeature.COASTLINE)

cs1 = ax1.contour(new_lon,new_lat,analysis_matrix[:,:,0],colors='k',levels=np.arange(0,8000,60),transform=ccrs.
plt.clabel(cs1,levels=np.arange(0,8000,60), colors = 'steelblue')
plt.title('500mb chart with ROI = 10cm')
plt.show()
```

500mb chart with ROI = 10cm



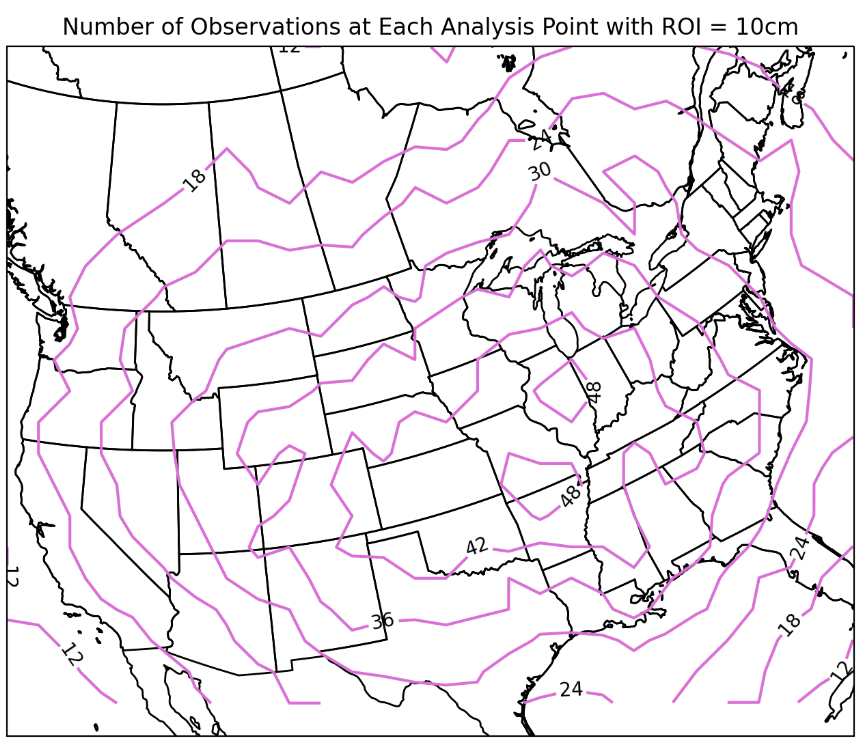
In [15]:
```
#500 mb Contour Chart, ROI = 20cm
proj = ccrs.Stereographic(central_longitude=-115,central_latitude=90,true_scale_latitude=60)
fig = plt.figure(figsize=(8,8),dpi=200)
ax1 = fig.add_subplot(111,projection=proj)
ax1.add_feature(cfeature.STATES)
ax1.add_feature(cfeature.COASTLINE)

cs1 = ax1.contour(new_lon,new_lat,analysis_matrix[:,:,1],colors='k',levels=np.arange(0,8000,60),transform=ccrs.
plt.clabel(cs1,levels=np.arange(0,8000,60), colors = 'coral')
plt.title('500mb chart with ROI = 20cm')
plt.show()
```

500mb chart with ROI = 20cm



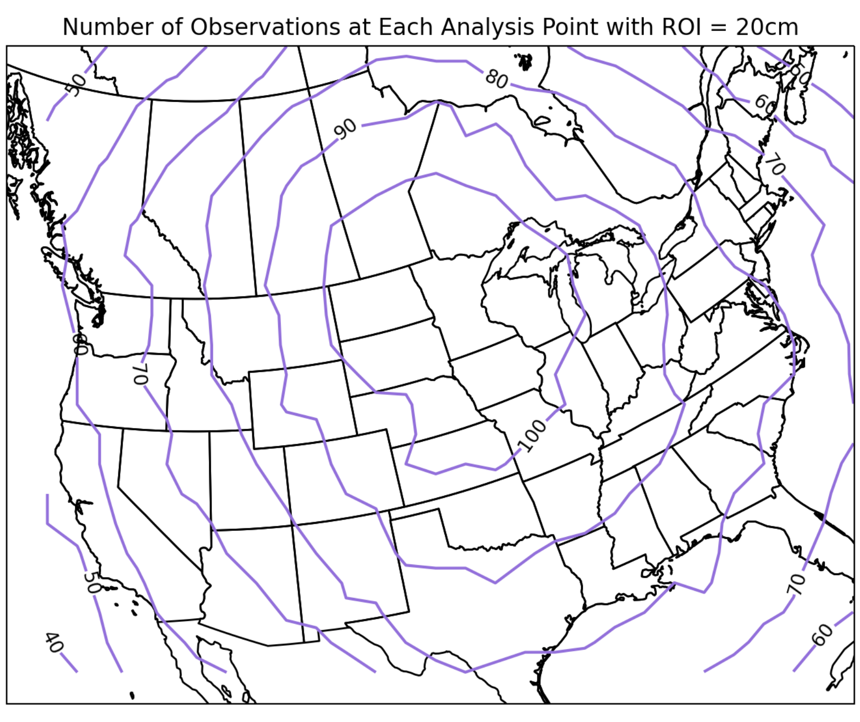
In [16]:
```
#Plot having the number of observations at each analysis point (ROI = 10cm)
proj = ccrs.Stereographic(central_longitude=-115,central_latitude=90,true_scale_latitude=60)
fig = plt.figure(figsize=(8,8),dpi=200)
ax1 = fig.add_subplot(111,projection=proj)
ax1.add_feature(cfeature.STATES)
ax1.add_feature(cfeature.COASTLINE)

#plot number of observations
cs1 = ax1.contour(new_lon,new_lat,obs_matrix[:,:,0],colors='orchid',transform=ccrs.PlateCarree())
plt.clabel(cs1, colors = 'black')
plt.title('Number of Observations at Each Analysis Point with ROI = 10cm')
plt.show()
```

Number of Observations at Each Analysis Point with ROI = 10cm



In [17]:
```
#Plot having the number of observations at each analysis point (ROI = 20cm)
proj = ccrs.Stereographic(central_longitude=-115,central_latitude=90,true_scale_latitude=60)
fig = plt.figure(figsize=(8,8),dpi=200)
ax1 = fig.add_subplot(111,projection=proj)
ax1.add_feature(cfeature.STATES)
ax1.add_feature(cfeature.COASTLINE)

#plot number of observations
cs1 = ax1.contour(new_lon,new_lat,obs_matrix[:,:,1],colors='mediumpurple',transform=ccrs.PlateCarree())
plt.clabel(cs1, colors = 'black')
plt.title('Number of Observations at Each Analysis Point with ROI = 20cm')
plt.show()
```

Number of Observations at Each Analysis Point with ROI = 20cm



In [18]:
```
#Store the analyses in text files (Idk how to change it to not be in scientific notation)
a_file = open("Analysis_Matrix_ROI_10", "w")
for line in analysis_matrix[:,:,0]:
    np.savetxt(a_file, line)

b_file = open("Analysis_Matrix_ROI_20", "w")
for line in analysis_matrix[:,:,1]:
    np.savetxt(b_file, line)

c_file = open("Observation_Matrix_ROI_10", "w")
for line in obs_matrix[:,:,0]:
    np.savetxt(c_file, line)

d_file = open( "Observation_Matrix_ROI_20", "w")
for line in obs_matrix[:,:,1]:
    np.savetxt(d_file, line)
```

In [19]:
```
#In a separte text file (or below), answer the following questions
'''
1 - Describe the general features that you see in your contoured analyses.

    In the ROI = 10cm map, there is a large trough in the midwest over New Mexico and Arizona. A ridge is locat
    North of the 5400 geopotential height line, the temperatures would be below freezing, as this line represen
    The ridge in the southeastern part of the country represents warmer temperatures than the temperatures asso
    In the southwest portion of the map, the isobars are closer together, which indicates faster winds. In gene
    together across the US, meaning faster winds and steeper gradient. Note: there is also a weird line feature
    and I do not know why.

    In the ROI = 20cm map, there is a smaller trough in the southwest US, and the isobars are more spread apart
    height field is also located more northward in Canada. The ridge in the southeastern part of the map isn't
    higher heights indicating warmer temperatures. And also the weird feature off the coast of Florida is gone.

2 - Describe the differences that you see in your contoured analyses.
    Does one analysis seem to be smoother than the other?  If so, what would cause this?

    In the 10 cm radius of influence analyses map, the contours are closer together when compared to the ROI of
    The ROI = 20 cm analyses map looks smoother when compared to the 10 cm one.
    When looking at the ROI = 20 observation map, the contours are more smoothed and circular compared to the R
    The ROI = 20 map is using a larger area and more observational points, so more points are in the analysis,
    everything to be smoothed out compared to when there are not a lot of points, meaning if something is bad,

3 - Run your program using a radius of influence of 6 cm (do not need to show).  (put 6 in array, will look fun
    Describe the results - do they look realistic?  If there are problems, what
    do you think might be causing them?

    The results from doing this are not realistic. One of the main problems is the distorition, especially near
    over Canada. Going back to the previous questions, I think that if you have less points (observations), the
    be as smooth, causing problems and distortions, similar to the small distortions when using the ROI = 10cm.
    amount of observation points, the results will not be as smoothed out, causing localized problems to be mor

4 - Suppose you ran this program with a small enough radius of influence that only one
    observation was available for determining a polynomial fit at a grid point.  Should
    you be able to perform the matrix inversion?  Why or why not?

   If only one observation was available, then that would mean that your matrix would just have the value "1" i
   the previous calculations and methods above, the only time you would run into any issue is because you are d
   matrix in the denominator. The equation is R_avg = R_matrix/obs_matrix[j,k,i] and O_avg = O_column/obs_matri
   by "1", that math works, so it shouldn't be an issue. So YES, you should be able to perform it.

'''
```

Out[19]:
```
'\n1 - Describe the general features that you see in your contoured analyses.\n    \n    In the ROI = 10cm map,
there is a large trough in the midwest over New Mexico and Arizona. A ridge is located over the southeast.\n
North of the 5400 geopotential height line, the temperatures would be below freezing, as this line represents t
o freezing line.\n    The ridge in the southeastern part of the country represents warmer temperatures than the
temperatures associated with the trough.\n    In the southwest portion of the map, the isobars are closer toget
her, which indicates faster winds. In general, the isobars are closer\n    together across the US, meaning fast
er winds and steeper gradient. Note: there is also a weird line feature off the coast of Florida,\n    and I do
not know why.\n    \n    In the ROI = 20cm map, there is a smaller trough in the southwest US, and the isobars
are more spread apart. The 5400 geopotential\n    height field is also located more northward in Canada. The ri
dge in the southeastern part of the map isn\'t as defined, but there are\n    higher heights indicating warmer
temperatures. And also the weird feature off the coast of Florida is gone.\n\n2 - Describe the differences that
you see in your contoured analyses.  \n    Does one analysis seem to be smoother than the other?  If so, what w
ould cause this?\n    \n    In the 10 cm radius of influence analyses map, the contours are closer together whe
n compared to the ROI of 20 cm.\n    The ROI = 20 cm analyses map looks smoother when compared to the 10 cm on
e.\n    When looking at the ROI = 20 observation map, the contours are more smoothed and circular compared to t
he ROI = 10cm.\n    The ROI = 20 map is using a larger area and more observational points, so more points are i
n the analysis, which allows\n    everything to be smoothed out compared to when there are not a lot of points,
meaning if something is bad, it doesnt go away.\n\n3 - Run your program using a radius of influence of 6 cm (do
not need to show).  (put 6 in array, will look funky)\n    Describe the results - do they look realistic?  If t
here are problems, what\n    do you think might be causing them?\n    \n    The results from doing this are not
realistic. One of the main problems is the distorition, especially near the coastlines and \n    over Canada. G
oing back to the previous questions, I think that if you have less points (observations), the analysis will not
\n    be as smooth, causing problems and distortions, similar to the small distortions when using the ROI = 10c
m. With the limited\n    amount of observation points, the results will not be as smoothed out, causing localiz
ed problems to be more enhanced.\n\n4 - Suppose you ran this program with a small enough radius of influence th
at only one\n    observation was available for determining a polynomial fit at a grid point.  Should\n    you b
e able to perform the matrix inversion?  Why or why not?\n    \n    If only one observation was available, then
that would mean that your matrix would just have the value "1" inside. Looking back to \n   the previous calcul
ations and methods above, the only time you would run into any issue is because you are dividing, with the ob\n
   matrix in the denominator. The equation is R_avg = R_matrix/obs_matrix[j,k,i] and O_avg = O_column/obs_matri
x[j,k,i]. When dividing\n   by "1", that math works, so it shouldn\'t be an issue. So YES, you should be able t
o perform it.\n    \n'
```