# Using MATLAB to Create 3-D Sounding Plots

**Updated: 26 April 2024**

**Authorship:** NSF NCAR Earth Observing Laboratory Data Management & Services Group
**Contact:** eol-archive@ucar.edu



**Acknowledgment:** This work was sponsored by the National Science Foundation. This material is based upon work supported by the National Center for Atmospheric Research, a major facility sponsored by the National Science Foundation and managed by the University Corporation for Atmospheric Research. Any opinions, findings and conclusions or recommendations expressed in this material do not necessarily reflect the views of the National Science Foundation.

## General Description

This document serves as a guide on how MATLAB can be used with OPeNDAP and txt files within the EOL Field Data Archive to generate 3-D plots of sounding variables. Two examples of these MATLAB sounding plotting scripts can be found in the files "**matlab_sounding_txt_xyz.m**" and "**matlab_sounding_nc_xyz.m**". The scripts were created by NSF NCAR EOL and show how 3D plots can be created in MATLAB. Note that these scripts are provided "as is" and are not supported.

## Getting Data from Files

The scripts "**txt_xyz.m**" and "**nc_xyz.m**" use txt and nc files respectively. The process for getting the needed data is different for these types of files. The data used for the "**txt_xyz.m**" script came from the following dataset:

**WINTRE-MIX: Multi-Network Composite Highest Resolution Radiosonde Data**
https://data.eol.ucar.edu/dataset/612.041

The information used in this script came from this file in the dataset:

**WINTRE-MIX_2022_HighRes_20220312.cls**

The data file contains various information for multiple stations in the WINTRE-MIX field project. The data used from this file was 4 stations at/around 12 March 2022 at time 0200 hours. The information for these 4 stations were exported to 4 separate text files, one for each station.

```
% Create a variable for local file. Use readtable() to get the data as
% an array of numbers.

myFile1 = readtable('Plattsburg.txt');
myFile2 = readtable('Gault.txt');
myFile3 = readtable('Jean.txt');
myFile4 = readtable('Sorel.txt');
```

**Figure 1**: This image shows how the files were added into the script.

Figure 1 shows how the files are read in MATLAB. The “readtable” command is used to read the data in the file in a usable format. This command looks at all elements of the file, including the delimiters and data types. Based on this information, the command then imports the data into a table.

```
% Create variables and get the temperature, latitude, longitude, and
% altitude from the columns in the txt file.

temp1 = table2array(myFile1(:,3));
lat1 = table2array(myFile1(:,12));
long1 = table2array(myFile1(:,11));
alt1 = table2array(myFile1(:,15));
```

**Figure 2**: The files used had the data organized into columns. Use the table2array command to convert from table to an array.

The “readtable” command reads the file and stores it as the “table” data type. Because of this, the data is unusable in its current state. Using the “table2array” command converts from the “table” data type to a traditional data type (double, string, etc.). For this example, the data is organized by column.

For netCDF (i.e., nc) files, exporting the data uses a different procedure. The data used for the “**nc_xyz.m**” was from the following datasets:

**SWEX: ISS Radiosonde Data - Rancho Alegre Site**
https://data.eol.ucar.edu/dataset/edit/600.003

**SWEX: ISS Radiosonde Data - Sedgwick Site**
https://data.eol.ucar.edu/dataset/edit/600.004 (one file was used from this dataset “NCAR_SWEX2022_ISS3_RS41_v1_20220331_221902.nc”)

```
% Create a variable for the opendap url or for the local file.

myFile1 = 'NCAR_SWEX2022_ISS2_RS41_v1_20220404_170007.nc';
myFile2 = 'NCAR_SWEX2022_ISS2_RS41_v1_20220317_221912.nc';
myFile3 = 'NCAR_SWEX2022_ISS2_RS41_v1_20220405_020004.nc';
myFile4 = 'NCAR_SWEX2022_ISS2_RS41_v1_20220406_020007.nc';
myFile5 = 'NCAR_SWEX2022_ISS2_RS41_v1_20220409_230219.nc';
myFile6 = 'NCAR_SWEX2022_ISS2_RS41_v1_20220413_170027.nc';
myFile7 = 'NCAR_SWEX2022_ISS2_RS41_v1_20220414_020003.nc';
myFile8 = 'NCAR_SWEX2022_ISS2_RS41_v1_20220417_170014.nc';
myFile9 = 'NCAR_SWEX2022_ISS2_RS41_v1_20220418_020014.nc';
myFile10 = 'NCAR_SWEX2022_ISS2_RS41_v1_20220513_020001.nc';
```

**Figure 3**: These were the files used for the "**nc_xyz.m**" script. Local files or urls can be used for netCDF files. The url can be found from the OPeNDAP feature on select datasets on the EOL Field Data Archive website at data.eol.ucar.edu.

The "ncdisp" command is used to extract the variables needed. In the Command Window, the command "ncdisp(myFile1)" was used to get the data (the "ncdisp" command can be used in the script. This is not recommended as it will fill large amounts of space in the Command Window each time the script is run).

```
% Use the 'ncread' command to take the desired variables. For this example
% latitude, longitude, altitude, and air temperature were
% used.
lat1 = ncread(myFile1, 'lat');
long1 = ncread(myFile1, 'lon');
alt1 = ncread(myFile1, 'alt');
air_temp1 = ncread(myFile1, 'tdry');
```

**Figure 4**: For netCDF files, the data is read through the "ncread" command.

In MATLAB, the "ncread" command is used to read the data from the file. The "ncread" command requires 2 fields. The first field is the file and the second field is the variable tag. This tag is found from the "ncdisp" result.

```
lat
       Size:       4968x1
       Dimensions: time
       Datatype:   single
       Attributes:
                   long_name     = 'north latitude'
                   units         = 'degree'
                   missing_value = -999
                   _FillValue    = -999
                   valid_range   = [-90  90]
                   standard_name = 'latitude'
                   axis          = 'X'
```

**Figure 5**: The "lat" variable tag is found from using the "ncdisp" command.

After using the "ncdisp" command, the Command Window will output various information about the nc file. One section of the Command Window should be dedicated to variables. Under the variable section, the output might look something similar to what is shown in Figure 5 above.

The variable tag is the top line that says “lat”. The “Attributes” section goes in depth about the variable. The “long_name” and “standard_name” fields help explain the field and can help differentiate variables in case there are multiple latitude fields. The format of this output will vary on the file.

After loading the data as arrays, plotting the data is the same for both text (.txt) and netCDF (*.nc) files.

```
% % Plot the data as scatter plots.

scatter3(lat1,long1,alt1,50,temp1,'filled');

% The text() function is used to add some context for each scatter plot.
% The FontSize and HorizontalAlignment were adjusted for clarity.
% The set() function is used to rotate the text for clarity.
t1 = text(lat1(1479), long1(1479), alt1(1479)+500, 'Plattsburg');
t1.FontSize = 10;
t1.HorizontalAlignment = "left";
set(t1,'Rotation',60);
```

**Figure 6**: The data was plotted with a 3D scatter plot.

Figure 6 shows how the data was plotted for these examples. “scatter3” is the command for a 3D scatter plot in MATLAB. The inputs for “scatter3” are:

scatter3(x-variable, y-variable, z-variable, marker size, color, marker type)

In this example, the first 3 variables are latitude, longitude, and altitude, respectively. The marker size was 50. The air temperature variable was used to color code each point in the scatter plot. “filled” was used for the scatter plot (“filled” means filled in points). These fields can be adjusted as necessary.

The “text()” function was used to add text to the plots. This function needs the location and the text to work. In this case, the last point in the array was used with a small offset in the z direction. The offset was used for clarity in reading the text. The font size, alignment, and rotation were adjusted for clarity.

```
% Add a black dot to see the origin of each plot for clarity.

plot3(lat1(1),long1(1),alt1(1),'k.', 'MarkerSize', 50);
plot3(lat2(1),long2(1),alt2(1),'k.', 'MarkerSize', 50);
plot3(lat3(1),long3(1),alt3(1),'k.', 'MarkerSize', 50);
plot3(lat4(1),long4(1),alt4(1),'k.', 'MarkerSize', 50);
```

**Figure 7**: For clarity, a black dot was added to the first point in the location arrays. This dot represents the station location.

```
% Additional plot features.

grid on
shading interp
xlabel('latitude')
ylabel('longitude')
zlabel('altitude (m)')
title("WINTRE-MIX IOP10, March 12, 2022 02:00")

c = colorbar;
set( c, 'YDir', 'reverse' );
c.Label.String = ([char(176) 'C']);
```

**Figure 8**: Other plot features like the title were added. (The char(176) refers to the degree symbol which is used in the colorbar).

Other metadata was added for the plot. These features can be adjusted. For this example, there is a grid in the background of the plot. The shading was set as interpolated. Labels were added for each axis. The title was added. The colorbar was created using the air temperature variable from earlier. Figures 9 and 10 below show the resulting plots from text and netCDF files, respectively. In MATLAB, after creating the plots, a window will open up for the plot. The plot can be rotated.

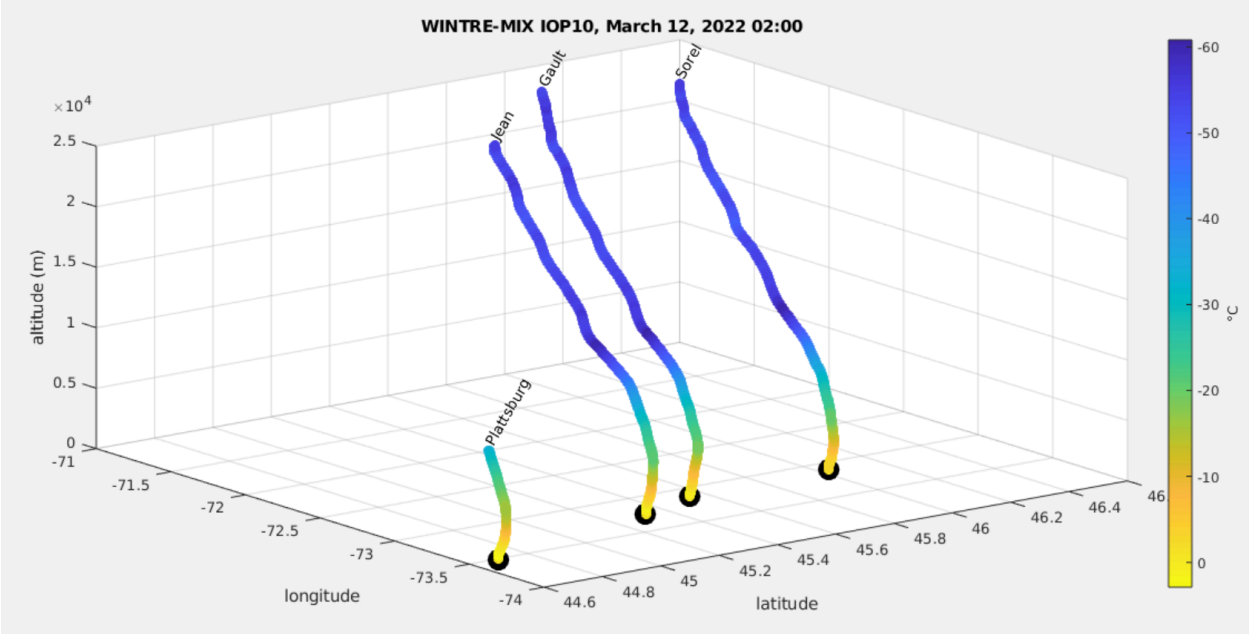


**Figure 9**: The plot for the sounding data files in text format.

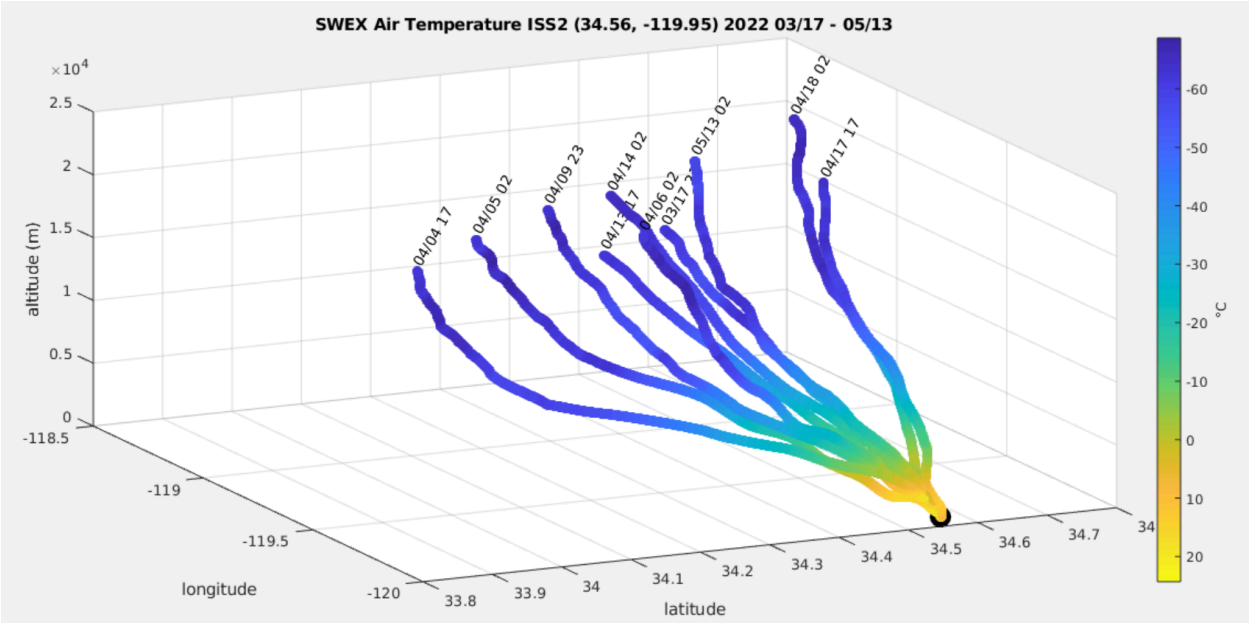


**Figure 10**: The plot for the sounding data files in NetCDF format.