
IEEE P802.15
Wireless Personal Area Networks

Project	IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs)		
Title	Matlab code for generating BAN fading power profile		
Date Submitted	21 November, 2008		
Source	[David Smith, Dino Miniutti, Andrew Zhang, Leif Hanlen] [NICTA] 7 London Circuit, ACT 2600, Australia	Voice: Fax: E-mail:	+61262676256 +61262676220 dino.miniutti@nicta.com.au
Re:			
Abstract	Matlab code that can be used to generate a signal (in terms of received power with respect to transmit power) for a mobile wireless body area network based on extensive measurement campaign carried out at NICTA.		
Purpose	To aid the generation of realistic BAN channels.		
Notice	This document has been prepared to assist the IEEE P802.15. It is offered as a basis for discussion and is not binding on the contributing individual(s) or organization(s). The material in this document is subject to change in form and content after further study. The contributor(s) reserve(s) the right to add, amend or withdraw material contained herein.		
Release	The contributor acknowledges and accepts that this contribution becomes the property of IEEE and may be made publicly available by P802.15.		

```
function [signal,time_samp] = generate_power_profile_wmban()
% Generate a signal (in terms of received power with respect to transmit
% power) for a mobile wireless body area network based on extensive
% measurement campaign carried out at NICTA.
%
% Author: David Smith ; affiliation: National ICT Australia (NICTA)
% With assistance from Jian (Andrew) Zhang and Dino Miniutti
% Date: 21 Nov 2008
%
% Return values:
%   signal      - Received power profile relative to transmit power (dB)
%   time_samp   - Timing epoch of each sample in 'signal' vector (s)
%
% Aim : - Generation of a signal power profile (signal variable) with
% fading statistics that approximately match statistics from NICTA's
% extensive measurement campaign reported by Miniutti et al. [Miniutti08].
%
% The following definitions apply to generated signal -
%   Fade:                Contiguous portion of signal that crosses
%                        below the mean signal power.
%   Fade duration:       Time below mean signal power for faded
%                        portion of signal.
%   Fade depth/magnitude: Maximum attenuation of individual fade
%                        portion relative to mean (within each
%                        individual fade).
%   Level crossing rate: The inverse of the time where power profile
%                        drops below mean to when it next crosses
%                        below mean.
%
% In line with the aim of this function -
%   We generate the power profile of a signal around its mean over time
%   such that the attenuated portions of this signal (fades)
%   statistically match the fades found in NICTA's measurement campaign
%   [Miniutti08].
%   In particular, this function attempts to generate a signal whereby
%   the statistical distribution of its: 1) fade durations; 2)
%   fade magnitudes; and 3) level crossing rate match the respective
%   distributions found in NICTA's channel measurements. The statistical
%   distribution of the generated signal around the mean also matches
%   that of NICTA's channel measurements.
%
% The following is a summary of the method used: -
%   1. Generate an initial signal that matches the desired fading
%      statistics using order statistics.
%   Notes:
%     - An appropriate set of Weibull distributed [Weibull1951]
%       random numbers are generated according to best fit to signal
%       statistical distribution around mean from NICTA's
%       measurements.
%     - Jake's model is used to generate a Rayleigh fading power
%       profile with an appropriate rate of fading.
%     - A Weibull fading signal power profile is generated by ordering
%       Weibull distributed random numbers according to the ordering
```

```
%      of the Rayleigh power profile.
%      - This signal will have appropriate fade durations and level
%      crossing rates
%
%      2. The initial signal is then manipulated such that in such a way
%      to make its fade depth statistics match those found in NICTA's
%      measurements. This occurs in the following manner:
%      2a. The signal is treated in portions, each portion being a fade
%      followed by a non-fade (a contiguous portion of the signal
%      above the mean power).
%      2b. Iterate through all the signal portions and compare each
%      fade to the desired fade depth statistics. The current
%      signal portion is manipulated depending on this comparison
%      in one of three ways.
%          - Good match to desired statistics: Keep the current
%            signal portion and insert it into the output signal.
%          - Too much attenuation: Remove the parts of the fade that
%            have too much attenuation. The remaining signal portion
%            is then inserted into the output signal.
%          - Bad match to desired statistics: Discard the current
%            signal portion.
%      2c. Due to the unnatural ordering that results from the fade
%      depth manipulation described above, the non-discarded
%      portions of the signal are reordered (though not reordering
%      within portions).
%      Note: This manipulation does not significantly affect match of
%      level crossing rates and fade duration of signal to statistics
%      from NICTA's measurements.
%
%      3. Adjust the final signal so its mean is equal to the mean
%      specified by the user.
%
%      Note: This function calls a number of sub-functions to perform the above
%      steps. These sub-functions contain more detailed explanation of the
%      method described here.
%
%      Note: A number of simulation parameters may be set in the following
%      section of the code.

% First initialize rand, Matlab's function to produce uniform randomly
% distributed numbers on [0,1] to different state to previous use. This will
% be used in generation of various random numbers.
rand('state', sum(100*clock));

%-----
% Simulation parameters
%-----

% The simulation parameters below may be changed to appropriate values of
% your choosing.
```

```
% Time (s)
time = 20;
% Time in seconds over which signal is generated

% Carrier frequency (MHz)
car_frequency = 820;
% Carrier frequency of transmitted signal.
% * Allowed range: 400 -- 2500 MHz
% * 820 MHz corresponds to NICTA's measurements.

% Sample rate (kHz)
sample_rate = 1;
% Sample rate of received power profile.
% * Allowed range: 0.75 -- 15 kHz.
% * 1 kHz corresponds to NICTA's measurements.

% Relative body movement velocity (km/h)
vel = 1.5 + 4 * rand;
% This is a rough approximation of Rx movement velocity with respect to
% scatterers and/or transmitter Tx. It is used to generate a signal with
% an appropriate rate of fading in the signal generating sub-function to this
% function.
% * Allowed range: 1.5 -- 20 km/h
% * Values of 1.5 -- 5.5 to correspond best to NICTA's measurements.

% Mean path loss (dB)
a = 60.2; % Scale parameter
b = 6.6; % Shape parameter
mean_path_loss = a*(-log(rand))^(1/b);
% This statistic gives a Weibull distributed random number representing
% mean path loss. With the values of a=60.2 and b=6.6, the statistic
% approximately reflects Weibull distribution of mean path loss from
% NICTA's measurement campaign, but this mean path loss can be set
% arbitrarily to any value of your choosing.
% * Note also this statistic follows directly from the Weibull cumulative
% distribution function.
% * Recommended range: 35 -- 70 dB
% Note: Appropriate generation of a set of Weibull distributed faded
signals
% with a varied set of typical mean path losses according to the above
% statistic results in a distribution characteristic approximately
matching that
% of a specific Lognormal distribution. This specified Lognormal
% distribution with log mean of -13.3 and log standard deviation 2.49
% was reported on NICTA's measurement campaign, as a best fit to the
% 12.7 million measurements (where mean path loss is not removed)
[Miniutti08].
```

```

%-----
% Generate signal according to above parameters
%-----

signal = gen_sig_wmban(vel, car_frequency, sample_rate, time,
mean_path_loss);
% Note that the mean_path_loss in dB specified above is equal to -
10*log10(mean(signal))

% Plot signal in dB, against time;
fs = sample_rate * 10^3; % Sampling rate (Hz)
time_samp = (1:length(signal)) / fs;
plot(time_samp, 10*log10(signal));
xlabel('Time (s)')
ylabel('Channel gain (dB)')
title('Channel gain from Tx to Rx for mobile wireless body area network')

%-----
% References:
% [Miniutti08] D. Miniutti, L. Hanlen, D. Smith, A. Zhang, D. Lewis, D.
% Rodda and B. Gilbert, "Characterisation of large-scale
% fading in BAN channels", October 2008.
% [Weibull1951] Weibull, W. A., "A statistical distribution function of
% wide applicability", Journal of Applied Mechanics, vol. 18,
% pp. 292-297, 1951.

%-----
% Subfunctions (this is where the real work is done)
%-----

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function signal = gen_sig_wmban(vel, car_frequency, sample_rate, time,
mean_path_loss)
% This function creates a received power profile vector for a received
% signal for an on-body mobile wireless BAN.
%
% Return value:
% signal - A power profile (or power envelope) of absolute magnitude
% (not dB) describing receive power as a fraction of transmit power. The
% mean of the signal is determined by the input parameter mean_path_loss.
%
```

```
% Parameters:
% vel          - Speed of movement of body for Rx with respect to
%              scatterers and/or Tx (km/h)
% car_frequency - Carrier frequency (MHz)
% sample_rate  - Rate at which signal envelope is sampled (kHz)
% time        - Time over which signal is generated (s)
% mean_path_loss - Expected mean path loss for the channel (dB)

% Check that parameters lie within suitable ranges
if car_frequency < 400 || car_frequency > 2500
    disp('Carrier frequency should be in range 400 -- 2500 MHz')
    signal = [];
    return
end
if sample_rate < 0.75 || sample_rate > 15
    disp('Sample rate should be in range 0.75 -- 15 kHz')
    signal = [];
    return
end
if vel < 1.5 || vel > 20
    disp('Velocity should be in range 1.5 -- 20 km/h')
    signal = [];
    return
end

%-----
% Set parameters used to generate signal.
%-----

% Maximal Doppler frequency (Hz)
fd = vel / 3.6 * car_frequency * 10^6 / (3 * 10^8);
% Note that this is an approximate target value as fades in the generated
% signal are manipulated later in the code.

% Sample rate (Hz)
fs = sample_rate * 10^3;

% Doppler fading parameter (normalized Doppler spread)
p = fd / fs;

% Total number of samples over which final signal sig is generated
% This may be increased below, depending on two conditions.
nsamp = round(time*fs);

%-----
% Increase number of samples if required
%-----

% Given that samples of the signal's power envelope are discarded when we
% later manipulate the fade depths, we may need to increase the number of
% samples here so we don't "run out" of samples later. If, at the end of
```

```
% manipulating the fade depths there are still samples left over, then the
% signal is truncated to the correct length.

% The two conditions that are used to determine whether we need to increase
% the number of samples were found by extensive empirical testing. These
% conditions are:
% 1. carrier frequency is less than 800 MHz and time specified is less
% than 30s; or
% 2. time specified is less than 20s.

time_change = 0; % Keep track of whether time parameter has been changed
if car_frequency < 800 && time < 35
    time = 35;
    nsamp1 = nsamp;
    nsamp = round(time*fs);
    time_change = 1;
end
if time < 20
    time = 20;
    nsamp1 = nsamp;
    nsamp = round(time*fs);
    time_change = 1;
end

%-----
% Signal generation loop
%-----

% In some very isolated cases (in particular given certain values of
% velocity, carrier frequency, time and sample rate) we may "run out" of
% samples in the signal generation routine (although this is still very
% unlikely). Thus, to avoid an error, we run a while loop until there are
% enough samples (in the great majority of cases there will be only one
% iteration of this loop).
enough_samples_indicator = 0;
while enough_samples_indicator == 0,

    %-----
    % Below we generate an initial Weibull fading power profile with
    % approximate specified Doppler fading parameter (normalized Doppler
    % spread).
    %
    % Order statistics of a Rayleigh power profile with the specified
    % Doppler fading parameter are used to order randomly generated Weibull
    % distributed numbers to obtain a Weibull faded signal.
    %
    % The order of the same length Rayleigh faded signal (corresponding to
    % Rayleigh distribution of specified Doppler parameter) determines the
    % order of the Weibull distributed random numbers to produce a relevant
    % Weibull faded signal profile with approximate Doppler parameter.
    %-----
```

```
% First generate a vector of Weibull distributed random numbers around
% the mean, Al.
% Note that it is necessary to generate more random samples than are
% asked for as we will later discard some samples when manipulating the
% fade depths. The code below uses a multiplier of 1.6 to specify how
% many samples should be generated. This number was chosen by empirical
% testing and is generally large enough to allow discarding of samples
% in the final signal after fade manipulation and as small as possible
% to produce a final signal that most accurately reflects the desired
% statistics.
randnum = rand(1, round(1.6*nsamp)); % Uniformly distributed random
numbers on (0,1)

% The following Weibull parameters are derived from the best overall
% data fit (with mean path loss removed - i.e. power profile normalized
% to mean) from NICTA's measurement campaign.
a = 0.9926; % Weibull shape parameter
b = 0.9832; % Weibull scale parameter

Al = a*(-log(randnum)).^(1/b); % Weibull distributed random numbers
AdB = 10*log10(Al); % Convert to dB

% Exclude those values above and below the mean that didn't occur in
% all 12.7 million measurements as they are unrealistic in the context
% of what received signal powers can be reasonably expected. Note that
% these discards will be very rare and should not affect the Weibull
% distribution in any significant way.
AdB = AdB(AdB>-73 & AdB<21);

% Generate a Rayleigh fading signal power profile using Jakes' model,
% whereby the ordering of Rayleigh fading generated statistics is used
% to create an appropriately ordered Weibull fading signal (based on
% the Rayleigh signal ordering with the same Doppler parameter over the
% length of the Weibull distributed samples). This method for producing
% appropriately ordered Weibull fading follows from the methods to
% produce Nakagami-m fading in [Filho07] and Lognormal fading in
% [Cotton07].
% NB: This is only an initial signal generation; following this we
% will manipulate this signal so that fade depths correspond to those
% from NICTA's measurement campaign.

% Rayleigh_power: a Rayleigh power profile generated using Jakes' model
% with appropriate Doppler fading parameter
Rayleigh_power = abs(jakesm_1_1(length(AdB), p)).^2;

% I is the indexing of the sorted Rayleigh_power (in ascending order)
% with respect to the original generated Rayleigh power profile
[Habs_s, I] = sort(Rayleigh_power);

% Including the mean path loss (in dB) we generate a set of received
```

```

% signal powers (unordered), P (in dB), based on Weibull random numbers
% generated.
P = AdB - mean_path_loss;
A_f = 10.^(P/10); % Absolute values of generated received signal powers

% Sort the initial signal according to Rayleigh ordering
initial_signal = sort(A_f);
initial_signal(I) = initial_signal;
% This initial signal is a Weibull faded signal power profile with
% approximate Doppler parameter (or normalized Doppler spread) p, and
% Weibull shape and scale parameters (a and b), as specified.

%-----
% Manipulate fade depths
%-----

% The initial_signal generated above will, in general, not have the
% desired distribution of fade depths, so the signal must be
% manipulated to achieve the desired distribution. We now call a
% function that will manipulate the initial signal. This function will
% produce a signal with a distribution of fade depths (in dB) that
% approximate those from NICTA's measurement campaign.

signal_dB = manipulate_fade_depths(initial_signal, mean_path_loss);
% Note that the Doppler fading parameter of the signal returned from
% the function below will not exactly match the input initial_signal to
% this function. However, it will still remain a rough approximation of
% the desired doppler fading parameter.

% If we have generated enough samples in the final signal then we can
% end the while loop. Otherwise, we perform another iteration of the
% loop.
if length(signal_dB) >= nsamp
    enough_samples_indicator = 1;
end

end % (while)

% Final generated signal is truncated to total number of desired samples.
% Note that the final generated signal power profile is an absolute
% magnitude (not dB).
signal = 10.^(signal_dB(1:nsamp)/10); % Truncate and convert from dB to
absolute magnitude

% If we changed the effective time to help signal generation, further
% truncate signal to time desired in function call.
if time_change == 1,
    signal = signal(1:nsamp1);
end

```

```

% Finally, we readjust the signal so its mean matches the specified mean
% path loss (mean will have changed slightly due to fade manipulation).
signal_mean_dB_non_fixed = 10*log10(mean(signal));
signal_dB_fin = 10*log10(signal) - (mean_path_loss +
signal_mean_dB_non_fixed);
signal = 10.^(signal_dB_fin/10);

%-----
% References:
% [Filho07] Filho, J.C.S.S, Yacoub, M. D. and Fraidenraich, G., "A Simple
% Accurate Method for Generating Autocorrelated Nakagami-m
% Envelope Sequences", IEEE Communication Letters, vol. 11, no.
% 3, pp. 231-233, Mar 2007.
% [Cotton07] Cotton, S. L. and Scanlon, W. G., "Higher Order Statistics for
% Lognormal Small-Scale Fading in Mobile Radio Channels", IEEE
% Antennas and Wireless Propagation Letters, vol. 6, pp. 540-543,
% 2007.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Hs = jakesm_ismo(N1, p)
% This function generates Rayleigh fading for a single-input single-output
% system according to Jakes' model [Jakes74].
%
% Return value:
% Hs - Complex fade amplitudes
%
% Parameters:
% N1 - Desired number of samples for the channel response
% p - Doppler fading parameter (or normalized Doppler spread by sample
rate)

Ns = 50; % Number of scatterers (chosen to get appropriately 'rich'
scattering in Rayleigh fading)
c = 1 / sqrt(Ns); % Normalization factor

ts = repmat(2*pi*rand(1,Ns), N1, 1);
ft = 2*pi*p*repmat((1:N1)', 1, Ns) .* repmat(cos(2*pi*rand(1,Ns)), N1, 1);
Hs = sum(c*exp(-j*(ft+ts)), 2);

%-----
% Reference:

```

```
% [Jakes74] Jakes, W. C., "Microwave Mobile Communications", Ed. John
%           Wiley, New York, 1974.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function signal_dB = manipulate_fade_depths(initial_signal, mean_path_loss)
% This function manipulates the initial_signal power profile parameter in
% such a way as to produce a signal power profile (in dB) with the desired
% distribution of fade depths.
%
% This is achieved in the following manner:
% * The signal is treated in portions, each portion being a fade followed
%   by a non-fade (a contiguous portion of the signal above the mean
%   power).
% * Iterate through all the signal portions and compare each fade to the
%   desired fade depth statistics. The current signal portion is
%   manipulated depending on this comparison in one of three ways.
%   - Good match to desired statistics: Keep the current signal portion
%     and insert it into the output signal.
%   - Too much attenuation: Remove the parts of the fade that have too
%     much attenuation. The remaining signal portion is then inserted into
%     the output signal.
%   - Bad match to desired statistics: Discard the current signal
%     portion (including the non-fade part of the portion).
% * Due to the unnatural ordering that results from the fade depth
%   manipulation described above, the non-discarded portions of the signal
%   are reordered (though not reordering within portions).
%
% We note that no manipulation of the non-faded portion is necessary.
%
% This method has been found to be statistically accurate from empirical
% testing. Furthermore, the resultant final signal power profile properties
% (and appearance) are a good match to NICTA's measured power profiles.
%
%
% Return value:
%   signal_dB - A signal power profile in dB with appropriate fade depth
%               distribution
%
% Parameters:
%   initial_signal - Signal to be manipulated
%   mean_path_loss - Desired mean path loss for output signal
%
% Notes:
% - The mean of signal_dB is generally within 1dB of the specified mean
%   path loss, due to the manipulation of the fades. That is, it doesn't
%   match mean_path_loss exactly, but the function that calls this one will
```

```

% make an appropriate adjustment to the mean.
% - signal_dB will have a Doppler fading parameter that is a rough
% approximation to that specified in function that calls this function.
% - The length of the vector signal_dB will be shorter than that of the
% initial_signal.

%-----
% Find: 1) the number of fades in the initial signal; and 2) where these
% fades occur.
%-----

% Find the number of fades in the initial signal, by first quantising the
% signal around the mean
quantised = initial_signal / mean(initial_signal) < 1;
crossings_up = find(diff(quantised) == -1); % Index of when signal crosses
above threshold of the mean, suggesting the end of a fade
crossings_down = find(diff(quantised) == 1); % Index of when signal crosses
below threshold of the mean, suggesting the beginning of a fade

% Remove stray crossings at start & end of data
if(length(crossings_up) > 1 && length(crossings_down) > 1)
    if(crossings_up(1) < crossings_down(1))
        crossings_up(1) = [];
    end
    if(crossings_down(end) > crossings_up(end))
        crossings_down(end) = [];
    end
end

% Hence the number of fades in the initial signal is...
nfades = length(crossings_up);

%-----
% Generate reasonable expected fade depths for each fade from a
% distribution that is based upon NICTA's measurement campaign.
%-----

% We generate a vector of appropriately distributed fade magnitudes (in dB)
% over the number of fades found in the signal. The distribution is a
% Weibull distribution that approximates the fade magnitude distribution
% (in dB) found in NICTA's measurement campaign.
% Note that the best fit for the fade magnitudes is actually a Gamma
% distribution (gamma parameters: shape = 0.669, scale = 14.46), but the
% Weibull is a very good approximation.
randnum = rand(1,nfades); % Uniformly distributed random numbers on (0,1)
a = 8.4; % Weibull shape parameter
b = 0.77; % ; Weibull scale parameter
wp = a*(-log(randnum)).^(1/b); % Weibull distributed random numbers
fade_mags = -mean_path_loss - wp; % Place Weibull random numbers around mean
path loss

```

```
% Exclude those fades that didn't occur in all 12.7 million of NICTA's
% measurements as they are unrealistic in the context of what fade depths
% can be reasonably expected. Note that such values will be very rare in
% the generation of fade_mags and their exclusion should not affect the
% Weibull distribution significantly.
fade_mags = fade_mags(fade_mags > -mean_path_loss-73); % Desired fade
magnitudes

% Sort these fades in descending order (for convenience)
fade_mags = sort(fade_mags, 2, 'descend');

% For ease of manipulation we convert our signal into dB
initial_signal_dB = 10*log10(initial_signal);

% Find the index of the start of the first fade
index_start_fade = crossings_down(1);
% - index_start_fade is the index of the start of fade in the signal

% If the initial signal starts in a fade, then we create an empty output
% signal to begin with. Otherwise, we insert the first portion of the
% initial signal (the portion before any fades have occurred) into the
% output signal.
if index_start_fade-1 > 0,
    signal_dB = initial_signal_dB(1:(index_start_fade-1));
else
    signal_dB = [];
end

% Find how many crossings above the threshold of the mean there are.
% This will be used in final signal generation.
total_crossings = length(crossings_up);

% Initialise a counter to keep track of how many signal portions we have
% used so far.
kt = 0;

%-----
% Fade manipulation:
% Apply the previously described method of keeping/clipping/removing faded
% sections of the initial signal.
%-----

% Iterate over each fade & non-fade portion of the initial signal
for j = 1:total_crossings

    % Find the index of the crossing (above the mean) in the initial
    % signal. This index marks the end of the fade.
    index_end_fade = crossings_up(j) - 1;
```

```

% Find the portion of signal that corresponds to this fade
current_fade = initial_signal_dB(index_start_fade:index_end_fade);

% Find the magnitude/depth of the current fade
current_fade_magnitude = min(current_fade);

% Compare the current fade magnitude to the sorted vector of desired
% fade magnitudes. We find the index of the closest desired fade depth
% whose fade is of greater magnitude than the current fade magnitude.
ks = max(find(fade_mags>current_fade_magnitude));

% If such an index exists we manipulate the initial faded signal
% portion appropriately
if isempty(ks)
    % No matching fade depth was found, do not use this portion of the
    % signal.
    ts = 0; % Flag that the current signal portion was not used

else
    fm = fade_mags(ks); % Find the value of given fade depth

    % Remove this fade magnitude from the vector of possible magnitudes
    % (we only use each fade magnitude once).
    fade_mags = fade_mags(fade_mags~=fm);

    % Remove the parts of the current signal portion which have too
    % much attenuation.
    current_fade = current_fade(current_fade>fm);

    % Use the current signal portion?
    if ~isempty(current_fade)
        kt = kt + 1; % Increase counter of signal portions used
        ts = 1; % Flag that the current signal portion was used
    else
        % The current signal portion is empty, can't use it.
        ts = 0; % Flag that the current signal portion was not used
    end
end

%-----
% Collect all the fades together
%-----

% If we have not iterated through all of the fades then insert the
% current signal portion (fade + non-fade) into a cell array of signal
% portions, as appropriate. This cell array will be used to generate
% the final output signal.
if j+1 <= total_crossings

    % Set index for start of next fade
    index_start_fade = crossings_down(j+1);

```

```
% If current signal portion was used...
if ts == 1
    % Take the current fade as well as the next contiguous non-fade
    % portion of the signal and insert it into the cell array.
    signal_portion{kt} = [current_fade,
initial_signal_dB((index_end_fade+1):(index_start_fade-1))];
end

else
    % There is no non-fade portion following the current fade, so we
    % just add the fade to the cell array.
    if ts == 1,
        signal_portion{kt} = current_fade;
    end
end

% If all the generated fade depths have been used, exit the 'for' loop
if isempty(fade_mags),
    break;
end

end %(for)

%-----
% Jumble the signal
%-----

% We finally jumble up each signal portion that we stored in the cell array
% (note: not jumbling within portions). This is necessary due to the
% unnatural ordering of fades that occurs from our method of matching fade
% depths.

Rp = randperm(kt);

for kl = 1:kt,
    signal_dB = [signal_dB,signal_portion{Rp(kl)}];
end
```

Published with MATLAB® 7.4