```
%Objective: read in an input signal and identify the positions and
 types
%of vowels in the signal.

%First, read in a voice signal at very high sampling frequency, here
 set to
%44.1kHz. That way, you can have a lot of information to process
 between
%any times t1 and t2, as opposed to if you had a small sampling
 frequency.

%The parameter y is a very long voice signal of length 44100*n , where
 n is
%the number of seconds of speech.

%Because of inevitable noise infiltrating our data, it is very
%difficult to establish the points in the signal where a sound
 definitively
%starts and where it stops. So, we adopt a different approach and look
 at
%the problem from a spectrogram-analysis point of view.

%Assumptions:
%1) The user will speak at a relatively constant amplitude. Note that
 this
%   is a little easier said than done, because words like "see" are
%   naturally much quieter than "hat."
%2) The user will speak slowly and clearly so that we can look for
%   consistency in determining the vowels said.

%"Recipe" for success:
%1) Establish a set of variables that store theoretical vowel
 formants.
%2) Record an audio signal of n seconds from the user.
%3) Preprocess the data by detrending with a low-order polynomial.
%4) Split the data into non-overlapping chunks of 4000 samples.
%5) Determine the transfer function of the vocal tract associated with
%   the current chunk using an ar model.
%6) Determine the peaks of the transfer function (the formants) and
 match
%   with the closest formant, filtering by a least means-squared
 matched
%   filter.
%7) If amplitude of signal exceeds 0.1 max amplitude of overall
 signal, we
%   assume it's potentially a vowel. Put its formants onto the "stack"
 of
%   recent_formant_pairs.
%8) If the last four formant pairs have been consistent (the same
 value),
%   then we will assume the vowel estimation has worked. Add the
 formant
```

```
%   pair to the "stack" of vowels identified in
 track_times_and_formants.
%9) Now we have processed our guesses for what vowels were said when.
 We
%   are going to have repeats, so run through a for-loop to clean up
 the
%   track_times_and_formants vector into a new, workable vector called
%   track_begin_end_formants.
%10)Output data and guesses.

%Limitations:
%1) Requires very clear enunciation. Difficult to establish because we
%   naturally change the pitch of our words as we start and end them.
 Example:
%   the end of the vowel in "rug" sounds remarkably similar to "hat"
%   because of the way you shape your vocal tract as you close your
 mouth.
%2) User has to say each vowel for a moderate duration of time. Too
 long,
%   and a wavering voice would affect success of the program. Too
 short,
%   and not enough data to assign formants.

% Establishes initial values and constants.

[giant_matrix, output_matrix, recent_formant_pairs,
 track_times_and_formants, track_counter] = initialize_all_data();

disp('Now the program will analyze your speech signal, section by
 section')
disp('It will split the signal into *non*overlapping chunks to
 determine whether')
disp('a vowel is present at the corresponding locations in time.')

for t = 1:4000:length(y)-8001

    [h,w, isNoise] = determine_frequency_response(y, t, Fs);

    % time to determine formants
    % first determine indices of local maxima of transfer function
    [pks, locs] = findpeaks(abs(h));
    [~, locs_ordered] = sort(pks, 'descend');

    formants = 1:length(pks);
    %this loop assigns formants to the locations in w corresponding to
 "peaks"
    % using relation discrete_freq = analog_freq * Sample_rate
    for i = 1:length(pks)
        analog_freq = w(locs(locs_ordered(i)))*Fs/(2*pi);
        if (analog_freq > 140)  % tries to filter out any ultra-low
 frequency sound messups
                formants(i) = w(locs(locs_ordered(i)))*Fs/(2*pi);
        end
    end
```

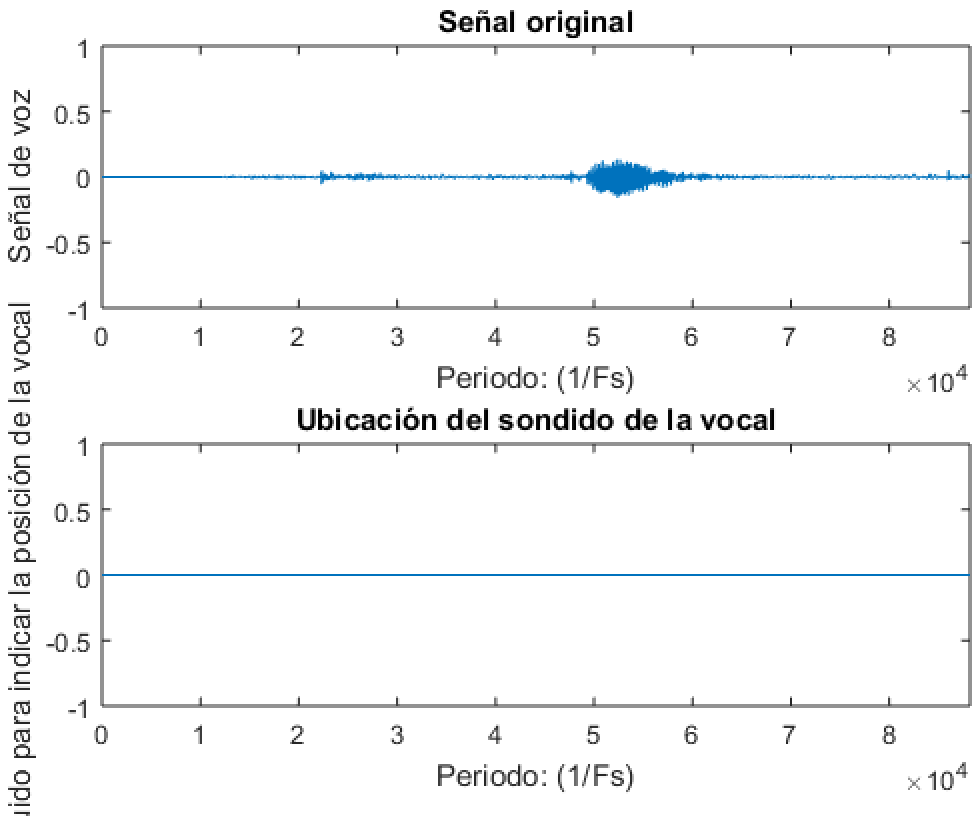
```
   recent_formant_pairs = [recent_formant_pairs(2,:);
recent_formant_pairs(3,:) ;  -1 0];


   if isNoise~=1
       %disp('THIS CONDITION WAS SATISFIED');
       formants_identified = (sort(formants(1:2)));
       %disp(['At time t  = ' num2str(t/Fs) ', the signal chunk
has'])
       %disp(['formants at ' num2str(formants_identified)])
       vector_dist = 1:6;
       for k = 1:6
          weights = [1 1]; %weight the first formant three times as
much as the second formant
          vector_dist(k) = norm(weights.*(giant_matrix(k,:) -
formants_identified));
       end

       %min_index identifies the vowel location in giant_matrix by
using a matched
       %filter dependent on the least-squares difference of the
experimental and
       %theoretical vowel spectra.
       if (max(formants_identified) < 4200)
           [~, min_index] = min(vector_dist);
       else
           min_index = 0;
         %  disp('The program rejects this set of formants because
the value of the ')
         %  disp('second formant is too high!')
       end

       %print t/Fs.0;
       %print '  ';
       %print output_matrix(min_index,:)
       if (min_index ~= 0)
           recent_formant_pairs = [recent_formant_pairs(1,:);
recent_formant_pairs(2,:) ; giant_matrix(min_index, :)];
           if (rank(recent_formant_pairs) == 1) % i.e. all the rows
are linearly dependent, implying they're the same in this case
              % disp('The program has determined that the last four
measurements have been')
              % disp('consistent. That means we shall consider this
set of measurements')
              % disp('as a vowel that appears in your signal!')
               track_times_and_formants(track_counter, :) = [t/Fs
giant_matrix(min_index, :)];
               track_counter = track_counter + 1;
               % If we see that the program identifies the same vowel
               % for at least 4 loops, then we can be confident that
some
               % vowel is being said.
```

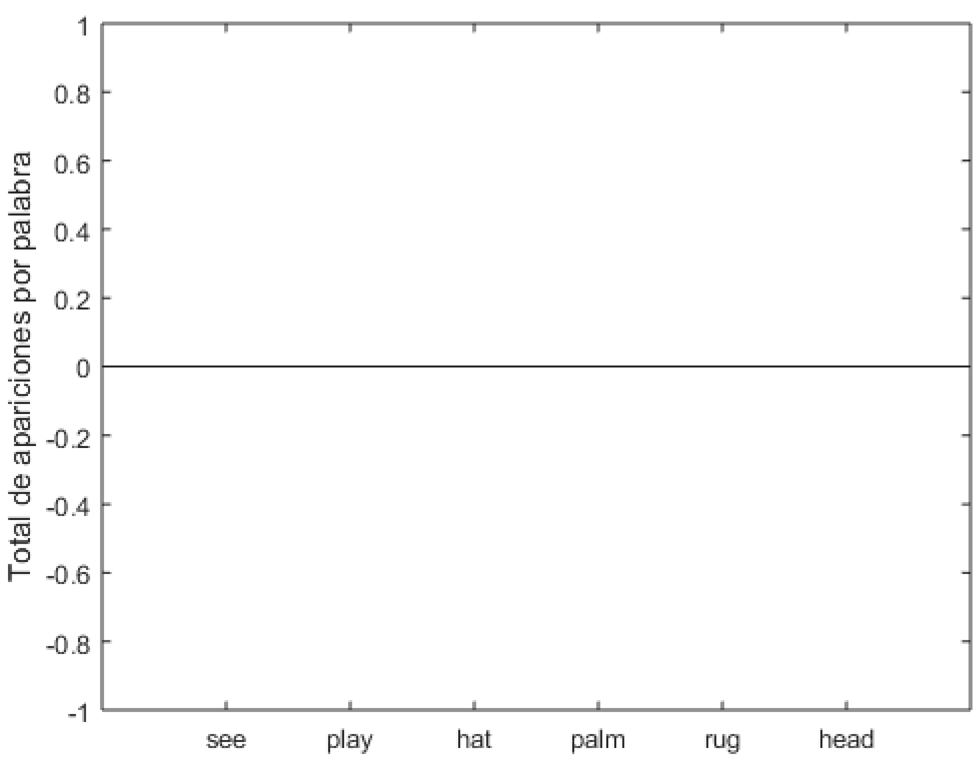
```
               %disp(['According to the frequency spectrum of your
 input signal, the vowel you said at time t = ' t/44100.0  'seconds is
 the central vowel of the word ' output_matrix(min_index,:)]);
            end
        end
    end
end

%Now we have a matrix track_times_and_formants which has done its job.
%However, we have to do some post-processing, because we anticipate
 many
%repetitions in the matrix. Aka if you say the word "see" for a long
 time,
%there will be multiple rows in track_times_and_formants corresponding
 to
%your voice signal.

%Create a track_begin_end_formants matrix that stores the beginning
 and
%ending time of a vowel as well as the appropriate formants.

%first column of track_begin_end_formants is t1, the start time of the
%vowel
%second column is t2, the end time of the vowel
%3rd and 4th columns store the corresponding formants at those times

[track_begin_end_formants, row_track] =
 cleanup_formant_data(track_times_and_formants);

display_matches(track_begin_end_formants, row_track, giant_matrix,
 output_matrix, y)
```

Now the program will analyze your speech signal, section by section
*It will split the signal into *non*overlapping chunks to determine whether*
a vowel is present at the corresponding locations in time.
Warning: Polynomial is badly conditioned. Add points with distinct X values,
reduce the degree of the polynomial, or try centering and scaling as described
in HELP POLYFIT.
Warning: Polynomial is badly conditioned. Add points with distinct X values,
reduce the degree of the polynomial, or try centering and scaling as described
in HELP POLYFIT.
Warning: Polynomial is badly conditioned. Add points with distinct X values,
reduce the degree of the polynomial, or try centering and scaling as described
in HELP POLYFIT.
Warning: Polynomial is badly conditioned. Add points with distinct X values,

```
reduce the degree of the polynomial, or try centering and scaling as
 described
in HELP POLYFIT.
Warning: Polynomial is badly conditioned. Add points with distinct X
 values,
reduce the degree of the polynomial, or try centering and scaling as
 described
in HELP POLYFIT.
Warning: Polynomial is badly conditioned. Add points with distinct X
 values,
reduce the degree of the polynomial, or try centering and scaling as
 described
in HELP POLYFIT.
Warning: Polynomial is badly conditioned. Add points with distinct X
 values,
reduce the degree of the polynomial, or try centering and scaling as
 described
in HELP POLYFIT.
Warning: Polynomial is badly conditioned. Add points with distinct X
 values,
reduce the degree of the polynomial, or try centering and scaling as
 described
in HELP POLYFIT.
Warning: Polynomial is badly conditioned. Add points with distinct X
 values,
reduce the degree of the polynomial, or try centering and scaling as
 described
in HELP POLYFIT.
Warning: Polynomial is badly conditioned. Add points with distinct X
 values,
reduce the degree of the polynomial, or try centering and scaling as
 described
in HELP POLYFIT.
Warning: Polynomial is badly conditioned. Add points with distinct X
 values,
reduce the degree of the polynomial, or try centering and scaling as
 described
in HELP POLYFIT.
Warning: Polynomial is badly conditioned. Add points with distinct X
 values,
reduce the degree of the polynomial, or try centering and scaling as
 described
in HELP POLYFIT.
Warning: Polynomial is badly conditioned. Add points with distinct X
 values,
reduce the degree of the polynomial, or try centering and scaling as
 described
in HELP POLYFIT.
Warning: Polynomial is badly conditioned. Add points with distinct X
 values,
reduce the degree of the polynomial, or try centering and scaling as
 described
in HELP POLYFIT.
```

```
Warning: Polynomial is badly conditioned. Add points with distinct X
 values,
reduce the degree of the polynomial, or try centering and scaling as
 described
in HELP POLYFIT.
Warning: Polynomial is badly conditioned. Add points with distinct X
 values,
reduce the degree of the polynomial, or try centering and scaling as
 described
in HELP POLYFIT.
Warning: Polynomial is badly conditioned. Add points with distinct X
 values,
reduce the degree of the polynomial, or try centering and scaling as
 described
in HELP POLYFIT.
Warning: Polynomial is badly conditioned. Add points with distinct X
 values,
reduce the degree of the polynomial, or try centering and scaling as
 described
in HELP POLYFIT.
Warning: Polynomial is badly conditioned. Add points with distinct X
 values,
reduce the degree of the polynomial, or try centering and scaling as
 described
in HELP POLYFIT.
Warning: Polynomial is badly conditioned. Add points with distinct X
 values,
reduce the degree of the polynomial, or try centering and scaling as
 described
in HELP POLYFIT.
Warning: Polynomial is badly conditioned. Add points with distinct X
 values,
reduce the degree of the polynomial, or try centering and scaling as
 described
in HELP POLYFIT.
The program has finished analyzing your input signal.

It has determined the following information about the words you said.
According to the frequency spectrum of your input signal:
```


Señal original
Señal de voz
Periodo: (1/Fs)
×10^4
Ubicación del sondido de la vocal
ido para indicar la posición de la vocal
Periodo: (1/Fs)
×10^4


Total de apariciones por palabra
see
play
hat
palm
rug
head

Published with MATLAB® R2015a