Hurricane Edouard (Aug. 21 – Sept. 3, 1996)

Information on almost every subject imaginable is available on the web. Often the information of interest is quantitative in nature and, in many cases, data files containing the numerical data of interest are readily available. Weather data, for example, are available on many different web sites. The data come in many different forms -- graphical, text, numerical, and various combinations thereof. Accessing, manipulating, and analyzing data from the web is now a relatively common task, and this application documents a typical case study.

Late in the summer of 1996, hurricane Edouard tracked just off the east coast of the United States and reached the New England area just in time for the Labor Day weekend. The track of the storm is shown in Fig. 1 along with visible and infrared images of the storm when it was just to the east of Cape Cod. I remember this one because, in its wake, it left a mess in my back yard (nothing serious -- just a bunch of small branches and a few somewhat larger ones…). So, I decided to search the web for some data on Edouard and see what I could discover -- since I don't know anything about hurricanes…

In any case, http://weather.unisys.com/hurricane/ had exactly the information I was looking for. I downloaded the pictures shown in Fig. 1 and some detailed numerical data for Edouard that I stored in a file called **edouard96.dat**. A portion of the raw data from **edouard96.dat** is shown in Table 1. This file contains tracking information (latitude and longitude), the wind speed in knots (1 knot = 1.15 mph), and the barometric pressure in millibars (1 bar is 14.5 psi or 100 kPa).

However, the data file has a mixed set of text and numerical data, and this is not easy to read directly into Matlab (or any other programming language). In this case, since the file was relatively small and the number of needed modifications was low, I decided the best course of action was simply to edit the raw data file and convert it into a Matlab script file (I do this a lot…). This was done with the Window's Wordpad editor, as follows:

1. Comment the first three lines by adding a % sign at the beginning of each line.
2. Add a line containing **A = [** before the start of the numerical data and another line containing **];** at the end of the data. Everything between the brackets will be part of the **A** array.
3. Now, recall that every row of a 2-D array in Matlab must have the same number of columns. In addition, since we are interested in some quantitative analysis, we want the **A** array to contain the numerical data for subsequent study. Thus, I performed a series of find and replace and other simple editing operations, as follows:
    a. Replace " TROP" with "; % TROP".
    b. Replace " HURR" with "; % HURR".
    c. Replace "/" with " ".
    d. Replace "Z " with " ".
    e. Edit the 37A, 38A, … , 49A tags in the first column to read 37.5, 38.5, … , 49.5, just so all the data in the **A** array is numerically based. Note that this column is simply a counter of sorts and I did not plan to use it anyway.

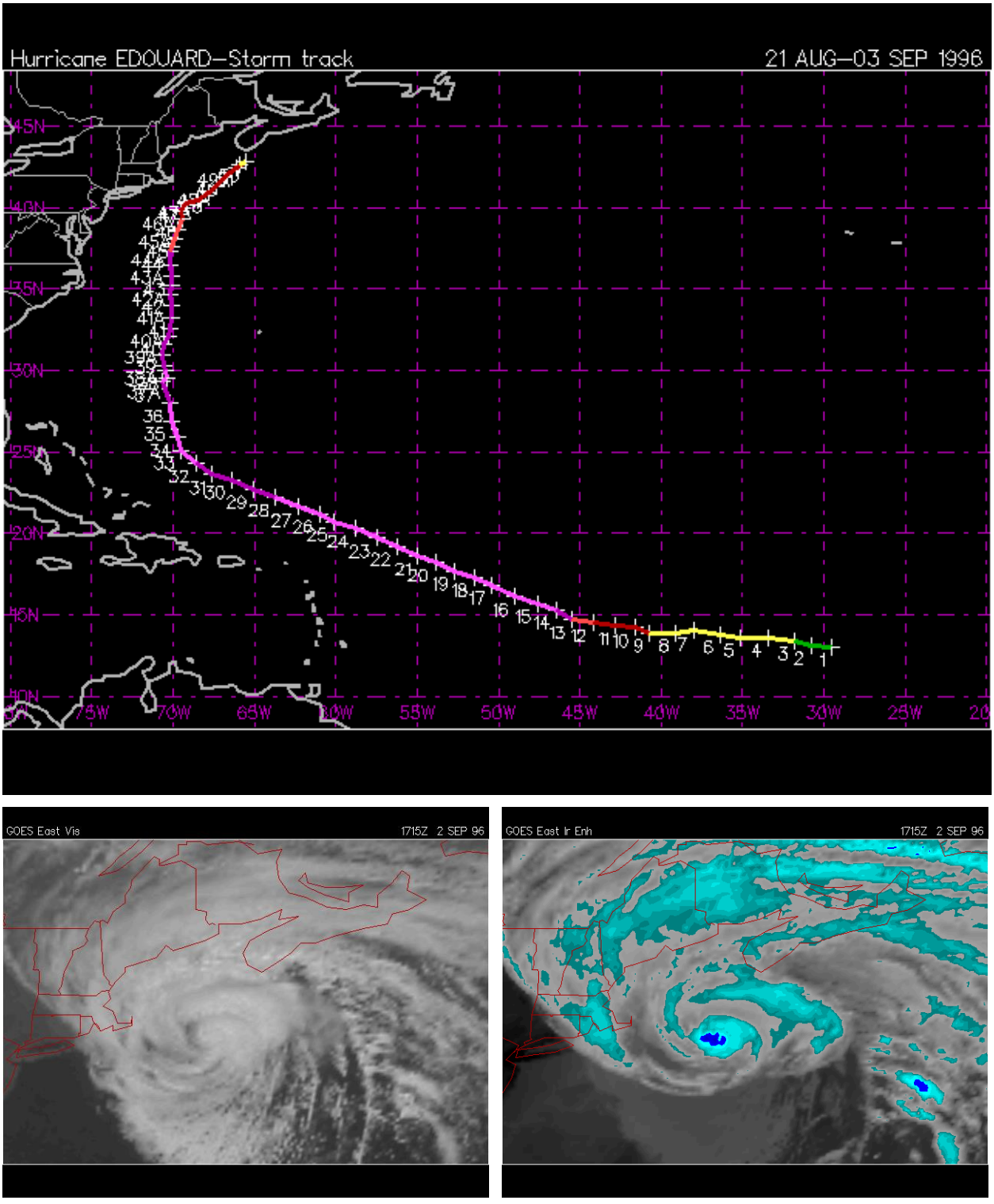

Fig. 1 Edouard's track along with visible and infrared images when east of Cape Cod.

Table 1 Portion of raw data in file edouard96.dat.

```
Date: 21 AUG-03 SEP 1996
Hurricane EDOUARD
ADV  LAT   LON     TIME    WIND  PR  STAT
  1  13.00  -29.50 21/21Z   30  1007 TROPICAL DEPRESSION
  2  13.10  -30.80 22/03Z   30  1007 TROPICAL DEPRESSION
  3  13.40  -31.80 22/09Z   35  1005 TROPICAL STORM
  4  13.60  -33.40 22/15Z   40  1003 TROPICAL STORM
  5  13.60  -35.10 22/21Z   40  1003 TROPICAL STORM
  6  13.80  -36.40 23/03Z   45  1000 TROPICAL STORM
  7  14.00  -38.00 23/09Z   50  1000 TROPICAL STORM
  8  13.90  -39.10 23/15Z   60   989 TROPICAL STORM
  9  13.90  -40.70 23/21Z   65   987 HURRICANE-1
 10  14.20  -41.60 24/03Z   65   987 HURRICANE-1
 11  14.30  -42.80 24/09Z   70   982 HURRICANE-1
 12  14.50  -44.20 24/15Z   90   970 HURRICANE-2
 13  14.70  -45.50 24/21Z  100   960 HURRICANE-3
 14  15.30  -46.40 25/03Z  120   942 HURRICANE-4
 15  15.70  -47.60 25/09Z  125   935 HURRICANE-4

   deleted some data so table fits on one page

 36  28.00  -70.20 30/15Z  105   938 HURRICANE-3
 37  29.30  -70.70 30/21Z  115   934 HURRICANE-4
37A  29.50  -70.40 31/00Z  115   940 HURRICANE-4
 38  30.00  -70.50 31/03Z  110   944 HURRICANE-3
38A  30.30  -70.60 31/06Z  110   950 HURRICANE-3
 39  31.00  -70.70 31/09Z  105   950 HURRICANE-3
39A  31.60  -70.60 31/12Z  105   950 HURRICANE-3
 40  32.10  -70.30 31/15Z  105   954 HURRICANE-3
40A  32.60  -70.20 31/18Z  105   954 HURRICANE-3
 41  33.20  -70.10 31/21Z  100   953 HURRICANE-3
41A  34.00  -70.10 01/00Z  100   957 HURRICANE-3
 42  34.70  -70.20 01/03Z  100   959 HURRICANE-3
42A  35.20  -70.10 01/06Z  100   958 HURRICANE-3
 43  35.80  -70.10 01/09Z  100   958 HURRICANE-3
43A  36.50  -70.20 01/12Z  100   958 HURRICANE-3
 44  37.30  -70.20 01/15Z   95   958 HURRICANE-2
44A  37.60  -70.10 01/18Z   90   959 HURRICANE-2
 45  38.10  -69.90 01/21Z   85   960 HURRICANE-2
45A  38.70  -69.70 02/00Z   85   961 HURRICANE-2
 46  39.30  -69.40 02/03Z   80   964 HURRICANE-1
46A  39.80  -69.50 02/06Z   80   961 HURRICANE-1
 47  40.30  -69.20 02/09Z   70   961 HURRICANE-1
47A  40.50  -68.50 02/12Z   70   962 HURRICANE-1
 48  40.90  -67.80 02/15Z   65   965 HURRICANE-1
48A  41.30  -67.30 02/18Z   65   971 HURRICANE-1
 49  41.90  -66.90 02/21Z   65   976 HURRICANE-1
49A  42.50  -66.10 03/00Z   65   976 HURRICANE-1
 50  42.80  -65.50 03/03Z   60   978 TROPICAL STORM
 51  42.60  -65.90 03/09Z   50   988 TROPICAL STORM
```

Upon completion of the above editorial steps, the file was saved as **ed96dat.m** -- a Matlab m-file. A portion of this file is displayed in Table 2. This m-file simply defines a single 2-D array that contains 7 columns of numerical data. In particular, the last three columns contain the time of day in military time (24 hour clock), the wind speed in knots, and the pressure in mB (millibar). A quick look at these data shows that, in general, as the pressure decreases, the wind speed increases. Thus, I decided to focus on this aspect of the data.

In particular, I decided to write a Matlab file called **ed96.m** to explore the relationship between wind speed and pressure in a hurricane. The resultant program is listed in Table 3 and the three plots produced by **ed96.m** are shown in Figs. 2-4.

Table 2 Portion of Matlab file ed96dat.m (after editing the raw data).

```
%Date: 21 AUG-03 SEP 1996
%Hurricane EDOUARD
%ADV  LAT   LON     TIME    WIND  PR  STAT
A = [
  1  13.00  -29.50 21 21   30  1007 ;   % TROPICAL DEPRESSION
  2  13.10  -30.80 22 03   30  1007 ;   % TROPICAL DEPRESSION
  3  13.40  -31.80 22 09   35  1005 ;   % TROPICAL STORM
  4  13.60  -33.40 22 15   40  1003 ;   % TROPICAL STORM
  5  13.60  -35.10 22 21   40  1003 ;   % TROPICAL STORM
  6  13.80  -36.40 23 03   45  1000 ;   % TROPICAL STORM
  7  14.00  -38.00 23 09   50  1000 ;   % TROPICAL STORM
  8  13.90  -39.10 23 15   60   989 ;   % TROPICAL STORM
  9  13.90  -40.70 23 21   65   987 ;   % HURRICANE-1
 10  14.20  -41.60 24 03   65   987 ;   % HURRICANE-1
 11  14.30  -42.80 24 09   70   982 ;   % HURRICANE-1
 12  14.50  -44.20 24 15   90   970 ;   % HURRICANE-2
 13  14.70  -45.50 24 21  100   960 ;   % HURRICANE-3
 14  15.30  -46.40 25 03  120   942 ;   % HURRICANE-4
 15  15.70  -47.60 25 09  125   935 ;   % HURRICANE-4

   deleted some data so table fits on one page

 30  23.30  -66.40 29 03  110   957 ;   % HURRICANE-3
 31  23.60  -67.60 29 09  110   961 ;   % HURRICANE-3
 32  24.30  -68.60 29 15  115   950 ;   % HURRICANE-4
 33  25.10  -69.50 29 21  120   948 ;   % HURRICANE-4
 34  25.90  -69.70 30 03  120   941 ;   % HURRICANE-4
 35  26.90  -70.10 30 09  120   939 ;   % HURRICANE-4
 36  28.00  -70.20 30 15  105   938 ;   % HURRICANE-3
 37  29.30  -70.70 30 21  115   934 ;   % HURRICANE-4
37.5 29.50  -70.40 31 00  115   940 ;   % HURRICANE-4
 38  30.00  -70.50 31 03  110   944 ;   % HURRICANE-3
38.5 30.30  -70.60 31 06  110   950 ;   % HURRICANE-3
 39  31.00  -70.70 31 09  105   950 ;   % HURRICANE-3
39.5 31.60  -70.60 31 12  105   950 ;   % HURRICANE-3
 40  32.10  -70.30 31 15  105   954 ;   % HURRICANE-3
40.5 32.60  -70.20 31 18  105   954 ;   % HURRICANE-3
 41  33.20  -70.10 31 21  100   953 ;   % HURRICANE-3
41.5 34.00  -70.10 01 00  100   957 ;   % HURRICANE-3
 42  34.70  -70.20 01 03  100   959 ;   % HURRICANE-3
42.5 35.20  -70.10 01 06  100   958 ;   % HURRICANE-3
 43  35.80  -70.10 01 09  100   958 ;   % HURRICANE-3
43.5 36.50  -70.20 01 12  100   958 ;   % HURRICANE-3
 44  37.30  -70.20 01 15   95   958 ;   % HURRICANE-2
44.5 37.60  -70.10 01 18   90   959 ;   % HURRICANE-2
 45  38.10  -69.90 01 21   85   960 ;   % HURRICANE-2
45.5 38.70  -69.70 02 00   85   961 ;   % HURRICANE-2
 46  39.30  -69.40 02 03   80   964 ;   % HURRICANE-1
46.5 39.80  -69.50 02 06   80   961 ;   % HURRICANE-1
 47  40.30  -69.20 02 09   70   961 ;   % HURRICANE-1
47.5 40.50  -68.50 02 12   70   962 ;   % HURRICANE-1
 48  40.90  -67.80 02 15   65   965 ;   % HURRICANE-1
48.5 41.30  -67.30 02 18   65   971 ;   % HURRICANE-1
 49  41.90  -66.90 02 21   65   976 ;   % HURRICANE-1
49.5 42.50  -66.10 03 00   65   976 ;   % HURRICANE-1
 50  42.80  -65.50 03 03   60   978 ;   % TROPICAL STORM
 51  42.60  -65.90 03 09   50   988 ;   % TROPICAL STORM
 ];
```

Table 3 Listing of Matlab file ed96.m.

```
%
%   ED96.M   Plots data for Hurricane Edouard (late Aug 1996) using Matlab
%
%   This is a demo that illustrates some aspects of programming within the
%   Matlab environment.  The goal here is simply to plot some data that were
%   downloaded from the Web related to hurricane Edouard (which started
%   Aug 21 1996).  This hurricane came up just off the east coast and caused
%   some minor damage (high winds, heavy rain, local flooding, etc.) in the
%   local area.
%
%   The Matlab functions for reading mixed text and numerical data from ascii
%   files are not particularly easy to use (especially for the beginner).  Therefore,
%   instead of reading the raw data directly, some editing of the original file
%   was performed.  Thus, there are two files with basically the same data (the
%   second file will be used here).
%     1. raw data file:  edouard96.dat (http://weather.unisys.com/hurricane/)
%     2. m-file containing data in modified format:  ed96dat.m
%   The original data was modified with the Wordpad editor...
%
%   A brief study of the raw data shows a correlation of the pressure and wind
%   speed versus time.  Thus two data representations are presented:
%     1. pressure and wind speed versus time
%          a. two plots on single page -> subplots
%          b. use of double y axes
%     2. wind speed versus pressure
%
%   File prepared by J. R. White, UMass-Lowell (last update: Sept. 2017)


%

      clear all,   close all,   nfig =0;
%
%   First let's execute script file ed96dat.m to get the raw data into Matlab
%   Note: This defines a matrix, A, that has 7 columns (see original and
%         modified data files), of which only the last 3 columns are used here.
%          col 5 - time of day (in military time so we will need to work on this)
%          col 6 - wind speed in knots
%          col 7 - pressure in millibar
%
      ed96dat           % this defines matrix A (a script file can call another script)
      tt = A(:,5);      % relative time (need to work on this)
      ws = A(:,6)*1.15; % wind speed (mph)
      press = A(:,7);   % pressure (mB)
%
%   Create a time vector that represents cumulative time
      NT = length(tt);   t = zeros(size(tt));
      for i = 2:NT
        if tt(i) > tt(i-1);   t(i) = t(i-1)+(tt(i)-tt(i-1));   end
        if tt(i) < tt(i-1);   t(i) = t(i-1)+(tt(i)+(24-tt(i-1)));   end
      end
%
%   Plot wind speed and pressure versus time (use subplots)
      nfig = nfig+1;   figure(nfig)
      subplot(2,1,1),plot(t,press,'go','LineWidth',2), grid
      ylabel('Pressure (millibar)')
      r = axis;  r(3) = 900;  r(4) = 1050;  axis(r)  % resets y-axis limits
      title('Ed96: Data for Hurricane Edouard (starting Aug 21, 1996)')
      subplot(2,1,2),plot(t,ws,'rs','LineWidth',2), grid,
      xlabel('Cumulative Time (hours)'),  ylabel('Wind Speed (mph)')
%
%   Plot wind speed & pressure using two y axes - this is a little advanced
      nfig = nfig+1;   figure(nfig)
      [ax,h1,h2] = plotyy(t,press,t,ws); grid
      xlabel('Cumulative Time (hours)'),
      title('Ed96: Data for Hurricane Edouard (starting Aug 21, 1996)')
      axes(ax(1)), ylabel('Pressure (millibar)'),  set(ax(1),'YColor',[0 0 0])
      set(ax(1),'YLim',[900 1050],'YTick',[900 950 1000 1050], ...
```

```
                'YTickLabel',[900 950 1000 1050]');
      axes(ax(2)), ylabel('Wind Speed (mph)'),     set(ax(2),'YColor',[0 0 0])
      set(ax(2),'YLim',[0 150],'YTick',[0 50 100 150],'YTickLabel',[0 50 100 150]');
      set(h1,'Marker','o','MarkerEdgeColor','g','LineStyle','none','LineWidth',2);
      set(h2,'Marker','s','MarkerEdgeColor','r','LineStyle','none','LineWidth',2);
      legend([h1 h2],'Pressure','Wind Speed')
%
%   Plot wind speed versus pressure and fit a best straight line through the data
      [coeff] = polyfit(press,ws,1);                % does linear fit
      wsfit = polyval(coeff,press);                 % does polynomial evaluation
      nfig = nfig+1;   figure(nfig)
      plot(press,ws,'go',press,wsfit,'b-','LineWidth',2), grid
      r = axis;  r(3) = 25;  r(4) = 150;  axis(r)    % resets y-axis limits
      title('Ed96: Data for Hurricane Edouard (starting Aug 21, 1996)')
      xlabel('Pressure (millibar)'),ylabel('Wind Speed (mph)')
      legend('Data points','Linear fit')
%
%   end of program
```

As apparent, Figs. 2 and 3 show about 300 hours of data on pressure and wind speed versus time (starting at 9 pm on Aug. 21, 1996). The worst part of the storm lasted nearly 150 hours (about 6 days) when the pressure dipped to about 950±15 mB. During this same time, the wind speed reached over 140 mph -- a Category 4 hurricane. Clearly, these data show a strong correlation between wind speed and pressure.

Note: The same information is contained in Figs. 2 and 3, with Fig. 2 displaying pressure vs. time and wind speed vs. time in a standard ***subplot*** format. Figure 3, on the other hand, was generated with the ***plotyy*** command, which allows two y-axes. This is a nice feature, but it is also a little more difficult to work with (see below).

To explore the speed-pressure relationship a little further, a plot of wind speed versus pressure was generated and it is displayed as Fig. 4. In addition to the raw data (the green circles), a linear fit was made to the data. Although there is a fair amount of scatter, Fig. 4 clearly shows a strong inverse relationship between wind speed and pressure that, to a first approximation, can be estimated by a linear fit (we will discuss the details of performing curve fits in a later lesson).

Well, although the hurricane data shown here are quite interesting, the real point of this example was to illustrate a number of aspects about writing programs with Matlab. As such, we now will elaborate a little on several parts of **ed96.m**, as follows (please refer to Table 3, as needed):

1. The first section of code executes Matlab file **ed96dat.m** and extracts the data of interest from the **A** array. Note that **ed96dat.m** is a Matlab script file that is executed from within another script file, **ed96.m**. Recall that any variables defined in a script file are stored in the Matlab workspace and are available for use as needed. The extraction of columns 5, 6, and 7 into the **tt**, **ws**, and **press** arrays was done simply for convenience and program clarity.

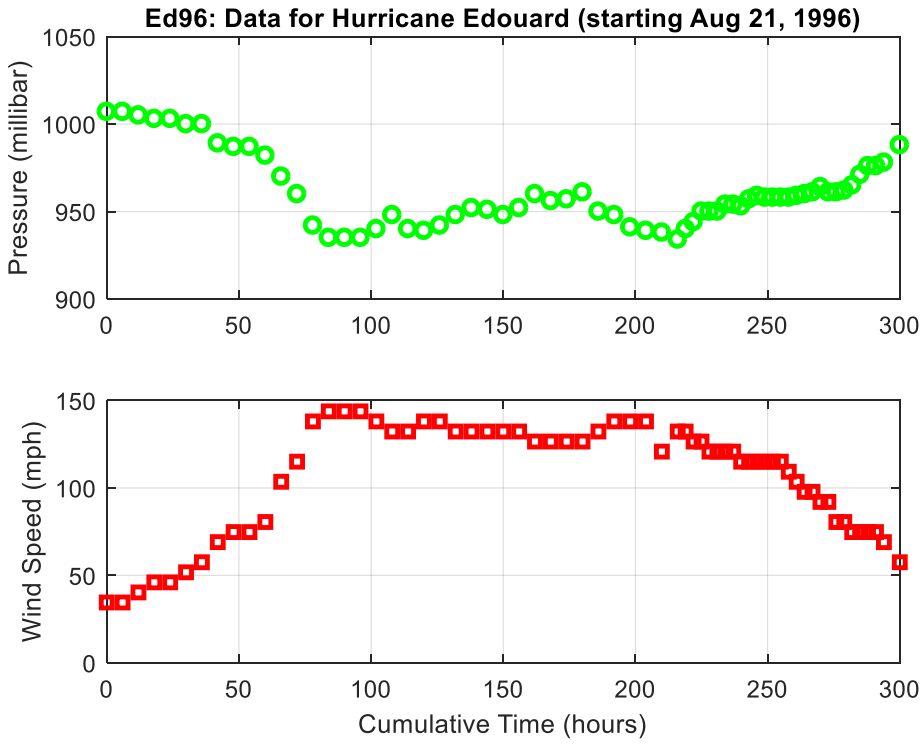

Fig. 2 Pressure and wind speed vs. time for hurricane Edouard (subplot format).

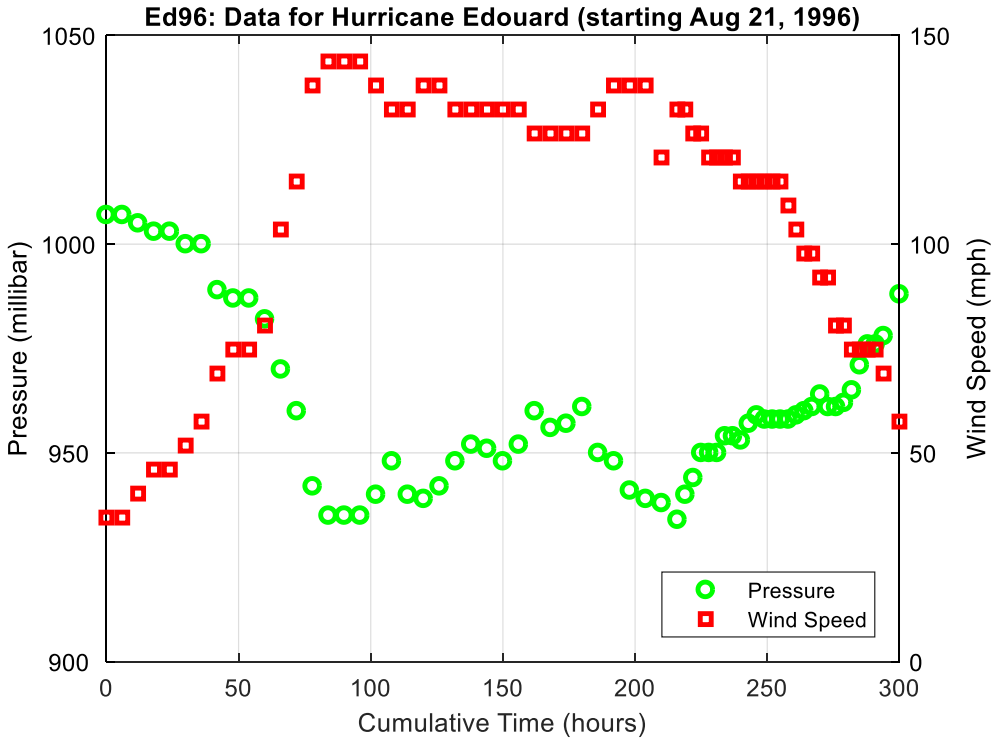

Fig. 3 Pressure and wind speed vs. time for hurricane Edouard (double y-axis format).

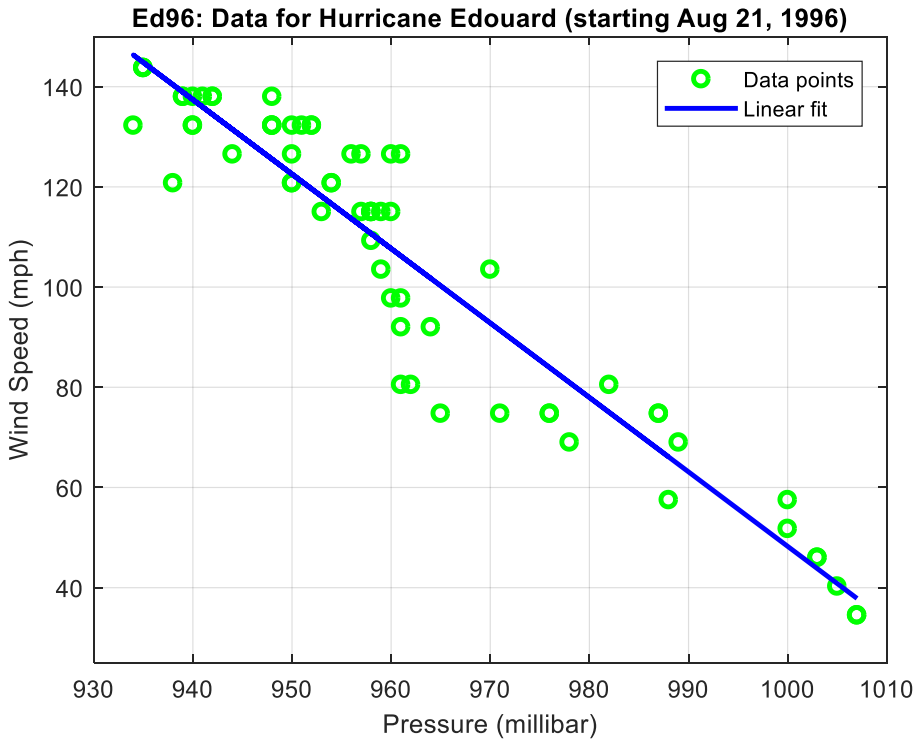

Fig. 4 Relationship between wind speed and pressure for hurricane Edouard.

2. Note that the data file contains time-of-day data based on a 24-hour clock. For plotting purposes, we desired a time vector that represents cumulative time since the initial measurement (at 9 pm on Aug. 21, 1996). This vector is created in a loop that adds the Δt between measurements to the running sum. A test was needed here to determine if a new day had started since the last measurement. This occurs when the measurement time, tt_i, based on a 24-hour clock is less than the previous recorded time, tt_{i-1}. Thus, the logic used to create a cumulative time vector, **t**, is given as follows:

   if $tt_i > tt_{i-1}$, then $t_i = t_{i-1} + tt_i - tt_{i-1}$

   and

   if $tt_i < tt_{i-1}$, then $t_i = t_{i-1} + tt_i + (24 - tt_{i-1})$

   For example, if the measurement at the i-1 point was at 9 pm (21 hours) and the next reading was at 3 am (3 hours), then the second branch is followed, with

   $\Delta t = tt_i + (24 - tt_{i-1}) = 3 + (24 - 21) = 6$ hours

   which is exactly what we wanted!

3. Figure 2 was generated with the use of the ***subplot*** command in Matlab -- nothing new here. However, sometimes it is appropriate to have completely different quantities (wind speed and pressure, for example) displayed on the same plot. This requires capability to define and annotate two different y-axes for the same plot and, in Matlab, this is accomplished with the ***plotyy*** command.

This command, however, is not as straightforward to use as the standard ***plot*** command. Getting the data plotted is easy, but properly labeling the axes is somewhat more challenging. In the command

```
[ax,h1,h2] = plotyy(t,press,t,ws);
```

the output variables, **ax**, h1, and h2, are handles to the two axes (ax(1) for the left axis and ax(2) for the right one) and to the graphics objects created for each curve (h1 for the first curve, pressure vs. time, and h2 for the second curve, wind speed vs. time). Now, with handles to these objects, we can label and/or modify the plot attributes as desired.

For example, to label the right axis, we first make this the active axis with the ***axes(ax(2))*** command, and then use the ***ylabel*** command as usual. Similarly, we can set the attributes of the wind speed vs. time curve with the ***set*** command by identifying the handle of the graphic object of interest (here h2 points to the desired curve) and then specifying the desired attributes. Thus, we see that the user has complete control -- although using Matlab's Handle Graphics can be a little tricky…

Note: Handle Graphics is the name associated with a series of low-level graphics routines within Matlab that does most of the work when generating graphics in Matlab. These functions are usually hidden inside the higher-level functions (***figure***, ***plot***, ***title***, ***legend***, etc.) that are utilized to handle most of our graphics processing needs. However, in some cases, we may need access to the more primitive commands that can provide a higher degree of control. We will not spend much time in this course discussing Matlab's general Handle Graphics capability because it is somewhat specialized and a little confusing for the beginning Matlab user. We will see its use on occasion, where needed, so you will at least get a brief glimpse of some of its capabilities at various times over the course of the semester.

You should be aware, however, that you can access many of the Handle Graphics features via the Figure Window toolbar and menus. This interactive capability is great for fine tuning a few plots for publication and special presentation, etc.. However, it is somewhat cumbersome for routine plot modification on every Matlab plot. Thus, if you have a need to specialize a series of plots, Matlab's Handle Graphics is the tool to use. Should you require this capability, I suggest that you consult Matlab's ***help*** facility or one of the many good Matlab reference books that are available.

---

4. The last part of **ed96.m** creates Fig. 4. The only new aspect here is the use of the ***polyfit*** command to do the linear curve fit, and the use of ***polyval*** to evaluate the linear polynomial at several values of pressure for plotting purposes. These functions are extremely useful and we will have several occasions where they will be used over the course of the semester. In particular, a full lesson will be devoted to the subject of curve fitting techniques (Lessons #7) where we will see the ***polyfit*** command again, and a section of Lesson #5 will highlight several different polynomial operations in Matlab. However, if you can't wait until then, you are encouraged to explore the ***polyfit***, ***polyval***, etc. functions using Matlab's ***help*** facility on your own. We will explore these functions further within these notes in a later lesson…

This completes the Hurricane Edouard example. The goal here was to introduce some additional programming features available within Matlab. In particular, we saw how some simple editing could convert a mixed data file into a Matlab numerical array -- that could then be further manipulated and analyzed. A looping structure with embedded conditional tests was used to modify existing information (time-of-day data) into something more useful for our present needs (cumulative time vector). A brief introduction to Matlab's Handle Graphics was also given to allow the use of the ***plotyy*** command for creating plots with two y-axes. Finally, a brief glimpse of some functions for working with polynomials (***polyfit*** and ***polyval***) was also given. These kinds of tasks occur frequently in many analysis situations, so this demo will hopefully serve as a useful guide for future reference.

Each demo within these notes, if studied carefully, should add something to your growing inventory of ways to use Matlab for solving real problems. Hopefully Hurricane Edouard has added some new tools to your toolbox…