Scripts for AER-WRT Ensemble Forecasts

Kerry Emanuel

March, 2011

Initial Preparations

First download recursively all the files/directories at ftp://texmex.mit.edu/pub/emanuel/AER/ . It is important to preserve the directory structure. Next compile the FORTRAN 95 programs *wrt_aer.f* and *read_all.F*, preferably using a high-end compiler such as Portland group with careful attention to optimization flags for *wrt_aer.f*. The executables should have the same names, i.e. *wrt_aer* and *read_all*. Also make sure that the linux scripts *masterec*, *readfiles*, *readlist*, and *runlist* have execute permissions.

In MATLAB, add the subdirectory *matlab_scripts* and all of its subdirectories to your matlab path.

Overview of scripts

There are 3 classes of scripts. First, there is a set of linux scripts for downloading the data as prepared by Susanna and reformatting it for direct use by the program. Next there is a linux script that actually runs the model, creates MATLAB output, and sorts into appropriate directories. Finally, there is a set of MATLAB scripts for plotting and analyzing the output.

ECMWF data reading and formatting scripts

readfiles: Usage: readfiles <datetime storm # of ensemble members # of 12-hour periods>

where *datetime* is the 4-digit year, 2-digit month, two-digit day, and 2-digit GMT time of the forecast, *storm* is the two-character UPPER CASE basin identifier (AL, EP, WP, IO, OR SH) and2-difit storm number, *# of ensemble members* is the total number of ECMWF ensemble members (usually 51), and *# of 12-hour periods* is the total number of 12-hour periods in the forecast (usually 21).

This script reads the appropriate files from Susanna's directory, formats them, and places them in appropriate subdirectories. You may want to alter this to point to different input directories. Example: readfiles 2010091500 AL11 51 21

readlist: A simple script that runs *readfiles* for a list of storms/times.

Scripts for running the model

masterec: Usage: masterec <basin storm # year month GMT>

where *basin* is the two-character UPPER CASE basin identifier (AL, EP, WP, IO, OR SH), *storm #* is the two-digit storm number, *year* is the 4-digit year, *month* is the two-digit month, and *GMT* is the 2-digit GMT time.

Note that you can specify the number of tracks to generate and the critical wind speed which the storm must exceed at some point in its lifetime to be included. You may also specify the number of days to skip until a storm forms in the ECMWF ensemble; this remains untested though. It also goes to the NHC web site (for AL and EP storms) and downloads the appropriate a- and b-decks. WARNING: These files will not be available in the usual directory after the season! Also, you will have to make arrangements for AER to receive similar decks for the WP, IO and SH basins. This script then runs the model itself, a MATLAB script that generates matlab binary output, and directs th output to various subdirectories.

This script has a great deal of vestigial material in it. This will not slow anything down to any perceptible degree, but may make it harder to read the script.

Example: masterec AL 09 2008 09 10 00

runlist: A simple script for running multiple storms/times.

Analysis and plotting scripts

We provide several MATLAB scripts that may prove useful in displaying and analyzing the ensemble. To use these, **you first need to run the matlab script *prep.m*.** (Note: You need to have zip and unzip utilities. In case you do not have them I include some Windows 7 utilities, but you should delete them if your system already has one or you are not running Windows 7. There are several scripts, including *prep.m*, in which you may have to change the name of the zip/unzip utilities that are called.) You will be asked to input the date and time, and the storm number. This runs virtually instantaneously and creates a bunch of matlab binaries in the main directory which can subsequently be analyzed using the other matlab scripts without having to enter the storm information each time.

After you run *prep*, you should be ready to display and analyze the output using the following scripts:

1) *alltrmap.m*: This plots multiple storm tracks on a map background provided by an image file; the image must be a map in equirectangular projection. (One is provided.) At the beginning of the script, once can choose such items as the track color. The tracks can be specified in the order they were produced in, ordered by their lifetime maximum wind speed, or ordered by the maximum wind speed at a point of interest or at the intersection points of a specified set of line segments.

2) *alltrplot.m*: Same *as alltrmap*, but the map background is supplied using *m_map* routines.

3) *ecmap.m*: Overlays the ECMWF track ensemble on the tracks produced by *alltrmap*, which must be run first.

4) *ecplot.m*: Same as *ecmap*, but for the conventional map background. *Alltrplot* must be run first.

5) *bdeckmap.m*: Overlays the latest "B-deck" track from NHC on tracks produced by *alltrmap.*

6) *bdeckplot.m*: Overlays the latest "B-deck" track from NHC on tracks produced by *alltrplot*.

7) *adeckmap.m*: Overlays latest "A-deck" forecast track from NHC on tracks produced by *alltrmap*.

8) *adeckplot.m*: Overlays latest "A-deck" forecast track from NHC on tracks produced by *alltrplot*.

9) *intensity.m*: Shows the evolution of the maximum 10-meter wind speed with time for each ensemble member.

10) *bdeckintensity.m*: Overlays the B-deck intensity on the ensemble intensity produced by *intensity*, which must be run first.

11) *adeckintensity.m*: Overlays the A-deck intensity on the ensemble intensity produced by *intensity*, which must be run first.

12) *errorcalc.m:* This script calculates certain rudimentary error statistics from a collection of forecasts, specified at the beginning of the script, and compares them to ECMWF track forecasts and the official forecast. It looks for the "a decks" and "b decks", files called "axxnnnnnn.dat' and 'bxxnnnnnn.dat', in a subdirectory called "decks"; these should be the final forecast and best track files from NHC or JTWC. (Here xx is the basin abbreviation and nnnnnnn is the storm number and year, e.g. bal072010.dat.) Note that the comparisons are homogeneous only in the sense that at least one ensemble member is present from the WRT-AER and from the ECMWF sets at each forecast at the initial time; if there are no ensemble members present for one product at some lead time the error stats are nevertheless calculated for the other product at that lead time. Also, only a single ensemble member need be present for an ensemble mean to be calculated. Finally, the deviations plotted here are not standard; rather, they are the averages of the absolute values of the differences between the ensemble members and the ensemble mean, or the difference between the ensemble mean an thee best track. (This could be easily changed.)

13) *errorplot.m*: Makes plots from results of *errorcalc*.

14) *trmap.m*: Plots individual storm tracks, color coded by intensity and with dates/times.

15) *trplot.m*: Same as *trmap*, but using conventional map background.

16) *trackdensity.m*: Displays the number of tracks per unit area over the duration of the forecast.

17) *pointwind.m:* Plots time series of wind speed and direction at a given point of interest, specified by maximum wind over the lifetime of the storm, or by track number.

18) *swath.m*: Shows a map contouring the peak wind experienced at each point in space during the lifetime of a single event. This script may take a little time to run.

19) *pointpdf.m*: Displays an ensemble derived probability density distribution of peak wind at a given point in space.

20) *skillcalc.m*: Calculates a set of ensemble statistics for later plotting and analysis. This script does not itself display any plots, but stores results in a matlab binary called *fields.mat*. Note that this may take quite awhile to run. Also note that, at present, the file *fields.mat* will be overwritten by subsequent calls to *skillcalc*, so it would be prudent to rename it and store it.

21) *skillplot.m*: Plots various statistics generated by *skillcalc*, which must be run first.

22) *windfield.m*: Contours the wind speed associated with a specified event at a specified time. The radial wind profile is calculated in *windprofile.m*. One may also plot the track of the event up to the specified time.

Background utilities

prep.m: Reads in the main track file and calculates the full wind speed, including the effect of storm translation. It also calculates the lifetime maximum wind speed of each event, the maximum wind speed at a point of interest, if the tracks have been filtered to pass within a certain distance of that point, or the maximum wind speed in the storm as it passes over any of the line segments specified in 'poly.in', of the tracks have been filtered to pass through such line segments. The "point of interest" is read from the master binary file, but can be specified independently if desired.

eccov.m: Used to calculate translation velocity means and covariances for use in the main program.

Function [ut,vt,jmax] = utrans (lat,long): Given 2-hour latitude and longitude positions along a track, this returns the west-eat and south-north components of the storm translation velocity. The function assumes that the lats and longs are two-D arrays whose first index is the event number and whose second index is the 2-hour position for the event. It also returns *jmax(n)*, where *n* is the number of events. This marks the last 2-hour position of the track; the translation speeds are padded with zeros from the end of each track to the end of the file. Some smoothing is done to the translation speeds; the mount of smoothing is adjustable using the parameter *smfac*.

Function [ut,vt,jmax] = utransbest (latstore,longstore,monthstore,daystore,hourstore): Similar to utrans, but requires month, day and hour information. This is ideal for historical tracks whose data points may be unevenly spaced in time.

Function [nint,xint,yint,jint,kint] = boxm (long,lat,xa,ya,xb,yb): Given 2-D latitude and longitude positions along a track (as in *utrans*), and *j-1* line segments whose end points are given by *xa(j), ya(j), xb(j),* and *yb(j)*, this function returns the intersection longitude and latitude (*xint(n)* and *yint(n)*) of each track with the set of segments. Nint is 0 or 1 denoting no intersection or an

intersection of the track with the set of line segments. Note that in the event that a track intersects the array of line segments more than once, only the first intersection point is calculated. The last point of the set *xa(j), ya(j), xb(j),* and *yb(j),* may equal the first, in which case one has a closed polygon. In this case, the routine also checks for tracks that lie entirely within the polygon. For this reason, the segments must be specified going CLOCKWISE around the polygon. The quantities *xint* and *yint* constitute the first intersection point, along the line segment numbered *jint* and the *kint'th* point along the track. In the case of a closed polygon, if the track is entirely within the polygon, *jint* is set equal to *-1* and *xint=yint=0.0.*

Function [vs, dir, dayk] = pointseries (latstore, longstore, vstore, rmstore, monthstore, daystore, hourstore, ut, vt, jmax, plat, plong): This function take the n X m matrices for the latitudes, longitudes, radii of maximum winds, dates, times, and translation speeds and, using the latitude (*plat*) and longitude (*plong*) of a point of interest, calculates the time series of wind speed and direction at the point of interest. A wind profile, calculated in *windprofile.m*, is fitted to the maximum wind speeds and radii of maximum winds to calculate the wind speed and direction at the point of interest, and the translation speed contribution to the net wind is accounted for. The routine *utrans.m* should be run before this one. Note that when this routine is called by *prep*, the point of interest is taken to be the center of the circle, if circular filtering has been used to generate the track set.

Function [vs, dir] = pointshort(latstore, longstore, vstore, rmstore, ut, vt, jmax, plat, plong): Same as function *pointseries* above, but omits calculation of date/time and is thus a little faster.

Function [V] = windprofile(r, rm, r0, vm): Given a radius vector *r*, a radius of maximum winds *rm*, a radius of vanishing wind *r0* (marking the outer boundary of the storm), and a maximum wind speed *vm*, this function returns the azimuthal velocity *V* as a function of radius *r*. Several choices of standard wind profiles are presented at the beginning of the script. Note that *r0* is not used by all the profiles.

Raw Matlab binary files *(in alphabetical order)*

Note that for the ECMWF tracks, not all of these files or parameters are used.

[**Convention:** In the descriptions below, the index *n* refers to the event number, while the index *m* refers to the 2-hour records of each event. For example, *latstore (n,m)* is an array containing *n* tracks each of which has *m* two-hour observations. (The arrays are padded with zeros between the end of each track and the end of the file.)]

bas (character scalar): The ocean basin for this track set.

city_radius (scalar): Used for circular filtering, this is the distance, in kilometers, from a specified point of interest that tracks must pass within to be included in this set. Not used in line-segment or polygon filtering.

clat (scalar): The latitude of the point of interest, used in circular filtering.

clong (scalar): The longitude of the point of interest, used in circular filtering.

daystore (m x n array): The day of the month of 2-hour points along each track.

factor (scalar): (not currently used).

freq (scalar): The annual frequency of all the events in the event set. For all subsets of size *x*, the annual frequency of the subset is just *x/n*.

gmeth (character): The genesis method used for this track set. *clim* denotes genesis by random draws from a best-track-based genesis climatology, while *rand* denotes random seeding and natural selection.

hourstore (m x n array): The Greenwich Mean Time of 2-hour points along each track.

latstore (m x n array): The latitude of 2-hour points along each track.

longstore (m x n array): The longitude of 2-hour points along each track.

monthstore (m x n array): The calendar month of 2-hour points along each track.

pstore (m x n array): The central surface pressure (hPa) of 2-hour points along each track.

rmstore (m x n array): The radius (km) of maximum circular wind of 2-hour points along each track.

shape (character): The filtering type used for this track set; *circ* denotes circular filtering, while *poly* denotes series of line segments or closed polygon.

shearstore (m x n array): The magnitude of the 850-250 hPa environmental wind shear at each 2-hour point along each track.

vpstore (m x n array): The potential intensity (knots) at each 2-hour point along each track.

vstore (m x n array): The maximum circular wind speed at each 2-hour point along each track. Note that a fraction of the translation speed is generally added to this to define the actual maximum wind speed, interpreted as a one-minute average at 10 m altitude.