Using MATLAB to View OPeNDAP Files in the NCAR/EOL Field Data Archive

Updated: 3 February 2022

Authorship: 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.

Note: There is an issue with **MATLAB R2021b**. If you are running a matlab program on a linux computer and you receive the following error:

"Error using matlab.internal.imagesci.netcdflib"

try creating a **.dodsrc** file in the directory where the matlab program exists that contains the following line:

HTTP.SSL.CAINFO=/etc/ssl/certs/ca-bundle.crt

Then restart matlab and try running the program again. It appears that something changed in the MATLAB code of MATLAB R2021b that changed the way it uses system certificates.

General Description

This document serves as a guide on how MATLAB can be used with OPeNDAP within the EOL Field Data Archive to plot variables found in files with OPeNDAP access without needing to download the files. Two examples of matlab plotting scripts can be found in the files "**matlab_xy_test.m**" and "**matlab_xyz_test.m**". The scripts were created by NCAR EOL and show how 2D and 3D plots can be created in matlab, respectively. Note that these scripts are provided "as is" and are not supported. For any questions about MATLAB features, functions, etc. refer to the MathWorks help center at:

https://www.mathworks.com/help/index.html

OPeNDAP within the EOL Field Data Archive

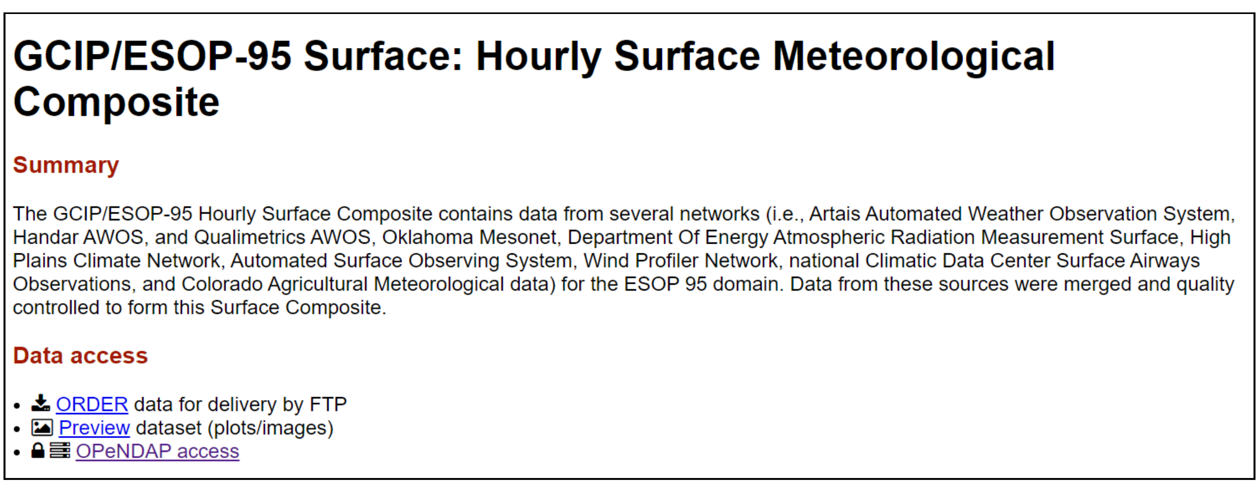
The EOL Field Data Archive (FDA) contains datasets that have OPeNDAP compatibility. An example dataset in the FDA would be GCIP/ESOP-95 Surface: Hourly Surface Meteorological Composite. This dataset was used to generate the plots in the sections below.

For this dataset, 2D plots were created for a one site (Ashton) for Temperature and Dew Point using script **matlab_xy_test.m**. See Figure 22. A 3D plot was also created using matlab_xyz_test.m that shows Temperature specific time over the entire area of interest in the file (See figure 29).

How to use the Matlab Plot tools with the OPeNDAP Capabilities in the EOL Field Data Archive

Choose a dataset in the EOL FDA that has OPeNDAP Capabilities

Figure 1 shows the top portion of the EOL FDA dataset description page for the GCIP/ESOP-95 Surface: Hourly Surface Meteorological Composite dataset. The OPeNDAP feature is located in the "Data access" section of the dataset page. Clicking on the "OPeNDAP access" link will show the OPeNDAP page for the dataset. This page will show a list of all the files and the dataset and their corresponding OPeNDAP links. If a dataset in the EOL FDA does not have an OPeNDAP link in the "Data access" section, then the dataset does not have OPeNDAP capabilities. The user must then download the data (via the Order link) and apply the provided matlab scripts to plot the data.



GCIP/ESOP-95 Surface: Hourly Surface Meteorological Composite

Summary

The GCIP/ESOP-95 Hourly Surface Composite contains data from several networks (i.e., Artais Automated Weather Observation System, Handar AWOS, and Qualimetrics AWOS, Oklahoma Mesonet, Department Of Energy Atmospheric Radiation Measurement Surface, High Plains Climate Network, Automated Surface Observing System, Wind Profiler Network, national Climatic Data Center Surface Airways Observations, and Colorado Agricultural Meteorological data) for the ESOP 95 domain. Data from these sources were merged and quality controlled to form this Surface Composite.

Data access

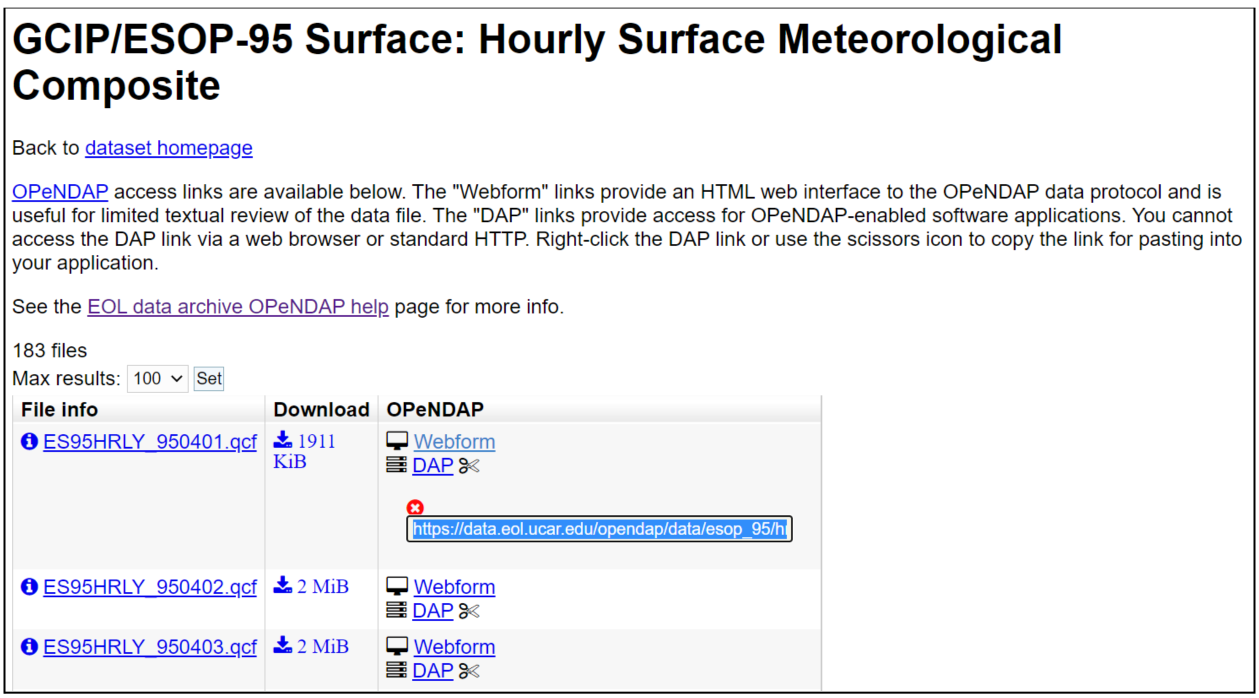
- ORDER data for delivery by FTP
- Preview dataset (plots/images)
- OPeNDAP access

Figure 1: Top portion of the dataset description page for the EOL GCIP/ESOP-95 Surface: Hourly Surface Meteorological Composite dataset. The OPeNDAP feature for datasets (if applicable) is located in the "Data access" section of the dataset description page.

Get the Link to the File for Plotting

Figure 2 shows the "OPeNDAP access" page for the EOL GCIP/ESOP-95 Surface: Hourly Surface Meteorological Composite dataset. This page lists all files in the dataset. From here the OPeNDAP link files can be found. Clicking the "scissors" symbol will show the OPeNDAP link for a specific file. This link can then be fed into the MATLAB scripts for plotting. For the plots shown in this document, file "ES95HRLY_950715.qcf" was used and the OPeNDAP link for that file is:

https://data.eol.ucar.edu/opendap/data/esop_95/hrly_sfc/ES95HRLY_950715.qcf



GCIP/ESOP-95 Surface: Hourly Surface Meteorological Composite

Back to dataset homepage

OPeNDAP access links are available below. The "Webform" links provide an HTML web interface to the OPeNDAP data protocol and is useful for limited textual review of the data file. The "DAP" links provide access for OPeNDAP-enabled software applications. You cannot access the DAP link via a web browser or standard HTTP. Right-click the DAP link or use the scissors icon to copy the link for pasting into your application.

See the EOL data archive OPeNDAP help page for more info.

183 files

Max results: 100 Set

File info	Download	OPeNDAP
ES95HRLY_950401.qcf	1911 KiB	Webform DAP https://data.eol.ucar.edu/opendap/data/esop_95/h
ES95HRLY_950402.qcf	2 MiB	Webform DAP
ES95HRLY_950403.qcf	2 MiB	Webform DAP

Figure 2: The OPeNDAP page for the EOL GCIP/ESOP-95 Surface: Hourly Surface Meteorological Composite dataset. Clicking on the "scissors" symbol for a file will show the OPeNDAP link for that specific file.

Store the Data File Link in the MATLAB script

In the MATLAB script, this link to the data file should be stored in a variable as shown in Figure 3 below. Figure 3 shows how the file link is stored in the provided sample scripts. (Note: All of the MATLAB figures come from the sample script "matlab_xy_test.m".) The variable "myFile" stores the link to the data file. If the file was downloaded locally onto the users system, the full path to the file could be included as well. If the file is located in the same directory as the script, the following line can be used instead:

myFile = ES95HRLY_950715.qcf

If the file is located in a different directory, this line can be used instead:

myFile = /directory1/directory2/ES95HRLY_950715.qcf

In the line above, “directory1” and “directory2” are placeholders indicating the full system path to the data file. Replace these directories with the proper path where the file is located on the machine. After the file link is stored, the data in the file can be displayed.

```
35    % Create the variable to store the opendap url or for the local file.
36
37    myFile='https://data.eol.ucar.edu/opendap/data/esop_95/hrly_sfc/ES95HRLY_950715.qcf';
38    %myFile='ES95HRLY_950715.qcf';
39
```

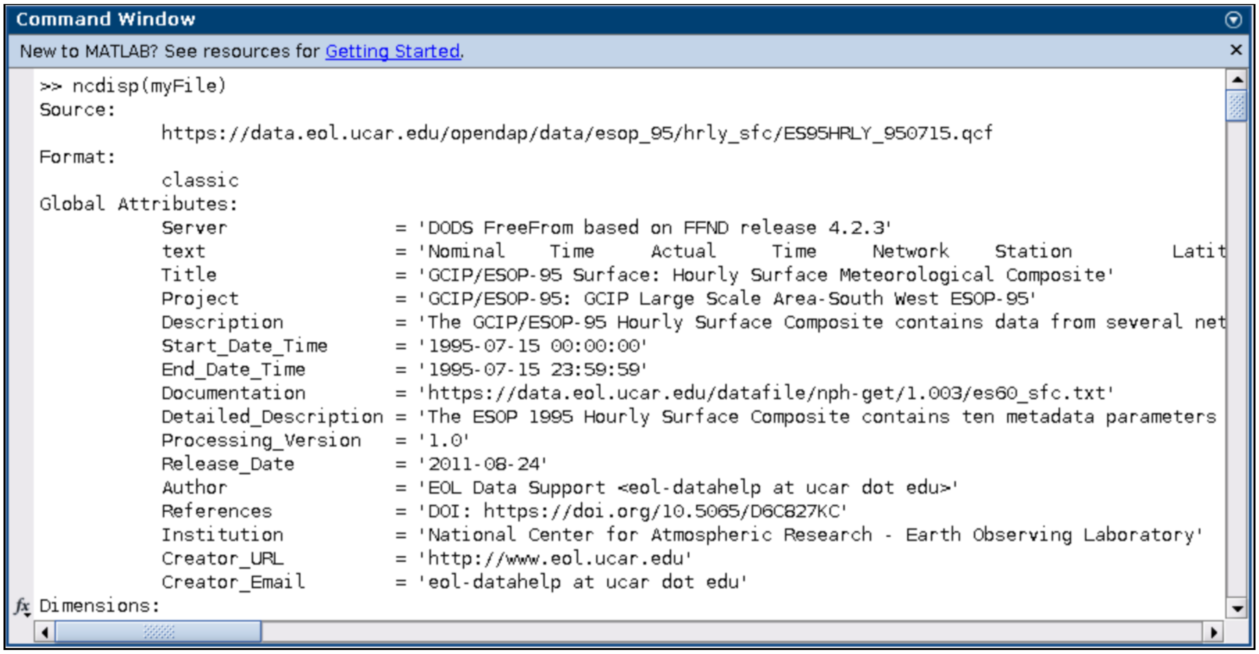
Figure 3: The link for the file should be stored in a variable in the MATLAB script. In this example, the link is stored in a variable named “myFile” as seen in line 37.

MATLAB ncdisp Command

The MATLAB command “ncdisp” is used to view the metadata and the variables in a file. Figures 4-6 show what is displayed in the Command Window when the “ncdisp” command is executed. The metadata and variables in the data file can be determined this way. These variable names are then used in the “ncread” MATLAB commands to retrieve and plot specific variables. See the “ncread” command example in Figure 7.

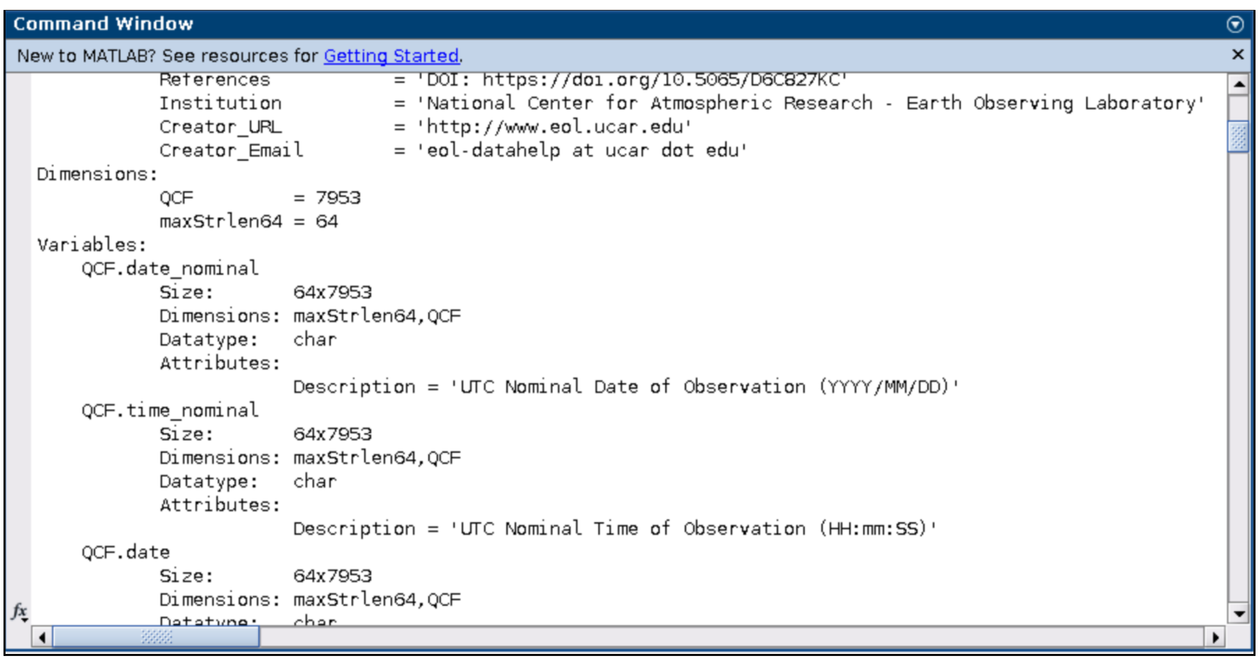
```
% Display the information in the netCDF file with 'ncdisp'.

ncdisp(myFile)
```

Figure 4: The “ncdisp” command is used to view the file in the Command Window in MATLAB.



```
Command Window
New to MATLAB? See resources for Getting Started.
>> ncdisp(myFile)
Source:
           https://data.eol.ucar.edu/opendap/data/esop_95/hrly_sfc/ES95HRLY_950715.qcf
Format:
           classic
Global Attributes:
           Server                = 'DODS FreeFrom based on FFND release 4.2.3'
           text                  = 'Nominal    Time     Actual     Time     Network    Station        Latit
           Title                 = 'GCIP/ESOP-95 Surface: Hourly Surface Meteorological Composite'
           Project               = 'GCIP/ESOP-95: GCIP Large Scale Area-South West ESOP-95'
           Description           = 'The GCIP/ESOP-95 Hourly Surface Composite contains data from several net
           Start_Date_Time       = '1995-07-15 00:00:00'
           End_Date_Time         = '1995-07-15 23:59:59'
           Documentation         = 'https://data.eol.ucar.edu/datafile/nph-get/1.003/es60_sfc.txt'
           Detailed_Description = 'The ESOP 1995 Hourly Surface Composite contains ten metadata parameters
           Processing_Version    = '1.0'
           Release_Date          = '2011-08-24'
           Author                = 'EOL Data Support <eol-datahelp at ucar dot edu>'
           References            = 'DOI: https://doi.org/10.5065/D6C827KC'
           Institution           = 'National Center for Atmospheric Research - Earth Observing Laboratory'
           Creator_URL           = 'http://www.eol.ucar.edu'
           Creator_Email         = 'eol-datahelp at ucar dot edu'
fx Dimensions:
```

Figure 5: After the “ncdisp” command is executed, the contents of the Command Window in MATLAB should look like this. The Command Window will show the metadata of the file under the “Source”, “Format”, “Global Attributes”, and “Dimensions” section.



```
Command Window
New to MATLAB? See resources for Getting Started.
           References            = 'DOI: https://doi.org/10.5065/D6C827KC'
           Institution           = 'National Center for Atmospheric Research - Earth Observing Laboratory'
           Creator_URL           = 'http://www.eol.ucar.edu'
           Creator_Email         = 'eol-datahelp at ucar dot edu'
Dimensions:
           QCF          = 7953
           maxStrlen64 = 64
Variables:
    QCF.date_nominal
           Size:        64x7953
           Dimensions: maxStrlen64,QCF
           Datatype:    char
           Attributes:
                        Description = 'UTC Nominal Date of Observation (YYYY/MM/DD)'
    QCF.time_nominal
           Size:        64x7953
           Dimensions: maxStrlen64,QCF
           Datatype:    char
           Attributes:
                        Description = 'UTC Nominal Time of Observation (HH:mm:SS)'
    QCF.date
           Size:        64x7953
           Dimensions: maxStrlen64,QCF
fx         Datatype:    char
```

Figure 6: After the metadata of the file, the section for “Variables” can be found. This section shows the variables for the file and the metadata for each variable can be found.

MATLAB ncread Command

The data for a variable can be saved in the MATLAB script by using the “ncread” command. Figure 7 shows how the “ncread” command is used. The data file and variable name are both needed. In this example, the nominal date is read from “myfile” into the “date” parameter. The exact variable name from the data file (i.e., “QCF.date_nominal”) must be used in the ncread command. Use the ncdisp command to determine the exact variable names. This process can then be repeated for all variables to be plotted.

```

date = ncread(myFile, 'QCF.date_nominal');

```

Figure 7: The “ncread” command is used to read and store the data from the input file for a given variable. In this case, the nominal date is read. Notice in Figure 6 that the variable name for the nominal date is “QCF.date_nominal”. This variable name is used in the “ncread” command.

Preprocessing of Input Data

Depending on the input file and data used from OPeNDAP, some preprocessing may be necessary before plotting the data. The following information explains the processing used in the sample input file mentioned in this document. Processing may vary depending on the file used.

Figures 8 and 9 show what the input data looks like before and after preprocessing. In Figure 8, the nominal date is stored with each character in its own cell with trailing whitespace. The whitespace needs to be removed and the nominal date should be one string as shown in Figure 9.

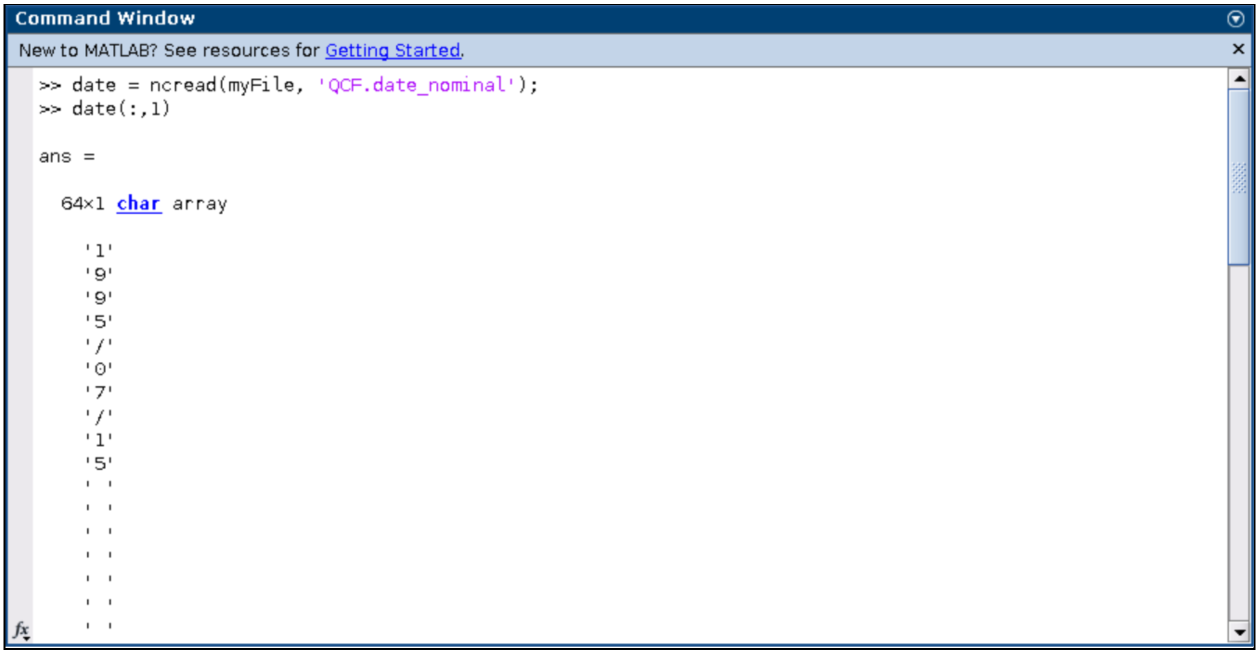


Figure 8: Depending on the file used, some preprocessing of the data may be needed prior to plotting. In the image above, the input form of the nominal date is stored in a column with each individual character in a different cell with trailing whitespace.

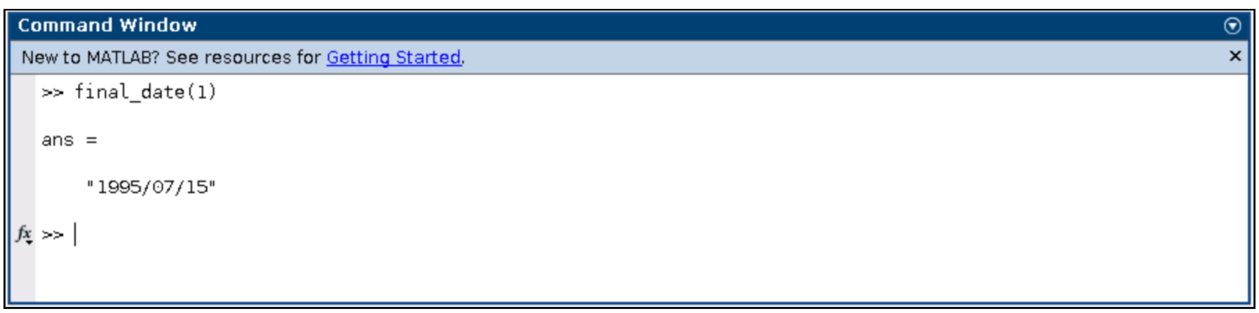


Figure 9: After preprocessing, the nominal date is now stored as a string in one cell with the white space removed.

In this example, some variables like the nominal date are stored as matrices. These variables need to be rearranged into arrays in order for plotting to be possible. Figures 10 and 11 show the process needed for the nominal date. The arrays are created for the nominal date and the whitespaces are removed. This process is repeated for the nominal time in the sample MATLAB scripts. The nominal date and nominal time are combined into the datetime variable as shown in Figure 12.

```
% Find the dimensions of the 'date' matrix.

[rows,cols] = size(date);

% Create an array of the type 'string'.

final_date = strings(cols,1);
```

Figure 10: The matrix for the nominal date needs to be rearranged into an array for the plots. The dimensions for the matrix are needed.

```
74  % This loop takes every 'char' column from the matrix and stores is as a
75  % 'string' into the array.
76
77  c = 1;
78  while c <= cols
79      final_date(c,1) = string([date(:,c)].');
80      final_time(c,1) = string([Time(:,c)].');
81      c = c + 1;
82  end
83
84  % The empty elements of the array are removed.
85
86  final_date(:,1) = deblank(final_date(:,1));
87  final_time(:,1) = deblank(final_time(:,1));
88
```

Figure 11: A loop is created to take each column of characters shown in Figure 8 and arrange them into one string shown in Figure 9 and place them into an array. The nominal time also needs to be processed like this. Lines 77-82 of the sample code show this process. After the array is filled, the whitespaces need to be removed. Lines 86-87 remove the whitespaces from the date and time.

```
% The date and time are then used to create the datetime.

date_time_arr = final_date + ' ' + final_time;
date_time = datetime(date_time_arr,'InputFormat','yyyy/MM/dd HH:mm:ss');
u_datetime = unique(date_time);
u_date = unique(final_date);
u_time = unique(final_time);
```

Figure 12: The nominal date and time are combined into a single datetime. All repeating datetimes are also removed.

Up to this point in the code, the process for “matlab_xy_test.m” and “matlab_xyz_test.m” scripts have been the same, but the code for the 2D plot and 3D plot scripts diverge after this. The following section describes the procedure specifically for the 2D “matlab_xy_test.m” script. Both the temperature and dew point variables are plotted on the sample 2D plot.

For the “matlab_xy_test.m” script, the variables for the dew point temperature, air temperature, network, and station/platform are used. Variables like the temperature did not need preprocessing as they are already stored as arrays. The network and station/platform need the preprocessing as done for the nominal date data. See Figures 13-16.

```
% Read the dew point and air temp.

dew_point = ncread(myFile, 'QCF.dew_point_temperature');
air_temp = ncread(myFile, 'QCF.air_temperature');
```

Figure 13: Some variables like the dew point temperature and air temperature in the example do not need processing. The temperatures are already in arrays.

```
% Read in the network and repeat the process above.

Network =ncread(myFile, 'QCF.network_name');

[netrows,netcols] = size(Network);

final_net = strings(netcols,1);

c = 1;
while c <= cols
    final_net(c,1) = string([Network(:,c)].');
    c = c + 1;
end

final_net(:,1) = deblank(final_net(:,1));
```

Figure 14: The process of putting the metadata into an array is repeated for the ”network” variable.

```
% Read in the platform and repeat the process.

Station =ncread(myFile, 'QCF.platform_name');

[starows,stacols] = size(Station);

final_sta = strings(stacols,1);

c = 1;
while c <= cols
    final_sta(c,1) = string([Station(:,c)].');
    c = c + 1;
end

final_sta(:,1) = deblank(final_sta(:,1));
```

Figure 15: The process of putting the metadata in an array is repeated for the “platform/station” variable.

```
% Combine the network and platform to make the location.

location = final_net + final_sta;

% Remove the duplicate locations.

unique_location = unique(location);
```

Figure 16: The network and station/platform are combined to create the location. Duplicate locations are then removed.

Plotting the Data with the 2D Matlab Plot Script

In the input data file used for this example, the data for multiple stations/sites is sorted by date and time and then by station, so all data collected for all stations at a specific time are listed together for each date/time. Here is a sample from the input data file:

```
Nominal    Time     Actual     Time     Network    Station         Latitude  Longitude  Occ Elev    STN     Q Sea Lvl Q Cmptd   Q Drybulb Q Dewpnt  Q Wind    Q Wind    Q Total
g CF Q CA Q
Date       UTC      Date       UTC                 ID                                   ur          Press   F Press   F Sea Lvl F Temp    F Temp    F Speed   F Dir     F Prcp
3    F    F
xxxxxxxxxx xxxxxxxx xxxxxxxxxx xxxxxxxx xxxxxxxxxx xxxxxxxxxxxxxxx xxxxxxxxxx xxxxxxxxxxx xxx xxxxxxx xxxxxxx x xxxxxxx x xxxxxxx x xxxxxxx x xxxxxxx x xxxxxxx x xxxxxxx x xxxxxxx
x xx x xx x
1995/07/15 00:00:00 1995/07/15 00:00:00 AWOSQ20    ILE                 31.08000   -97.69000   0  256.00  983.80 G -999.99 N 1012.20 G   32.75 G   18.85 G    3.60 G  120.00 G    0.00
9 15 N 15 N
1995/07/15 00:00:00 1995/07/14 23:55:00 NCDC       ILE                 31.08333   -97.68333   0  257.00  983.70 G -999.99 M 1012.10 G   33.35 G   18.95 G    4.10 G  100.00 G -999.99
9 15 N 15 N
1995/07/15 00:00:00 1995/07/15 00:00:00 AWOSQ20    TPL                 31.15000   -97.41000   0  208.00 -999.99 M -999.99 N -999.99 M -999.99 M -999.99 M -999.99 M -999.99 M    0.00
9 15 N 15 N
1995/07/15 00:00:00 1995/07/14 23:56:00 ASOSH      LFK                 31.23000   -94.75000   0   88.00 1004.70 U -999.99 N -999.99 M -999.99 M -999.99 M -999.99 M -999.99 M    0.00
9 15 M 15 M
1995/07/15 00:00:00 1995/07/14 23:51:00 NCDC       LFK                 31.23333   -94.75000   1   86.00 1005.00 G 1014.50 G 1014.60 G   29.45 G   22.85 D    3.60 G  130.00 G -999.99
9 15 N 15 N
1995/07/15 00:00:00 1995/07/14 23:51:00 NCDC       AEX                 31.33333   -92.55000   0   27.00 1013.00 G -999.99 M 1016.10 G   23.35 D   22.25 G    1.00 G  360.00 G -999.99
9 15 N 15 N
1995/07/15 00:00:00 1995/07/14 23:50:00 NCDC       SJT                 31.36667  -100.50000   1  580.00  946.70 G 1011.10 G 1009.70 G   34.45 G   14.45 G    5.70 G  140.00 G -999.99
9 15 N 15 N
1995/07/15 00:00:00 1995/07/14 23:56:00 ASOSH      SJT                 31.37000  -100.50000   0  584.00  945.60 U -999.99 N -999.99 M -999.99 M -999.99 M -999.99 M -999.99 M    0.00
9 15 M 15 M
1995/07/15 00:00:00 1995/07/14 23:56:00 ASOSH      ESF                 31.40000   -92.30000   0   34.00 1013.20 G 1016.70 G 1017.10 G   22.25 B   21.65 G    1.50 G   60.00 D    0.00
9 15 N 15 N
1995/07/15 00:00:00 1995/07/14 23:50:00 NCDC       ESF                 31.40000   -92.30000   1   33.00 1013.30 G 1016.80 G 1017.10 G   22.85 B   21.75 G    2.60 G   10.00 D -999.99
9 15 N 15 N
1995/07/15 00:00:00 1995/07/15 00:00:00 AWOSQ20    OCH                 31.57000   -94.71000   0  108.00 1001.60 G -999.99 N 1013.80 G   30.05 G   20.55 D    3.10 G  140.00 G    0.00
9 15 N 15 N
1995/07/15 00:00:00 1995/07/15 00:00:00 AWOSQ20    HEZ                 31.61000   -91.30000   0   83.00 1007.30 G -999.99 N 1016.90 G   22.75 G   21.15 G    1.50 G  210.00 G    0.30
9 15 N 15 N
1995/07/15 00:00:00 1995/07/14 23:56:00 NCDC       ACT                 31.61667   -97.21667   0  152.00  995.40 G 1012.40 G 1012.40 G   31.75 G   20.65 G    5.10 G  100.00 G -999.99
9 15 N 15 N
1995/07/15 00:00:00 1995/07/14 23:56:00 NCDC       HEZ                 31.61667   -91.28333   0   83.00 1007.30 G -999.99 M 1016.90 G   22.85 G   21.75 G    1.50 G  230.00 G -999.99
9 15 N 15 N
1995/07/15 00:00:00 1995/07/14 23:56:00 ASOSH      ACT                 31.62000   -97.22000   0  157.00  994.80 G 1012.40 G 1012.40 G   31.65 G   20.55 G    5.10 G  100.00 G    0.00
9 15 N 15 N
1995/07/15 00:00:00 1995/07/15 00:00:00 WPDN       Palestine           31.77917   -95.71333   0  119.00 -999.99 N 1013.70 U -999.99 N   28.05 U   22.95 U -999.99 M -999.99 M    0.00
9 15 N 15 N
1995/07/15 00:00:00 1995/07/14 23:50:00 NCDC       INK                 31.78333  -103.20000   0  857.00  915.10 G 1008.90 G 1006.80 G   34.45 G   11.15 G    6.20 G  110.00 G -999.99
9 15 N 15 N
1995/07/15 00:00:00 1995/07/14 23:56:00 NCDC       ELP                 31.80000  -106.40000   1 1199.00  878.80 D 1006.40 D 1006.10 D   31.75 D   10.65 G    0.00 D    0.00 D -999.99
9 15 N 15 N
1995/07/15 00:00:00 1995/07/14 23:56:00 ASOSH      ELP                 31.80000  -106.40000   0 1198.00  878.80 D 1006.40 D 1006.10 D   31.65 D   10.55 G    0.00 D    0.00 D    0.00
9 15 N 15 N
1995/07/15 00:00:00 1995/07/14 23:56:00 ASOSH      E02                 31.92000  -102.38000   0  915.00  909.60 G 1010.70 G 1007.60 G   33.35 G   13.35 G    6.70 G  110.00 G    0.00
9 15 N 15 N
```

To plot a particular station's data, code must be included to pull only that station's data into a plottable array. **Figures 13-21** show the procedure for preparing the input data from the specific sample file for the 2D "matlab_xy_test.m" script. Figure 22 shows the final result of the script.

```
% Use a for loop to repeat the process for any number of locations.

for a = 1:2

% Take the 'a' location and use it later in the loop.

example_location = unique_location(a);

% The information in this file is organized by time. The following lines
% here gathers the information for the given location. The array "array"
% stores the indices where the chosen station information is located.

time_size = length(final_time);
array = zeros(time_size,1);
array(1) = 1;
```

Figure 17: The "for" loop shown above is used for plotting the data for a unique station location. In this example, the first 2 locations are plotted. Originally, all of the data in the file is organized by time. Because of this, the values for one station are scattered throughout the file. The loop takes the site location wanted for plotting and then compares the station to every other site location in the file. The index for every instance of the site location matching in the file as this would show where all the temperature, dew point temperature, etc. are located in the file.

```
inc = 2;

for i = 1:time_size

    % The script saves the current network, station, and occurrence for the
    % current iteration of the loop.

    current_station = location(i);

    % If the location information is the same as the current station, save
    % the index into the array. Note that "..." means continue the code on
    % a different line.

    if isequal(current_station, example_location) && ...
            (final_time(1)~=final_time(i))

        array(inc) = i;
        inc = inc + 1;
    end
 end
```

Figure 18: The loop shown above shows how all the indices for one location are found. The unique site location saved in Figure 17 is then compared to all of the other station locations in the location array.

```
% The "array" variable contains the indices but also contains unnecessary
% zeros at the end. The following loop removes the ending zeros
% Because the previous for loop icremented the "inc" one
% more than needed, this loop subtracts the "inc" by 1.

final_arr = zeros(inc-1,1);
for j = 1:(inc-1)
    final_arr(j) = array(j);
end

% With the given indices, gather the time, temperature, and dew point
% temperature that correspond to the given location.

final_temp = zeros(length(final_arr),1);
final_dew_temp = zeros(length(final_arr),1);
```

Figure 19: The loop in Figure 18 only stores the indices where there are matches with the unique site location being plotted. The array that stores these indices was initialized as an array full of 0s. Because there are many site locations in the array, many did not match with the unique site being plotted leaving 0s in the index array. A final array is created to remove these 0s. The dimensions of this array are then used to create the arrays for the temperature and dew point temperature.

```
% The file contains values of -999 temp. Change these values to 'NaN' so it
% does not plot these values.

for k = 1:(length(final_arr))
    if air_temp(final_arr(k)) < -100
        final_temp(k) = NaN;
    end
    if air_temp(final_arr(k)) > -100
        final_temp(k) = air_temp(final_arr(k));
    end
    if dew_point(final_arr(k)) < -100
        final_dew_temp(k) = NaN;
    end
    if dew_point(final_arr(k)) > -100
        final_dew_temp(k) = dew_point(final_arr(k));
    end

end
```

Figure 20: The temperatures in the file may have a missing value of -999. These values are changed to “Not a Number” values or NaN so they will not be plotted. Matlab ignores NaN values for plotting.

```
% Plot the data. x axis is date time and y axis is temperature in Celsius.

figure(a)
plot(u_datetime,final_dew_temp,'b','DisplayName','dew point temperature')
hold on;
scatter(u_datetime,final_dew_temp,'b', 'DisplayName', 'dew point temperature')
hold on;
plot(u_datetime,final_temp,'r', 'DisplayName', 'air temperature')
hold on;
scatter(u_datetime,final_temp,'r', 'DisplayName', 'air temperature' )

% xlabel('date time ')
ylabel('°C')
%legend({'dew point temp', 'air temp'}, 'Location', 'northwest')
legend('Location', 'northwest')
title("GCIP/ESOP-95 Surface Temp UTC" + strjoin(split(example_location)))
```

Figure 21: The data is then plotted and the resulting 2D plot is shown in Figure 22.

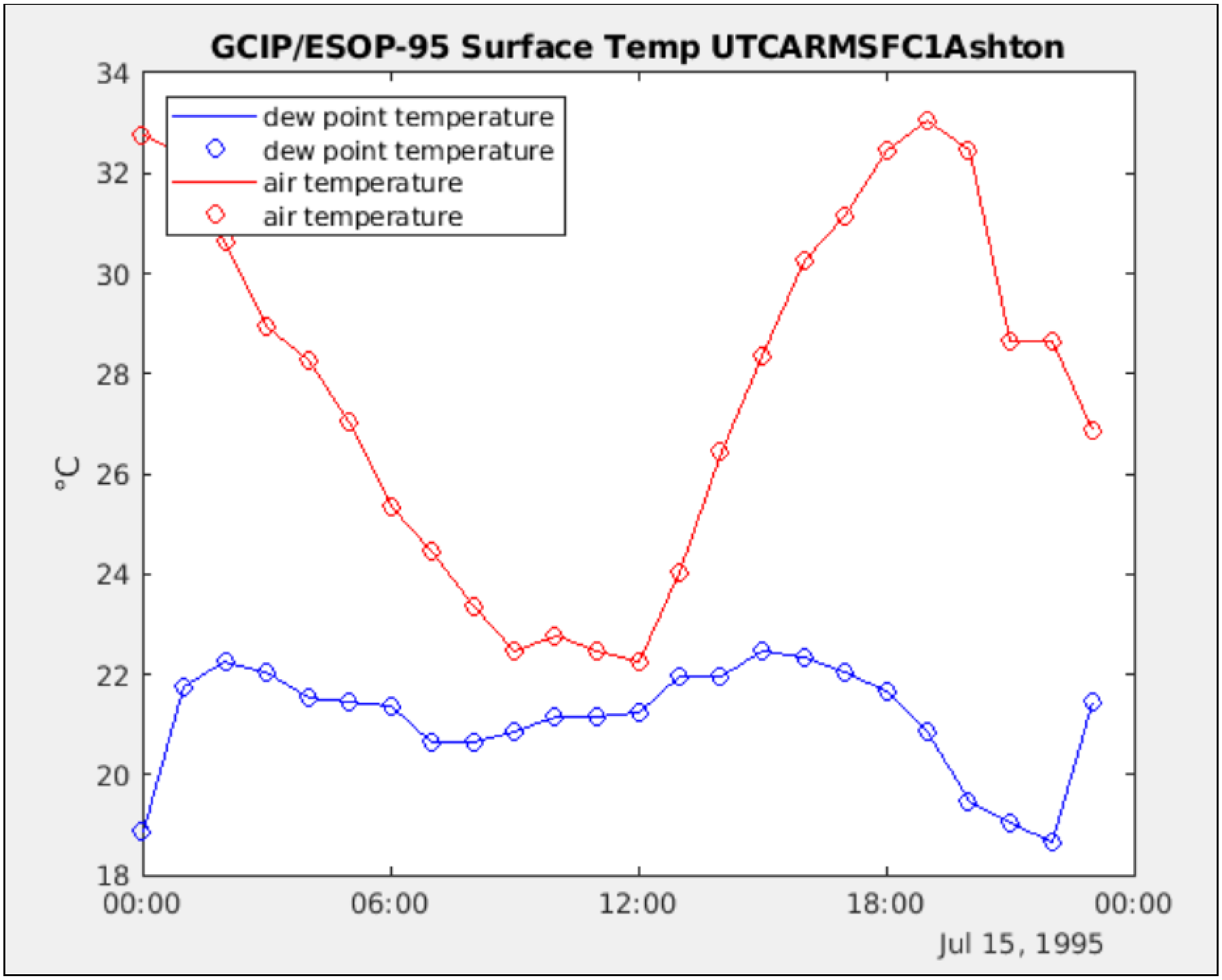


Figure 22: The final result of the 2D “matlab_xy_test.m” script for Temperature and Dew Point for the Ashton site.

Plotting the Data with the 3D Matlab Plot Script

This section shows how the 3D “matlab_xyz_test.m” script differs from the 2D plot script. The 2D and 3D matlab scripts are basically the same through the datetime step described above. **Figures 23-28** show the process for preparing the input data for the 3D plot using the “matlab_xyz_test.m” script. Figures 29 and 30 show the final resulting 3D plot.

```
% Take one time and find all of the indices where the time shows in the
% file. This is used to find all of the information for the given time.

test_time = 1;

t = u_time(test_time);
array_time = strings(cols,1);

for p = 1:cols
    if t == final_time(p)
        array_time(p) = t;
    end
end
```

Figure 23: The indices for one given time are found. The 3D script generates a plot of the temperature field for the entire surface area at a given time.

```
% This statement removes all empty values in the array.

array_time = array_time(~cellfun('isempty',array_time));
% % Create arrays to store lat, long, and dew temp and air temp.

latitude = zeros(length(array_time),1);
longitude = zeros(length(array_time),1);
dew_point_temp = zeros(length(array_time),1);
air_temp_arr = zeros(length(array_time),1);
```

Figure 24: The empty values are removed from the time array. Arrays are created for various variables for later use/plotting. Unlike the 2D script, this script uses latitude and longitude.

```
% % This for loop finds the location info and the dew temp and air temp
% % for the given time.
%
ind = 1;
for p = 1:cols
    if t == final_time(p)
        latitude(ind,1) = lat(p);
        longitude(ind,1) = long(p);
        dew_point_temp(ind,1) = dew_point(p);
        air_temp_arr(ind,1) = air_temp(p);
        ind = ind + 1;
    end
end
```

Figure 25: A loop is used to find all of the location and temperature values for the test time at t=00:00:00 using the index values found above.

```
for p = 1:length(dew_point_temp)
    if dew_point_temp(p) < -100
        dew_point_temp(p) = NaN;
    end
end

for p = 1:length(air_temp_arr)
    if air_temp_arr(p) < -100
        air_temp_arr(p) = NaN;
    end
end
```

Figure 26: The values of temperature may be missing or -999. These missing values are changed to NaN to prevent these values from being plotted.

```
% % Create the 3D grid for the plot.
%
xv = linspace(min(latitude), max(latitude), 100);
yv = linspace(min(longitude), max(longitude), 100);
[X,Y] = meshgrid(xv, yv);
%
% % Create a variable to store the 3D plot.
%
Z = griddata(latitude,longitude,air_temp_arr,X,Y);
```

Figure 27: The grid is created for the 3D plot.

```
% % Plot the data.
%
figure(1)
%
% % The surf command creates the surface plot.
%
surf(X, Y, Z);
hold on
%
% % The stem3 command creates the vertical lines in the plots.
%
stem3(latitude, longitude, air_temp_arr, 'k');
grid on
xlabel('latitude')
ylabel('longitude')
zlabel('°C')
title("GCIP/ESOP-95 Surface Air Temp" + " " + datestr(u_datetime(test_time,1)))
c = colorbar;
c.Label.String = '°C';
```

Figure 28: The 3D plot is created.

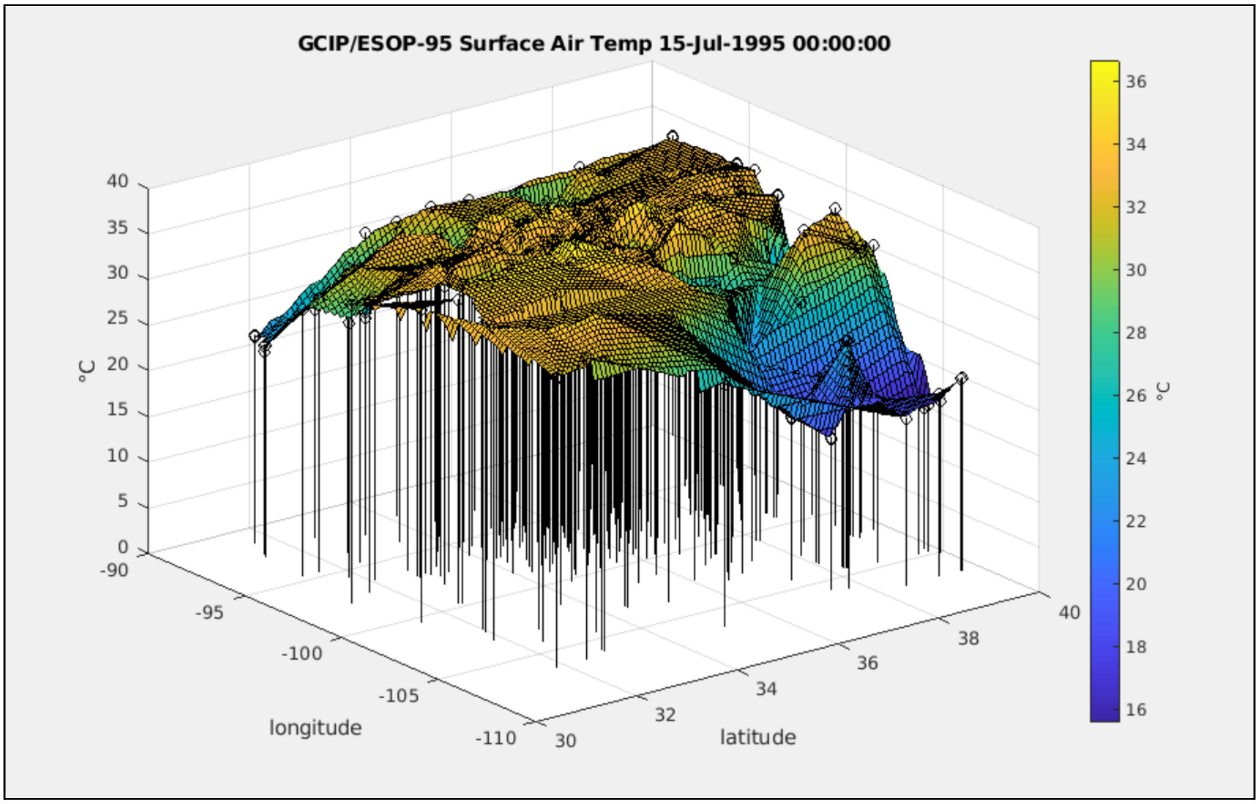


Figure 29: The finished 3D plot for the surface air temperatures for all sites on 15 July 1995 at 00:00:00.

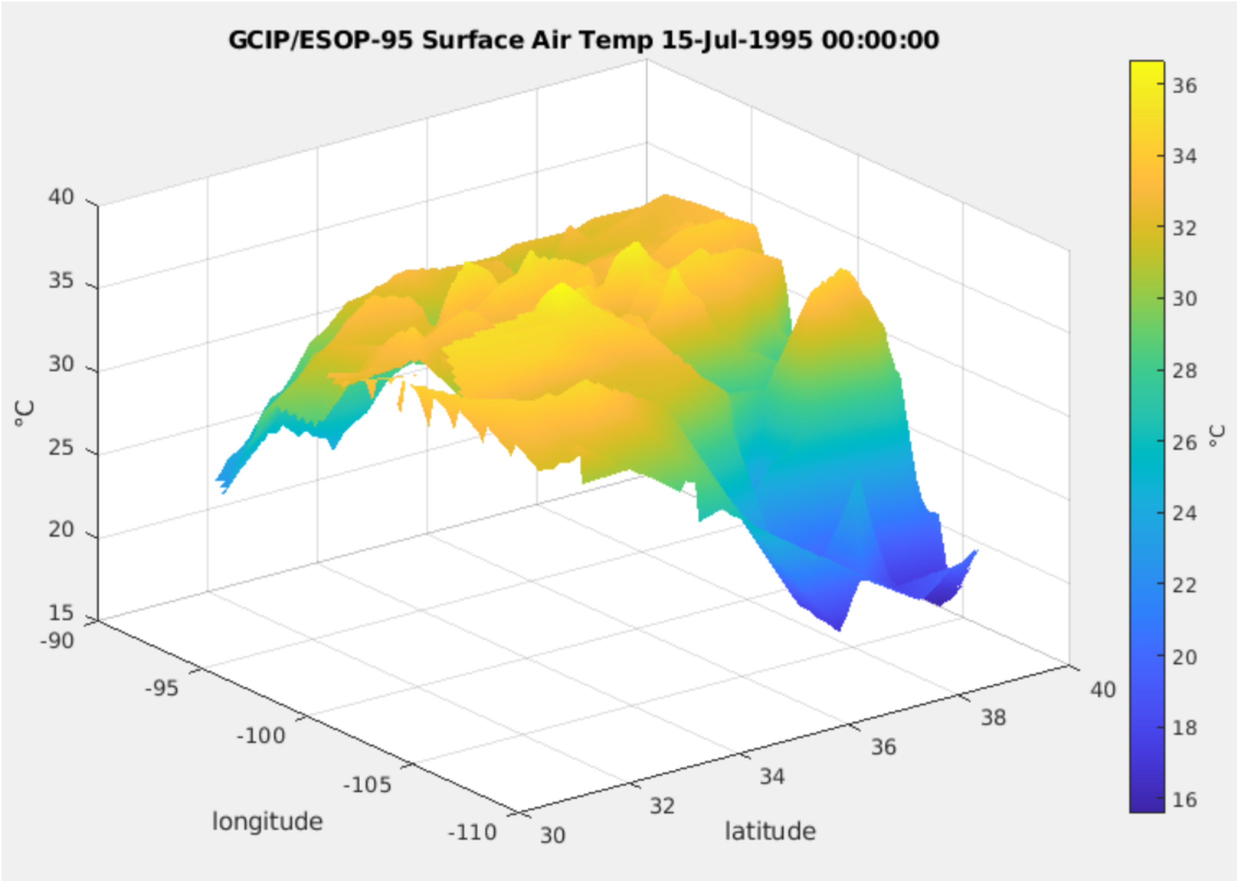


Figure 30: The 3D plot without the grid and site lines.