APPENDIX F:
ACROSS-WIND EXCITATION ALGORITHM

SCOPE

This appendix presents the method used to identify across-wind excitation of a high-mast lighting tower (HMLT) using data from strain gages, or channels, placed around the perimeter of the pole. Along with analysis of stress-range histogram data, this method was used to evaluate the effectiveness of devices for mitigating vortex shedding. For this research, across-wind excitation is defined as a resonant excitation of the HMLT pole in a plane orthogonal to the direction of the wind. Modal frequencies two and three are most commonly induced by vortex shedding for structures of this type; however, the method may be applied to the fundamental frequency or frequencies higher than mode three if required. The method consists of two steps: identifying strong resonant oscillations in the sensor signals and sorting the signals into along-wind and across-wind directions.

IDENTIFYING STRONG RESONANT OSCILLATIONS

Signal Properties of Across-Wind Excitation

Tapered, slender structures such as HMLTs are subject to two types of wind-induced loading: buffeting and vortex shedding. Buffeting is associated with higher wind speeds when gusts of wind act to excite a structure. Oscillations of this type occur in the along-wind direction. Signal response from buffeting is generally limited to the fundamental mode and exhibits variable amplitude behavior. Vortex shedding is associated with lower wind speeds when periodic lateral forces act to excite a structure. Oscillations of this type occur in the across-wind direction. The signal response from vortex shedding generally occurs in higher modes and exhibits a constant amplitude behavior. This constant amplitude behavior becomes most prominent when the aero-elastic, or ``lock-in'', phenomena occurs where the frequency of vortex shedding matches the modal frequency of the structure.

A sample signal illustrating the time-domain and frequency-domain characteristics of buffeting is provided in Figure F-1. Notice the signal consists of three main frequencies: the first, second and third modal frequencies. Also, notice the fundamental period is the primary component of the signal; this is typical for buffeting induced oscillation.

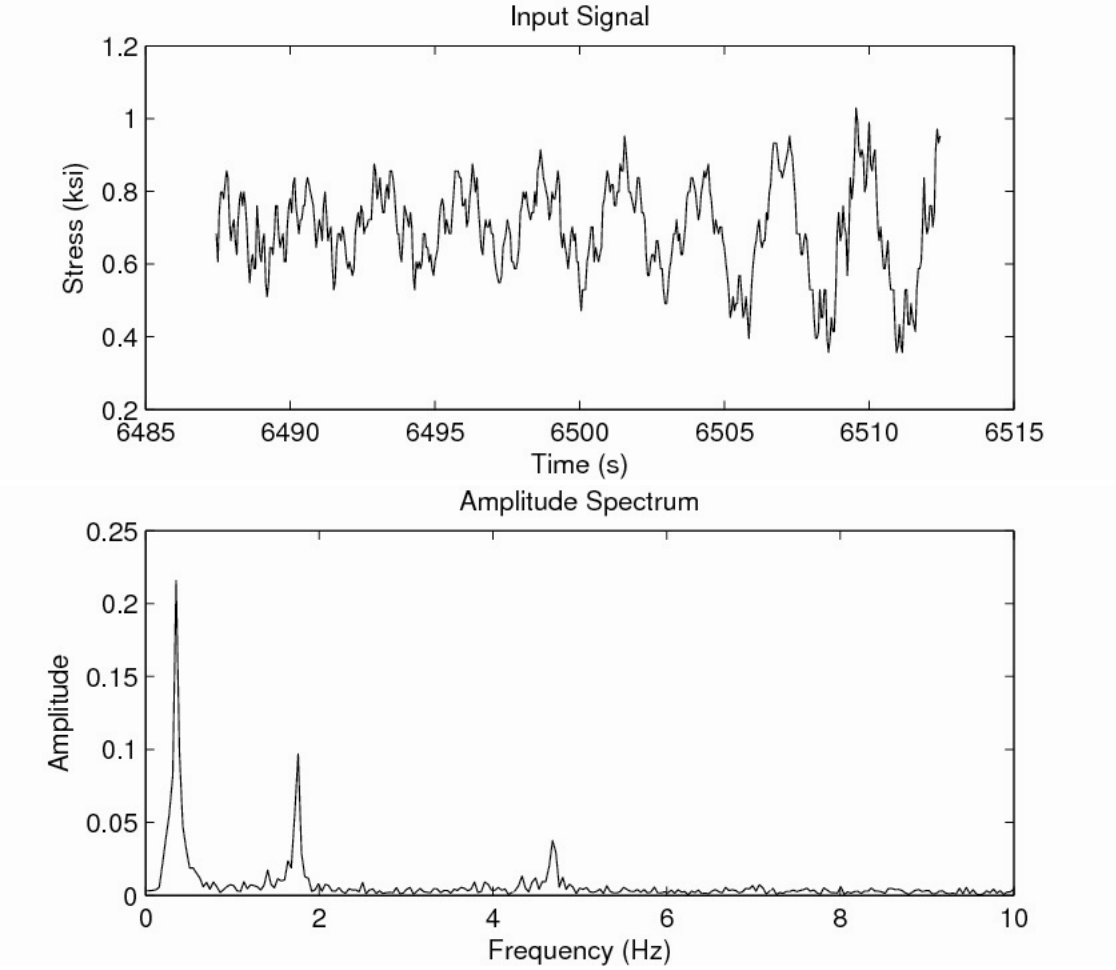

Figure F-1: Sample signal – buffeting

A sample signal illustrating the time-domain and frequency-domain characteristics of vortex shedding is provided in Figure F-2. The sample was taken from the same signal used for Figure F-1, but at a later point in time. Notice the second modal frequency is now the primary component of the signal. The lock-in effect causes an increase in the amplitude of the modal frequency associated with vortex shedding. This increase in amplitude will serve as a sign that an across-wind excitation has occurred.

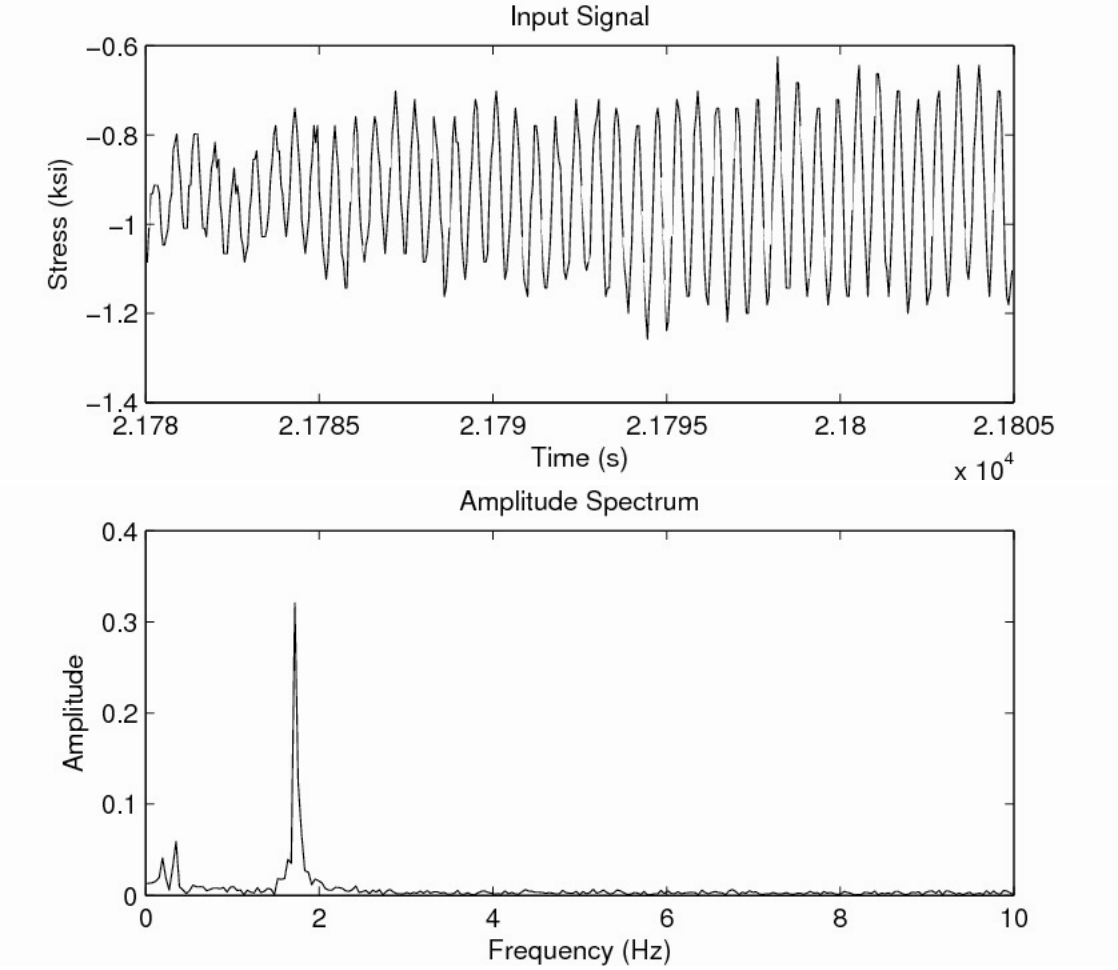

Figure F-2: Sample signal – vortex shedding

Frequency Amplitude Time-history

Identifying occurrences of across-wind excitation becomes troublesome for large amounts of data. Searching through days, or even hours, of time-history data for periods of constant amplitude behavior at a specific frequency is unrealistic. By dividing the time-history into a number of short-term intervals and performing a Fast Fourier Transform (FFT) on each interval, the transitions can be observed as increases or decreases in frequency amplitude from one interval to the next. Peak amplitude values can be collected by assuming a reasonable passband centered about the modal frequency of interest. A 128 point FFT performed on a signal sampled at 20 Hz will occupy a 6.4 second interval in the time-domain. Plotting the peak frequency amplitude values at intervals of 6.4 seconds will result in a low-resolution graph where the transitions can be easily identified. This is illustrated in Figure F-3. The graph includes the data presented in Figure F-2.

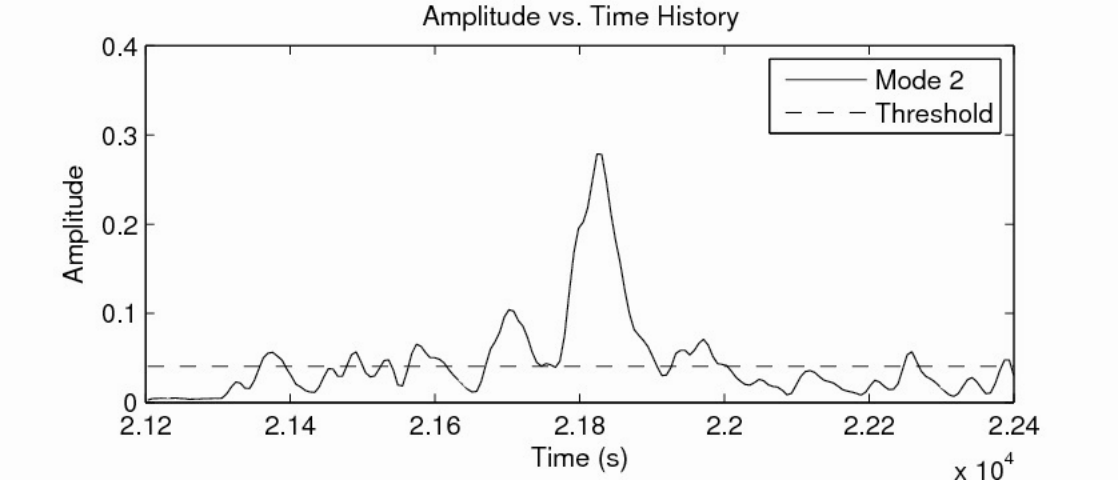

Figure F-3: Sample frequency amplitude time-history

The ambient data collected in this research was sampled at 20 Hz. The algorithm used a 256 point FFT to generate the individual frequency spectrums, and stepped through the time-domain at 128 point intervals. This was done to achieve greater resolution in the frequency spectrum for peak picking and to smooth the output time-history.

Cataloging Events

After creating the frequency time-history, periods of excitation, or events, are identified and cataloged. Events are defined as a period of time in the signal during which significant excitation has occurred. To qualify as an event, the frequency amplitude for the mode of interest must exceed an amplitude threshold for a minimum amount of time. In Figure F-3, the threshold is plotted with the amplitude time-history. The threshold is determined by adding one standard deviation to the mean value of the amplitude time-history. The mean and standard deviation are calculated using data from all the strain gages located around the perimeter of the pole. The strain gages with the minimum and maximum responses are excluded from the data set since errors or noise are commonly recorded as ``not-a-number'' or unusually large values. An event must last for a minimum of 35 seconds to qualify, which is a reasonable amount of time from observed behavior. Data recorded for each event includes: channel number, time period, average wind speed, average wind direction, average frequency amplitude, and peak stress range.

SORTING

Selecting the Dominant Channel

After cataloging all potential events for all the strain gages, or channels, they are sorted by start time. Where multiple channels overlap in a single event, the channels with the maximum amplitude are selected and the others are discarded. Typically, at least two channels overlap since the sensors are usually opposite each other on the pole. For particularly strong events, more than

two channels will qualify as an event. For this case, the maximum value, or dominant channel, is considered most representative of the event.

Categorizing as Along-wind or Across-wind

To qualify an excitation as across-wind, the location of the dominant channel must be checked against the average recorded wind direction. To do this, boundaries are established for the across-wind direction which consists of the two quadrants orthogonal to the average wind direction. The boundaries are illustrated in Figure F-4. Using this method of sorting, a signal cannot be considered across-wind if there is a greater response in the along-wind direction.

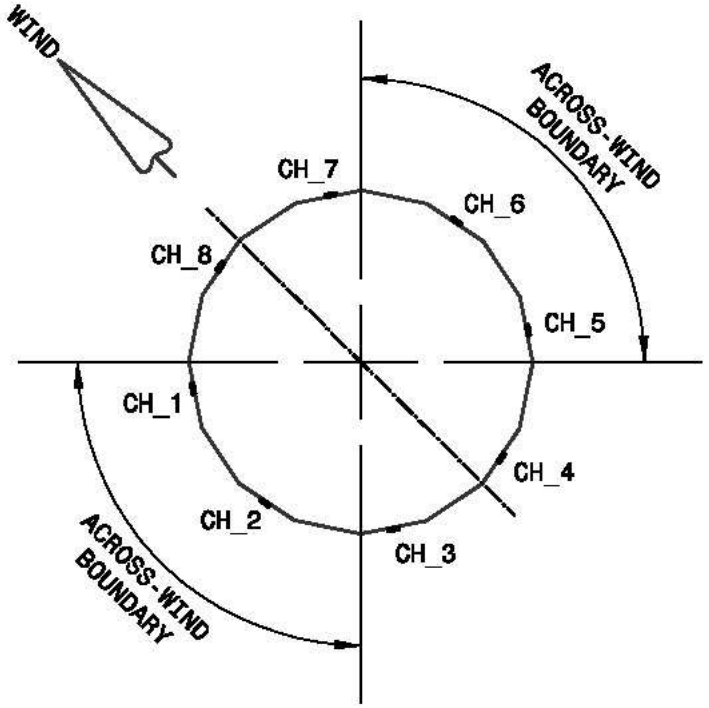

Figure F-4: Across-wind boundaries

Note that the average wind direction must be computed using a radial, or circular, mean rather than numerical. Otherwise, errors may occur where a mean value of zero degrees is interpreted as 180 degrees, etc.

OUTPUT

Lastly, the sorted events are presented in tabular format. This output is useful for manipulating the data in programs such as MS Excel. Tabulated data includes: channel number, event start time, event end time, average wind direction, average wind speed, average frequency amplitude, and peak stress range. The two key parameters from each event are stress range and wind speed; these are the values plotted to show instances of across-wind excitation. A sample of the tabulated output is given in Table F-1; note, the table includes the event shown in Figure F-3.

Channel No.	Start Time (s)	End Time (s)	Wind Dir. (deg)	Wind Speed (mph)	Amp.	Peak S_R (ksi)
3	21357	21402	333	5.23	0.10	0.16
3	21562	21626	12	5.88	0.16	0.28
8	21779	21907	300	7.23	0.31	0.65
8	21926	21990	308	5.75	0.11	0.20

Table F-1: Sample tabulated output

Using this method, the large amounts of ambient data collected for this study could be easily processed and instances of across-wind excitation could be quantified.

MATLAB CODE

```
% VS_CONVERT - A routine to create frequency vs. time data
% for modal frequencies 1, 2, & 3 and store the data in a file
% for use with VS_EVENTS routine

clear all
dead_ga = [];
mfname = uigetfile('*.mat','MAT file to load...');
load(mfname);

step = 128;         %Size of time step - must be power of 2
fftwin = 2*step;       %Size of FFT window - must be power of 2 >= step
[Nsmp Nch] = size(M);   %Nsmp = # of samples, Nch = # of channels
Nfft = ceil((Nsmp-fftwin)/step);      %Nfft = # of FFT segments
T_0 = time(1);      %Timestamp at beginning of data stream
time = time - T_0;      %Modify time vector to start at zero

%Preallocate output amplitude matrix, AM, and output time vector, Tm
AM_1 = zeros(Nfft,Nch);
AM_2 = zeros(Nfft,Nch);
AM_3 = zeros(Nfft,Nch);
Tm(1:Nfft,1) = 0;

%Boundary values for mode 1 frequencies
mode_1 = [0.20 0.50];
mode_1 = [floor(mode_1(1)*(fftwin/2+1)*2/smp),...
    ceil(mode_1(2)*(fftwin/2+1)*2/smp)];

%Boundary values for mode 2 frequencies
mode_2 = [0.70 2.3];     %0.70 2.3 original boundaries
mode_2 = [floor(mode_2(1)*(fftwin/2+1)*2/smp),...
    ceil(mode_2(2)*(fftwin/2+1)*2/smp)];

%Boundary values for mode 3 frequencies
mode_3 = [2.6 5];   %2.6 5 original boundaries
mode_3 = [floor(mode_3(1)*(fftwin/2+1)*2/smp),...
    ceil(mode_3(2)*(fftwin/2+1)*2/smp)];

modes = [mode_1; mode_2; mode_3];

%Build amplitude vs. time history data for modes 1 & 2
for n = 1:Nfft;         %For each segment
    Tm(n) = median(time(((n-1)*step+1):((n-1)*step+fftwin)));
    for j = 1:Nch;          %For each channel
        Td = M(((n-1)*step+1):((n-1)*step+fftwin),j);   %Td = segment vector
        Td = Td-mean(Td);      %Remove mean
        [b a] = cheby1(4,0.5,0.02,'high');      %Filter parameters
        Td = filtfilt(b,a,Td);        %Perform filter
        Fd = fft(Td,fftwin);      %Perform FFT
        AM_1(n,j) = max(abs(Fd(mode_1(1):mode_1(2))));
        AM_2(n,j) = max(abs(Fd(mode_2(1):mode_2(2))));
        AM_3(n,j) = max(abs(Fd(mode_3(1):mode_3(2))));
    end
```

```
end

%Plot output
go = input('\nPlot amplitude vs. time data? y/n <n>:  ','s');
if isempty(go), go = 'n'; end
while go == 'y'
    sig = input('\nChannel to plot:  ');
    plot_amp(Tm,AM_1(:,sig),AM_2(:,sig),0);
    go = input('\nChoose another channel? y/n <n>:  ','s');
    if isempty(go), go = 'n'; end
end

%Save data Y/N?
go = input('Save data? y/n <n>:  ','s');
if isempty(go);
    go = 'n';
end
if (go=='y');
    mfname = strrep(mfname,'_w','_x');
    fname = input(['File name for ambient data <' mfname '>:  '],'s');
    if isempty(fname), fname = mfname; end
    save(fname,'M','smp','time','wnd',...
        'AM_1','AM_2','AM_3','step','T_0','Tm','dead_ga');
end
%End routine
```

```
% VS_EVENTS - A routine to search for and catalog across-wind events

clear all
filename_r = uigetfile('*.mat','MAT file to load...');
load(filename_r);

site = site_menu();

switch site
    case 1
        modes = [0.2 0.5; 1.5 2.0; 4.4 5.0];    %CA
        N_ga = 8;
    case 2
        modes = [0.2 0.5; 1.4 1.9; 4.1 4.7];    %KS
        N_ga = 6;
    case 3
        modes = [0.2 0.5; 1.0 1.5; 2.9 3.5];    %ND
        N_ga = 6;
    case 4
        modes = [0.2 0.5; 0.8 1.3; 2.5 3.2];    %OKNE
        N_ga = 8;
    case 5
        modes = [0.2 0.5; 1.0 1.5; 3.0 3.6];    %OKSW
        N_ga = 8;
    case 6
        modes = [0.2 0.5; 0.9 1.4; 2.7 3.3];    %SD
        N_ga = 8;
    case 7
        modes = [0.2 0.5; 1.3 1.8; 3.6 4.2];    %WYCJE
        N_ga = 8;
    case 8
        modes = [0.2 0.5; 1.3 1.8; 3.6 4.2];    %WYCJW
        N_ga = 8;
    case 9
        modes = [0.2 0.5; 1.1 1.6; 3.1 3.7];    %IA
        N_ga = 6;
end

%Get events for which mode?
while 1
    mode = input('\nExamine which mode of vibration?  2/3:  ');
    if mode == 2, AM_V = AM_2;
    elseif mode == 3, AM_V = AM_3;
    end
    if (mode == 3)||(mode == 2), break; end
end

%Clean data & determine threshold amplitude
N_cons = 1:N_ga;
N_cons(dead_ga) = [];

th_amp = [];                    th_std = [];

for gg = N_cons
    am_v = AM_V(:,gg);
    real_nums = logical(true(size(am_v)) - isnan(am_v));
```

```
    th_amp = [th_amp mean(am_v(real_nums))];
    th_std = [th_std std(am_v(real_nums)) ];
end

[~,a] = max(th_amp);
[~,b] = min(th_amp);
th_amp([a,b]) = [];             th_std([a,b]) = [];

thresh = mean(th_amp+th_std);

AM_V(isnan(AM_V)) = mean(th_amp);

%Find VS events
events = [];
for sig = N_cons;
    [Mpts Npts] = peak_pass(AM_V(:,sig),smp,step,thresh);
    for ii = 1:size(Mpts,2)
        dir_avg = wind_dir_avg(wnd(Npts(1,ii):Npts(2,ii),1));
        spd_avg = mean(wnd(Npts(1,ii):Npts(2,ii),2));
        amp_avg = mean(AM_V(Mpts(1,ii):Mpts(2,ii),sig));
        t1 = Npts(1,ii)/smp;
        t2 = Npts(2,ii)/smp;
        rng = M(Npts(1,ii):Npts(2,ii),sig);
        [N,D] = cheby1(4,0.5,modes(mode,:)*2/smp);
        mod_rng = filtfilt(N,D,rng);
        Sr_pk = max(mod_rng)-min(mod_rng);
        events = [events; sig t1 t2 dir_avg spd_avg amp_avg Sr_pk];
    end
end

%Sort events by time they occur
events = sortrows(events,[2 3]);
[R,~] = size(events);
events_sort = [];
block = events(1,:);        %block = group of overlapping time stamps
block_tot = {};
for jj = 2:R
    [C,I] = max(block,[],1);
    if events(jj,2) > C(3) || jj == R
        %Check for valid wind direction
        switch site
            case 1
                %NOTE: set to collect events based on AMP values, [I(6)]
                if normal_ca(block(I(6),1),block(I(6),4))
                    events_sort = [events_sort; block(I(6),:)];
                end
            case 2
                if normal_ks(block(I(6),1),block(I(6),4))
                    events_sort = [events_sort; block(I(6),:)];
                end
            case 3
                if normal_nd(block(I(6),1),block(I(6),4))
                    events_sort = [events_sort; block(I(6),:)];
                end
            case 4
                if normal_okne(block(I(6),1),block(I(6),4))
```

```
                    events_sort = [events_sort; block(I(6),:)];
                end
            case 5
                if normal_oksw(block(I(6),1),block(I(6),4))
                    events_sort = [events_sort; block(I(6),:)];
                end
            case 6
                if normal_sd(block(I(6),1),block(I(6),4))
                    events_sort = [events_sort; block(I(6),:)];
                end
            case 7
                if normal_cje(block(I(6),1),block(I(6),4))
                    events_sort = [events_sort; block(I(6),:)];
                end
            case 8
                if normal_cjw(block(I(6),1),block(I(6),4))
                    events_sort = [events_sort; block(I(6),:)];
                end
            case 9
                if normal_ia(block(I(6),1),block(I(6),4))
                    events_sort = [events_sort; block(I(6),:)];
                end
        end
        block_tot = [block_tot; block];
        block = events(jj,:);
    else
        block = [block; events(jj,:)];
    end
end

filename_w = strrep(filename_r,'.mat',['m',num2str(mode),'.txt']);
print_events(events_sort,filename_w);
```

```
% SITE_MENU - A subroutine to VS_EVENTS

function [choice] = site_menu()

fprintf('\n');
fprintf('\t[1]\tCA\n');
fprintf('\t[2]\tKS\n');
fprintf('\t[3]\tND\n');
fprintf('\t[4]\tOKNE\n');
fprintf('\t[5]\tOKSW\n');
fprintf('\t[6]\tSD\n');
fprintf('\t[7]\tWYCJE\n');
fprintf('\t[8]\tWYCJW\n');
fprintf('\t[9]\tIA\n');
fprintf('\n');
while 1
    choice = input('Enter site #:  ');
    if (choice(1) >= 1)&&(choice(1) <= 9), break; end
end
```

```
% PEAK_PASS - A function identifying peaks in amplitude vs. time
% history
% Subroutine to VS_EVENTS

function [m_out n_out] = peak_pass(am_v,rate,points,th)

minwin = 35;        %Minimum length of vortex shedding event to record
window = ceil(minwin*rate/points);      %Number of data points in minwin

mi = 1;
m_out = [];
while mi < size(am_v,1)-window
    if all(am_v(mi:mi+window) > th)
        vs = am_v(mi:mi+window);
        mj = mi+window+1;
        while (mj < numel(am_v))&&(am_v(mj) > th)
            %vs = [vs am_v(mj)];
            mj = mj + 1;
        end
        m = [mi; mj];
        m_out = [m_out m];
        mi = mj;
    else
        mi = mi + 1;
    end
end

n_out = m_out*points;

end
```

```
% WIND_DIR_AVG – A function for averaging polar directions
% subroutine to VS_EVENTS

function [wdir_avg] = wind_dir_avg(wdir)

%   Normalize values for north winds (360 ==> 0)
Di = wdir(1);
SD = wdir(1);
for ii = 2:size(wdir,1)
    delta = wdir(ii) - Di;
    if abs(delta) < 180
        Di = Di + delta;
    elseif delta < -180
        Di = Di + delta + 360;
    elseif delta >  180
        Di = Di + delta - 360;
    end
    SD = SD + Di;
end

%   Calculate average value
wdir_avg = SD/size(wdir,1);

if wdir_avg < 0
    wdir_avg = wdir_avg - 360*floor(wdir_avg/360);
elseif wdir_avg > 360
    wdir_avg = wdir_avg - 360*floor(wdir_avg/360);
end

end
```

```
% NORMAL_XX - A logic function to determine across-wind direction
% subroutine to VS_EVENTS
% ‘XX’ is an identifier for a specific site, ‘CA’ is provided as an example

function [go] = normal_ca(ch,wdir)
go = 0;
switch ch
    case 1
        if ((wdir>=34)&&(wdir<=124))||((wdir>=214)&&(wdir<=304))
            go = 1;
        end
    case 2
        if ((wdir>=0)&&(wdir<=79))||((wdir>=169)&&(wdir<=259))||...
                ((wdir>=349)&&(wdir<=360))
            go = 1;
        end
    case 3
        if ((wdir>=0)&&(wdir<=34))||((wdir>=124)&&(wdir<=214))||...
                ((wdir>=304)&&(wdir<=360))
            go = 1;
        end
    case 4
        if ((wdir>=79)&&(wdir<=169))||((wdir>=259)&&(wdir<=349))
            go = 1;
        end
    case 5
        if ((wdir>=34)&&(wdir<=124))||((wdir>=214)&&(wdir<=304))
            go = 1;
        end
    case 6
        if ((wdir>=0)&&(wdir<=79))||((wdir>=169)&&(wdir<=259))||...
                ((wdir>=349)&&(wdir<=360))
            go = 1;
        end
    case 7
        if ((wdir>=0)&&(wdir<=34))||((wdir>=124)&&(wdir<=214))||...
                ((wdir>=304)&&(wdir<=360))
            go = 1;
        end
    case 8
        if ((wdir>=79)&&(wdir<=169))||...
                ((wdir>=259)&&(wdir<=349))
            go = 1;
        end
end

end
```