

Appendix for 3rd Trimester 2008 Tracking Report D. Clark

Reproduced below are the SQL queries, Matlab m-code, and Python application code used to generate and analyze the data referenced in the MMTO Internal Technical Memorandum #2009-02. This information is presented for the purpose of making it available to those who might wish to reproduce this work (or improve upon it).

MySQL Query

The original data query for the tracking data was:

```
SELECT * FROM `tracking_performance_log` WHERE `type` =  
'ELEVATION' AND `timestamp` < '2008-12-31' AND `timestamp`  
> '2008-09-01' AND `length` > '119.0'
```

This query extracted only elevation entries for the 3rd trimester months and excluded truncated files that weren't the standard file length of 120s.

Python Application

Tom Trebisky provided a Python application that used the timestamps produced within the MySQL query output population to extract from the MMTO MySQL database the relevant wind sensor values so that wind influence on tracking could be quantified. That code is reproduced below:

```
-----BEGIN-----  
-----  
#!/usr/bin/python  
  
import MySQLdb  
import datetime  
import sys  
import math  
  
# Vaisala returns wind speeds in meters per second.  
# multiply by this to get miles per hour  
# (The Vaisala give wind direction in degrees from north,  
# just like telescope azimuth (imagine that!))  
  
ms2mh = 2.237  
d2r = math.pi / 180.0  
r2d = 180.0 / math.pi  
  
def csvout ( rec ) :  
    first = 1  
    for item in rec:  
        if first:  
            rv = str(item)  
        else:  
            rv += "," + str(item)
```

```

        first = 0
    return rv

def average_wind ( records, type ) :
    global ms2mh
    global d2r, r2d

    #   print "awmax sp, dir =", record[15], record[14]
    #   return ( 0, 0, 0, 0, 0 )

    if type == 3 :
        ixws = 15
        ixwd = 14
    else :
        ixws = 14
        ixwd = 13

    record = records[0]
    count = len(records)
    #print "Processing", count, "records"

    first = 1
    num = 0
    for i in range(count) :
        record = records[i]
        if record[3] == "NA" :
            continue
        if record[3] == "" :
            continue

        num += 1
        #print "record", i, "max, maxd =", record[ixws],
record[ixwd]
        #print "record", i, "avg, avgd =", record[4], record[3]

        if first :
            first = 0
            xsum = 0.0
            ysum = 0.0
            aws = 0.0
            xws = record[ixws]
            xw_index = i

            aws += record[4]
            dir = float(record[3]) * d2r
            xsum += math.sin ( dir )
            ysum += math.cos ( dir )
            if record[ixws] > xws :
                xws = record[ixws]
            xw_index = i

    if num < 1 :
        return ( 0, 0, 0, 0, 0 )

    aws /= num
    xwd = records[xw_index][ixwd]
    xsum /= num

```

```

ysum /= num
awd = math.atan2 ( xsum, ysum ) * r2d
if awd < 0.0 :
    awd += 360.0

#   print "final max =", xws
#   print "final max dir =", xwd

#   print "aws =", aws
#   print "awd =", awd
#   print "count, num =", count, num

return ( num, awd, aws * ms2mh, xwd, xws * ms2mh )

def get_wind ( sname, ts, type ) :
    global db
    global ms2mh

    tse = ts + datetime.timedelta(seconds=120);
    logname = sname + "_background_log"

    sql = "SELECT * FROM %s" % logname
    #sql += " WHERE timestamp > %s LIMIT 1"
    sql += " WHERE timestamp > '%s'" % ts
    sql += " AND timestamp < '%s'" % tse
    #print sql

    cursor = db.cursor()
    cursor.execute ( sql )

    nw = cursor.rowcount

    if nw == 0 :
#       print "query on %s fails" % sname
        return (nw,0,0,0,0)

    #record = cursor.fetchone()
    #print "ts = ", record[0]
    #print "lanstat = ", record[13]
    #sys.exit()

    if nw == 1 :
        record = cursor.fetchone()
        # average wind direction
        awd = record[3]
        # average wind speed
        aws = record[4]
        if type == 3 :
            # maximum wind direction
            xwd = record[14]
            # maximum wind speed
            xws = record[15]
        else :
            # maximum wind direction
            xwd = record[13]
            # maximum wind speed

```

```

        xws = record[14]
#         print "max dir,speed = ", xws, xwd
        return ( 1, awd, aws * ms2mh, xwd, xws * ms2mh )
    else:
        rv = average_wind ( cursor.fetchall(), type )
        #if rv[0] > 10 :
        #     sys.exit()
        return rv

def get_wind3 ( ts ) :
    return get_wind ( "vaisala3", ts, 3 )

def get_wind4 ( ts ) :
    return get_wind ( "vaisala4", ts, 4 )

db = MySQLdb.connect ( host = "localhost", db = "mmtlogs", user =
"mmtstaff", passwd = "multiple" )
cursor = db.cursor()

what = "Topaz"
# note that the execute method adds single quotes around the %s
thingie
cursor.execute ( "SELECT * FROM tracking_performance_log WHERE
type = 'ELEVATION' AND length > '119' AND timestamp > '2008-09-
01' AND timestamp < '2008-12-31' ORDER BY timestamp" )

print cursor.rowcount, " matching records found"

result = cursor.fetchall()

# print "Record has %s items" % len(rec)

for record in result:
    # Vaisala3 is the east unit.  Its data is only good when the
wind is from the east.
    w3 = get_wind3 ( record[0] )
    # Vaisala4 is the west unit.  Its data is good except for
easterly winds.
    w4 = get_wind4 ( record[0] )
    print csvout ( record + w3 + w4 )

# THE END
-----END-----
-----

```

Matlab M-file Code

The data from the MySQL queries were combined into a single comma-separated value (csv) file and imported into Matlab R2008b for analysis. The m-file below is the code used to calculate all the reported statistics and figures for the Tech Memo; the data this code depends on is archived in the network folder that contains this same appendix. The would-be user is cautioned that this m-code is written in cell mode with no particular regard for what variables might be in the workspace, or other error checking. If you don't know what cell mode is or

know what to do about matlab workspace variables, you will need to learn about it!

-----BEGIN-----

-

```
%An m-file for evaluation of the 3T 2008 tracking data
%Source file: El_tracking_3T2008.csv
%work file with wind data: el_tracking_wind_3T2008b.mat
%% Source Data Preliminaries
pk_pk_error = data(:,4);
pkbins = 0:0.1:round(max(pk_pk_error));
[pki, pkr] = hist(pk_pk_error,pkbins);
npkpk = length(pk_pk_error);
pctpkpk = (pki / npkpk) * 100;
figure(); subplot(3,1,1:2);
bar(pkr,pctpkpk, 'EdgeColor', 'k', 'FaceColor', 'm');
set(gca, 'XLim',[0 5]);
xlabel('Peak to peak error, arcseconds');
ylabel('Percent of Data Population');
title(['3rd Trimester 2008 Tracking Data with ' ...
       sprintf('%d',length(pk_pk_error)) ' entries']);
legend(['Median: ' sprintf('%1.3f',median(pk_pk_error)) '']);
grid on;
set(gca, 'Layer','Top'); set(gca, 'XLim',[0 3]);
subplot(3,1,3);
boxplot(pk_pk_error, 'orientation', 'horizontal', 'widths', 0.25);
set(gca, 'XLim',[0, 1.7]); set(gca, 'YLim',[0.8, 1.2]);
xlabel('Peak-to-peak Error Distribution Box Plot');
iqrs = ['IQR: ',sprintf('%1.3f',iqr(pk_pk_error)),''];
annotation(gcf, 'textbox', [0.7565 0.2739 0.07266 0.03043], ...
          'String', {iqrs}, ...
          'FitBoxToText', 'on', 'Color', [0,0,1]);
grid on;

%% Histogram of error
%rms_err = textdata(:,3);
%rms_error = cells2nums(rms_err);
errbins = 0.01:0.01:round(max(rms_error));
[ri,rr] = hist(rms_error,errbins);
nerr = length(rms_error);
%find the winning bin
medianError = median(rms_error);
pcterr = (ri / nerr) * 100;
figure(); subplot(3,1,1:2);
bar(rr,pcterr, 'EdgeColor', 'k', 'FaceColor', 'm');
set(gca, 'XLim',[0, 0.5]);
grid on;
title(['3rd Trimester 2008 Tracking Data with ' ...
       sprintf('%d',length(pk_pk_error)) ' entries']);
set(gca, 'Layer','Top');
xlabel 'RMS Error Distribution, arcseconds';
ylabel 'Percent of Data Population';
grid on;
%err_devs = [sprintf('%1.3f',iqr(rms_error)), ''];
```

```

err_devs = [sprintf('%1.3f',medianError), ''];
legend(['Median: ',err_devs]);
subplot(3,1,3);
boxplot(rms_error,'orientation','horizontal','widths',0.25);
set(gca,'XLim',[0, 0.2]); set(gca,'YLim',[0.8, 1.2]);
xlabel('RMS Error Distribution Box Plot');
iqrs = ['IQR: ',sprintf('%1.3f',iqr(rms_error)), ''];
annotation(gcf,'textbox',[0.7565 0.2739 0.07266 0.03043],...
    'String',{iqrs},...
    'FitBoxToText','on','Color',[0,0,1]);
grid on;
%% Exclude outliers in the peak-to-peak error
%%need wind, az data for later in this m-file
smallerr = []; windspeedsmall = []; winddirsmall = [];
az_angle3 = []; el_angle3 = [];
wind_angle0 = abs(az_angle2 - winddir');
pks_small = find(pk_pk_error < 1.7); %boxplot has outlier at
1.67"
for k = 1:length(pks_small)
    smallerr(k) = rms_error(pks_small(k));
    windspeedsmall(k) = windspeed(pks_small(k));
    winddirsmall(k) = wind_angle0(pks_small(k));
    az_angle3(k) = az_angle(pks_small(k));
    el_angle3(k) = el_angle(pks_small(k));
end
%%
%%repeat the histogram of the error distribution
errbins = 0.01:0.01:max(smallerr);
[ri,rr] = hist(smallerr,errbins);
nerr = length(smallerr);
%%find the winning bin
medianError = median(smallerr);
pcterr = (ri / nerr) * 100;
err_meds = [sprintf('%1.3f',medianError), ''];
figure(); subplot(3,1,1:2);
bar(rr,pcterr,'EdgeColor','k','FaceColor','m');
grid on;
title(['3rd Trimester 2008 Tracking Data with ' ...
    sprintf('%d',length(smallerr)) ' entries']);
set(gca, 'Layer','Top');
set(gca,'XLim',[0, 0.3]);
xlabel 'RMS Error Distribution, arcseconds';
ylabel 'Percent of Data Population';
grid on;
%err_devs = [sprintf('%1.3f',iqr(rms_error)), ''];
err_devs = [sprintf('%1.3f',medianError), ''];
legend(['Median: ',err_devs]);
subplot(3,1,3);
boxplot(rms_error,'orientation','horizontal','widths',0.25);
set(gca,'XLim',[0, 0.2]); set(gca,'YLim',[0.8, 1.2]);
xlabel('RMS Error Distribution Box Plot');
iqrs = ['IQR: ',sprintf('%1.3f',iqr(smallerr)), ''];
annotation(gcf,'textbox',[0.7565 0.2739 0.07266 0.03043],...
    'String',{iqrs},...
    'FitBoxToText','on','Color',[0,0,1]);
grid on;
%% Pie chart of tracking error distribution

```

```

hb = 0:max(rms_error)/8:max(rms_error); %tweaked distribution
vector
[hi,hr] = hist(rms_error,hb);
figure();
pie(hr);
colormap(winter);
BinNames = {};
%add labeling
for k = 1:length(hb)
    BinNames{k} = strcat(sprintf('%1.2f',hb(k)), '');
end
% Legend on Pie Chart
title('Tracking Error Totals');
legend(fliplr(BinNames), 'Location', 'NorthEastOutside');

%% Rate Dependence
[sr,si] = sort(rate);
srt_error = [];
for k = 1:length(rate)
    j = si(k);
    srt_error(k) = rms_error(j);
end
figure(); plot(sr,srt_error, '.');
grid on; title('RMS Error vs. Tracking Rate');
%% Rate dependent limit cycling
rms_err_dir = [];

for k = 1:length(rate)
    if rate(k) < 0
        rms_err_dir(k) = -rms_error(k);
    else
        rms_err_dir(k) = rms_error(k);
    end
end
figure(); plot(abs(rate),rms_err_dir, '.');
set(gca, 'YLim', [-0.25, 0.25]);
grid on; xlabel('Elevation Tracking Rate, Arcseconds/second');
ylabel('RMS Error, signed by direction');
title('RMS Error and Velocity');
%% Elevation dependence
el_angle = data(:,1);
figure(); plot(el_angle,rms_error, '.');
grid on;
xlabel('Elevation, degrees'); ylabel('RMS Error, arcseconds');
title('Elevation vs. Tracking Error');
%sorted variance
[sa,sai] = sort(el_angle);
sort_aerror = [];
for k = 1:length(el_angle)
    j = sai(k);
    sort_aerror(k) = rms_error(j);
end
figure(); plot(sa,sort_aerror, '.');
grid on; title('Sorted El Angle and Associated Error Value');
xlabel 'Elevation, Degrees'; ylabel 'RMS Error for each angle';
%% Azimuth Dependence

```

```

az_angle = data(:,2);
figure(); plot(az_angle,rms_error, '.');
%% do some data smoothing and collection
azelerr = [];
%remove fliers
j = 1;
for k = 1:length(rms_error)
    if rms_error(k) > 0.1
        azelerr(j,:) = [az_angle(k) el_angle(k) rms_error(k)];
        j = j + 1;
    end
end

azelsmall = downsample(azelerr,10);
tri = delaunay(azelsmall(:,1),azelsmall(:,2));
figure();
trisurf(tri,azelsmall(:,1),azelsmall(:,2),azelsmall(:,3));
%% Sort az el error data
%sort by azimuth
[saz,sazi] = sort(az_angle);
sazel = []; sazerr = [];
for k = 1:length(saz)
    saz(k) = el_angle(sazi(k));
    sazerr(k) = rms_error(sazi(k));
end
figure(); plot3(saz,sazel,sazerr, '.');
%% AZ and EL distribution frequency
% map azimuth angles to 0 < az < 360
az_angle2 = [];
for k = 1:length(az_angle)
    if az_angle(k) < 0
        az_angle2(k) = az_angle(k) + 360;
    else
        az_angle2(k) = az_angle(k);
    end
end
ba = find(az_angle2 > 360); %there's a bug in the above -- fix:
for k = 1:length(ba)
    az_angle2(ba(k)) = az_angle2(ba(k)) - 360;
end
%% For the no-outlier data
% map azimuth angles to 0 < az < 360
az_angle3b = [];
for k = 1:length(az_angle3)
    if az_angle3(k) < 0
        az_angle3b(k) = az_angle3(k) + 360;
    else
        az_angle3b(k) = az_angle3(k);
    end
end
ba = find(az_angle3b > 360); %there's a bug in the above -- fix:
for k = 1:length(ba)
    az_angle3b(ba(k)) = az_angle3b(ba(k)) - 360;
end
%% Bin into azimuth values
%divide 0 -> 360 range into discrete bins
Az_bins = 0:360/24:360;

```

```

j = 2;
%empty arrays for results
Azvals = [];
pop_total = [];
pop_middle = zeros(length(Az_bins),1);
pop_bottom = zeros(length(Az_bins),1);
%err_dev = iqr(rms_error);
%err_devs = strcat(sprintf('%1.3f',err_dev),'');
%loop to collect up the data
for k = 1:length(Az_bins)
    %find indices of values between entries in the bin list
    %was az_angle2, use az_angle3b for no-outlier data
    Azvals = find(az_angle3b >= Az_bins(k) & az_angle3b <
Az_bins(j));
    j = j + 1;
    if j > length(Az_bins)
        j = length(Az_bins);
    end
    %total number of tracking entries in the data by azimuth bin
    pop_total(k) = length(Azvals);
    %number of rms_error entries for each azimuth bin
    for n = 1:length(Azvals)
        if smallerr(Azvals(n)) <= 0.1 %use rms_error for complete
data
            pop_middle(k) = pop_middle(k) + 1;
        end
        if smallerr(Azvals(n)) <= medianError
            pop_bottom(k) = pop_bottom(k) + 1;
        end
    end
end
%pop_middle = pop_middle(1:length(pop_total));
%pop_bottom = pop_bottom(1:length(pop_total)); %extra empty entry
- lose it.
%azline = Az_bins(1:length(pop_bottom));
pct_low = (pop_bottom./pop_total) * 100;
%% Plotting routine for azimuth distribution data
%pop_total has all the tracking error values per azimuth bin
%pop_low has all error values < 0.1"
%bar plot of the total and < 0.1" values
figure();
bar(azline,pop_total,'FaceColor','r','EdgeColor','k');
hold on;
bar(azline,pop_middle,'FaceColor','y','EdgeColor','k');
bar(azline,pop_bottom,'FaceColor','g','EdgeColor','k');
set(gca,'XLim',[-10 360]);
hold off;
ylabel('Tracking Error Population');
legend('Total Error Entries','Error <= 0.1',[ 'Error <= ',...
err_meds]);
%legend('Total Error Entries','Error < 0.1" rms');
legend('Location','NW');
%a line with the % total < 0.1"
h1 = gca;
h2 = axes('Position',get(h1,'Position'));
plot(azline,pct_low,'ok'); hold on;
hs = stem(azline,pct_low);

```

```

set(hs(1), 'MarkerFaceColor', 'black');
set(h2, 'XLim', get(h1, 'XLim'), 'Layer', 'top');
set(h2, 'YAxisLocation', 'right', 'Color', 'none', 'XTickLabel', [])
legend('Percentages', 'Location', 'NE');
grid on;
set(gca, 'Layer', 'Top');
xlabel('Azimuth, degrees');
ylabel('Percent at Median or Below');
title('Tracking Error Distribution over Azimuth');
% Bin error values over elevation
%divide 0 -> 90 range into discrete bins
El_bins = 0:90/15:90;
j = 2;
%empty arrays for results
Elvals = [];
elpop_total = [];
elpop_middle = zeros(length(El_bins),1);
elpop_bottom = zeros(length(El_bins),1);
%loop to collect up the data
for k = 1:length(El_bins)
    %find indices of values between entries in the bin list
    %use el_angle for the complete data set
    Elvals = find(el_angle3 >= El_bins(k) & el_angle3 <
El_bins(j));
    j = j + 1;
    if j > length(El_bins)
        j = length(El_bins);
    end
    %total number of tracking entries in the data by azimuth bin
    elpop_total(k) = length(Elvals);
    %number of rms_error entries for each azimuth bin
    %use rms_error for the complete data set
    for n = 1:length(Elvals)
        if smallerr(Elvals(n)) <= 0.1
            elpop_middle(k) = elpop_middle(k) + 1;
        end
        if smallerr(Elvals(n)) <= medianError
            elpop_bottom(k) = elpop_bottom(k) + 1;
        end
    end
end
%elpop_middle = elpop_middle(1:length(elpop_total));
%elpop_bottom = elpop_bottom(1:length(elpop_total));
%elline = El_bins(1:length(elpop_bottom)); %poplow is 1 entry too
long!
elpct_low = (elpop_bottom'./elpop_total) * 100;
elnans = isnan(elpct_low);
for n = 1:length(elnans)
    if elnans(n) == 1
        elpct_low(n) = 0;
    end
end
end
% Plotting routine for elevation distribution data
%pop_total has all the tracking error values per azimuth bin
%pop_low has all error values < 0.1"
%bar plot of the total and < 0.1" values
figure();

```

```

bar(elline,elpop_total,'FaceColor','r','EdgeColor','k');
hold on;
bar(elline,elpop_middle,'FaceColor','y','EdgeColor','k');
bar(elline,elpop_bottom,'FaceColor','g','EdgeColor','k');
set(gca,'XLim',[-5 95]);
hold off;
ylabel('Tracking Error Population');
legend('Total Error Entries','Error < 0.1" rms',[ 'Error <=
',err_meds]);
legend('Location','NW');
%a line with the % total < 0.1"
h1 = gca;
h2 = axes('Position',get(h1,'Position'));
hs = stem(elline,elpct_low);
set(hs(1),'MarkerFaceColor','black');
set(h2,'XLim',get(h1,'XLim'),'Layer','top');
set(h2,'YAxisLocation','right','Color','none','XTickLabel',[])
legend('Percentages','Location','NE');
grid on;
set(gca,'Layer','Top');
xlabel('Elevation,degrees');
ylabel('Percent at Median or Below');
title('Tracking Error Distribution over Elevation');
%% mesh of az/el distribution
%elmesh = meshgrid(elline,elpct_low);
%azmesh = meshgrid(azline,pct_low);
[azmesh, elmesh] = meshgrid(azline,elline);
%calculation the Z distribution as an RSS of the error
distributions
%%
Zlow = [];
for k = 1:length(azline)
    Zlow(k,:) = sqrt(pct_low(k).^2 + elpct_low.^2);
end
%%
figure();
surf(azmesh',elmesh',Zlow);
colormap(pink);
view([-132,50]);
xlabel('Azimuth Bins'); ylabel('Elevation Bins');
title('RSS of error distributions');

%% Distribution of error, sorted by wind speed
%imported Tom's wind-queried data in as dataw and textdataw, have
5 entries
%for each: 1. # of samples averaged. 2. average wind direction.
3. average
%wind speed. 4. max wind direction. 5. max wind speed.
%v3wind = dataw(:,6:10);
%v4wind = dataw(:,11:end);
%wind speed is in m/s, and Vaisala3 is essentially worthless,
so...
winddir = v4wind(:,2); windspeed = v4wind(:,3);
%sorted variance
[sws,swsi] = sort(windspeed);
sort_werror = [];
for k = 1:length(windspeed)

```

```

        j = swsi(k);
        sort_werror(k) = rms_error(j);
    end
    figure(); plot(sws,sort_werror, '.');
    grid on; title('Sorted Average Wind Speed and Associated Error
Value');
    xlabel('Wind Speed, mph'); ylabel('RMS Error');
    %% Distribution of error over average wind speed, without
    outliers
    maxwind = round(max(windspeedsmall));
    WindBins = 0:maxwind/10:maxwind;
    j = 2;
    %%empty arrays for results
    Windvals = [];
    windpop_total = [];
    windpop_middle = zeros(length(WindBins),1);
    windpop_bottom = zeros(length(WindBins),1);
    %%loop to collect up the data
    for k = 1:length(WindBins)-1
        %%find indices of values between entries in the bin list
        Windvals = find(windspeedsmall >= WindBins(k) &
windspeedsmall < WindBins(j));
        j = j + 1;
        if j > length(WindBins)
            j = length(WindBins);
        end
        %%total number of tracking entries in the data by the bin
        windpop_total(k) = length(Windvals);
        %%number of rms_error entries for each bin
        for n = 1:length(Windvals)
            if smallerr(Windvals(n)) <= 0.1 %was rms_error
                windpop_middle(k) = windpop_middle(k) + 1;
            end
            if smallerr(Windvals(n)) <= medianError
                windpop_bottom(k) = windpop_bottom(k) + 1;
            end
        end
    end
    windpop_middle = windpop_middle(1:length(windpop_total));
    windpop_bottom = windpop_bottom(1:length(windpop_total));
    windline = WindBins(1:length(windpop_bottom));
    windpct_low = (windpop_bottom./windpop_total) * 100;
    %% Plotting for the wind speed - rms error distribution
    figure();
    bar(windline,windpop_total,'FaceColor','r','EdgeColor','k');
    hold on;
    bar(windline,windpop_middle,'FaceColor','y','EdgeColor','k');
    bar(windline,windpop_bottom,'FaceColor','g','EdgeColor','k');
    hold off;
    ylabel('Tracking Error Population');
    legend('Total Error Entries','Error < 0.1" rms',[ 'Error <=
',err_meds]);
    legend('Location','NE');
    %%a line with the % total < 0.1"
    h1 = gca;
    h2 = axes('Position',get(h1,'Position'));
    hs = stem(windline,windpct_low);

```

```

set(hs(1), 'MarkerFaceColor', 'black');
set(h2, 'XLim', get(h1, 'XLim'), 'Layer', 'top');
set(h2, 'YAxisLocation', 'right', 'Color', 'none', 'XTickLabel', [])
legend('Percentages', 'Location', 'NW');
grid on;
set(gca, 'Layer', 'Top');
xlabel('Average Wind Speed, mph');
ylabel('Percent At Median or Below');
title('Tracking Error Distribution over Wind Speed');
%% Accounting for the relative wind exposure
%%wind influence functions
%used the curve fitting toolbox to get the following curves-
azA = 0.98; azB = -0.02337; azC = 0.03265; azD = 0.01747;
elA = -0.000346; elB = 0.02889; elC = 0.4;
%calculate absolute relative wind (e.g. 0 -> 180)
wind_angle = abs(az_angle3 - winddirsmall);
for k = 1:length(wind_angle)
    if wind_angle(k) > 180
        wind_angle(k) = abs(wind_angle(k) - 360);
    end
end
end
%%
expaz = (azA * exp(azB * wind_angle)) + (azC * exp(azD *
wind_angle));
expel = (elA * el_angle.^2) + (elB * el_angle) + elC;
windexp = (expaz + expel) .* windspeed';
%write sorting routine to sort this into wind exposure # and get
a bar plot
%% Wind Angle distribution of tracking error
%divide 0 -> 180 relative wind range into discrete bins
Angle_bins = 0:180/30:180;
j = 2;
%empty arrays for results
Anglevals = [];
angpop_total = [];
angpop_middle = zeros(length(Angle_bins),1);
angpop_bottom = zeros(length(Angle_bins),1);
%err_dev = iqr(rms_error);
%err_devs = strcat(sprintf('%1.3f',err_dev),'');
%loop to collect up the data
for k = 1:length(Angle_bins)
    %find indices of values between entries in the bin list
    Angvals = find(wind_angle >= Angle_bins(k) & wind_angle <
Angle_bins(j));
    j = j + 1;
    if j > length(Angle_bins)
        j = length(Angle_bins);
    end
    %total number of tracking entries in the data by azimuth bin
    angpop_total(k) = length(Angvals);
    %number of rms_error entries for each azimuth bin
    for n = 1:length(Angvals)
        if smallerr(Angvals(n)) <= 0.1
            angpop_middle(k) = angpop_middle(k) + 1;
        end
        if smallerr(Angvals(n)) <= medianError
            angpop_bottom(k) = angpop_bottom(k) + 1;
        end
    end
end

```

```

        end
    end
end
%angpop_middle = angpop_middle(1:length(angpop_total));
%angpop_bottom = angpop_bottom(1:length(angpop_total)); %extra
empty entry - lose it.
%angline = Angle_bins(1:length(angpop_bottom));
angpct_low = (angpop_bottom./angpop_total) * 100;
%% Plotting routine for relative wind error distribution
figure();
bar(angline,angpop_total,'FaceColor','r','EdgeColor','k');
hold on;
bar(angline,angpop_middle,'FaceColor','y','EdgeColor','k');
bar(angline,angpop_bottom,'FaceColor','g','EdgeColor','k');
set(gca,'XLim',[-5 185]);
hold off;
ylabel('Tracking Error Population');
legend('Total Error Entries','Error < 0.1" rms',[ 'Error <=
',err_meds]);
legend('Location','NE');
%a line with the % total < 0.1"
h1 = gca;
h2 = axes('Position',get(h1,'Position'));
hs = stem(angline,angpct_low);
set(hs(1),'MarkerFaceColor','black');
set(h2,'XLim',get(h1,'XLim'),'Layer','top');
set(h2,'YAxisLocation','right','Color','none','XTickLabel',[]);
legend('Percentages','Location','NW');
grid on;
set(gca,'Layer','Top');
xlabel('Relative Wind Azimuth');
ylabel('Percent At Median or Below');
title('Tracking Error Distribution over Relative Azimuths');
%% Wind exposure influence function
%Binnitize
expmax = round(max(windexp));
exp_bins = 0:expmax/30:expmax;
j = 2;
%empty arrays for results
Expvals = [];
exppop_total = [];
exppop_middle = zeros(length(exp_bins),1);
exppop_bottom = zeros(length(exp_bins),1);
%err_dev = iqr(rms_error);
%err_devs = strcat(sprintf('%1.3f',err_dev),'');
%loop to collect up the data
for k = 1:length(exp_bins)
    %find indices of values between entries in the bin list
    expvals = find(windexp >= exp_bins(k) & windexp <
exp_bins(j));
    j = j + 1;
    if j > length(exp_bins)
        j = length(exp_bins);
    end
    %total number of tracking entries in the data by azimuth bin
    exppop_total(k) = length(expvals);
    %number of rms_error entries for each azimuth bin

```

```

    for n = 1:length(expvals)
        if rms_error(expvals(n)) <= 0.1
            exppop_middle(k) = exppop_middle(k) + 1;
        end
        if rms_error(expvals(n)) <= medianError
            exppop_bottom(k) = exppop_bottom(k) + 1;
        end
    end
end
exppop_middle = exppop_middle(1:length(exppop_total));
exppop_bottom = exppop_bottom(1:length(exppop_total)); %extra
empty entry - lose it.
expline = exp_bins(1:length(exppop_bottom));
expct_low = (exppop_bottom'./exppop_total) * 100;
%% Plot the wind influence function error distribution
figure();
bar(expline,exppop_total,'FaceColor','r','EdgeColor','k');
hold on;
bar(expline,exppop_middle,'FaceColor','y','EdgeColor','k');
bar(expline,exppop_bottom,'FaceColor','g','EdgeColor','k');
hold off;
ylabel('Tracking Error Population');
legend('Total Error Entries','Error < 0.1" rms',[ 'Error <=
',err_devs]);
legend('Location','NE');
%a line with the % total < 0.1"
h1 = gca;
h2 = axes('Position',get(h1,'Position'));
hs = stem(expline,expct_low);
set(hs(1),'MarkerFaceColor','black');
set(h2,'XLim',get(h1,'XLim'),'Layer','top');
set(h2,'YAxisLocation','right','Color','none','XTickLabel',[])
legend('Percentages','Location','NW');
grid on;
set(gca,'Layer','Top');
xlabel('Wind Exposure Value');
ylabel('Percent At IQR or Below');
title('Tracking Error Distribution over Wind Exposure Function');
%% 3-D graph of relative wind angle, elevation, and rms error
%use the bins for relative wind angle (0..180) and elevation
(0..90)
[azmesh, elmesh] = meshgrid(angline,elline);
%calculate z-heights
Zh = [];
for k = 1:length(angline)
    Zh(k,:) = (angpct_low(k) + elpct_low) / 2;
    if (isnan(Zh(k,:)))
        Zh(k,:) = Zh(k-1,:);
    end
end
end
%bug, got NaN, investigate later...
%plot!
figure();
surf(azmesh',elmesh',Zh);
title('Error Distribution Over Relative Azimuth and Elevation');
xlabel('Relative Wind Angle'); ylabel('Elevation');
view([-137, 40])

```

```
%% Wind data display
figure(); subplot(1,2,1);
hist(windspeed,20); grid on; xlabel('Wind Speed, mph');
ylabel('Population of Wind Speed');
legend(['Median Wind:
',sprintf('%2.1f',median(windspeed)), 'mph']);
title('Vaisala 4 Sensor Wind Data');
subplot(1,2,2);
rose(winddir);xlabel('Wind Direction');
title('Vaisala 4 Sensor Wind Data');
```

-----END-----
