Lecture 20: Fourier Transform and Speech Recognition

University of Southern California

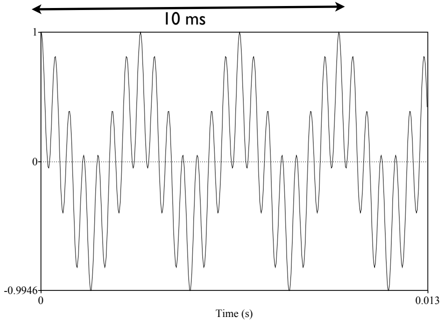
Linguistics 285

USC Linguistics

November 8, 2015

Fourier Analysis (Fourier Transform)

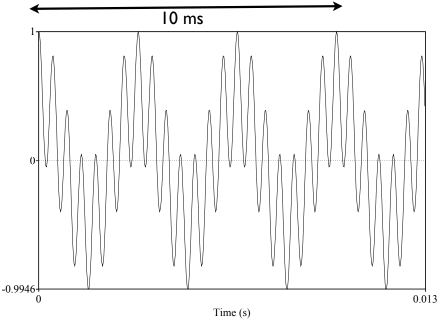
- How do we find the frequencies that compose a signal?
- In a simple case, we can find them by careful observation of a waveform

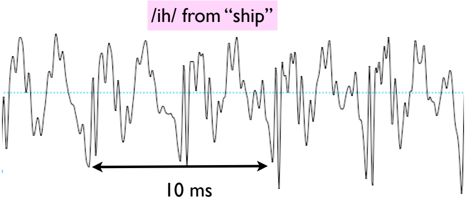


- But not in a more complex case

Fourier Analysis (Fourier Transform)

- How do we find the frequencies that compose a signal?
- Observation of waveform in simple, artificial case, but not in complex, real case





Fourier Analysis (Fourier Transform)

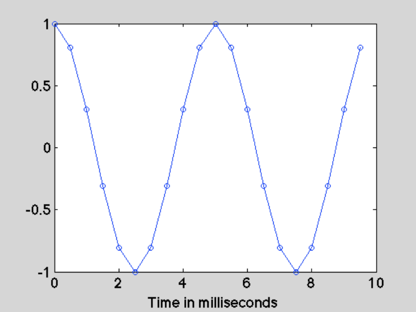
- **Fourier Analysis** is the technique that does this.
- Produces something called the **Fourier Transform**, which contains the information in the spectrum.
- But how does it work?
- By using the inner product!
- Take the inner product of the signal (waveform) with pure tones of all possible frequencies.
- The size of the IP tells us how much that signal is similar to each tone.
- That quantitative similarity is the relative amplitude of the frequency in complex signal.

Fourier Analysis (Fourier Transform)

- But how can we take the inner product of a signal with tones?
- Signals are just sequences of numbers, vectors!
- In this example, a new number (called a sample) every .5 ms

1.0000
0.8090
0.3090
-0.3090
-0.8090
-1.0000
-0.8090
-0.3090
0.3090
0.8090
1.0000
0.8090
0.3090
-0.3090
-0.8090
-1.0000
-0.8090
-0.3090
0.3090
0.8090

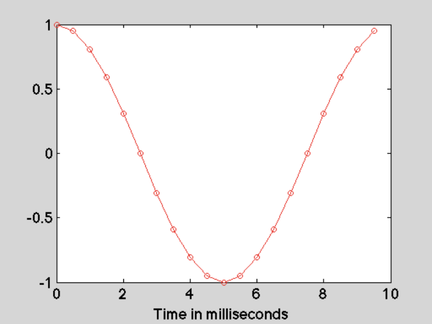
200 Hz tone



Fourier Analysis (Fourier Transform)

1.0000
0.9511
0.8090
0.5878
0.3090
0.0000
-0.3090
-0.5878
-0.8090
-0.9511
-1.0000
-0.9511
-0.8090
-0.5878
-0.3090
-0.0000
0.3090
0.5878
0.8090
0.9511

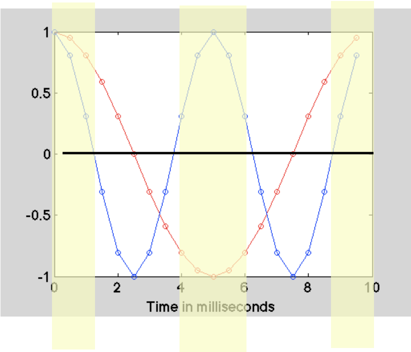
100 Hz tone



Fourier Analysis via inner product

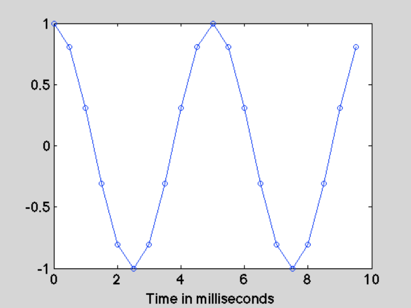
- Let's treat the 200 Hz tone as our signal whose spectrum we want to measure.
- Take inner product with 100 Hz, 200 Hz...500 Hz, etc.

- There are times where the product is large where both vectors are positive, but each one is exactly canceled by a time when the vectors have opposite signs, so the product is negative. The sum (IP) is zero.



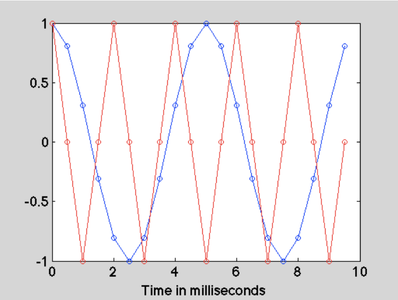
s200	s100	s200.*s100
1.0000	1.0000	1.0000
0.8090	0.9511	0.7694
0.3090	0.8090	0.2500
-0.3090	0.5878	-0.1816
-0.8090	0.3090	-0.2500
-1.0000	0.0000	-0.0000
-0.8090	-0.3090	0.2500
-0.3090	-0.5878	0.1816
0.3090	-0.8090	-0.2500
0.8090	-0.9511	-0.7694
1.0000	-1.0000	-1.0000
0.8090	-0.9511	-0.7694
0.3090	-0.8090	-0.2500
-0.3090	-0.5878	0.1816
-0.8090	-0.3090	0.2500
-1.0000	-0.0000	0.0000
-0.8090	0.3090	-0.2500
-0.3090	0.5878	-0.1816
0.3090	0.8090	0.2500
0.8090	0.9511	0.7694

Sum = 0



s200	s200	s200.*s200
1.0000	1.0000	1.0000
0.8090	0.8090	0.6545
0.3090	0.3090	0.0955
-0.3090	-0.3090	0.0955
-0.8090	-0.8090	0.6545
-1.0000	-1.0000	1.0000
-0.8090	-0.8090	0.6545
-0.3090	-0.3090	0.0955
0.3090	0.3090	0.0955
0.8090	0.8090	0.6545
1.0000	1.0000	1.0000
0.8090	0.8090	0.6545
0.3090	0.3090	0.0955
-0.3090	-0.3090	0.0955
-0.8090	-0.8090	0.6545
-1.0000	-1.0000	1.0000
-0.8090	-0.8090	0.6545
-0.3090	-0.3090	0.0955
0.3090	0.3090	0.0955
0.8090	0.8090	0.6545

Sum = 10

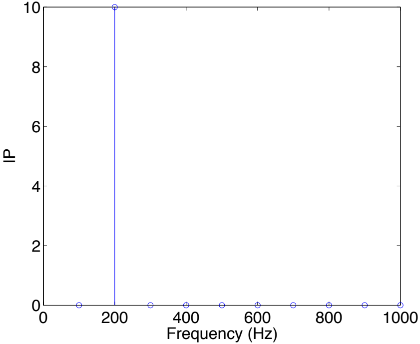


s200	s500	s200.*s500
1.0000	1.0000	1.0000
0.8090	0.0000	0.0000
0.3090	-1.0000	-0.3090
-0.3090	-0.0000	0.0000
-0.8090	1.0000	-0.8090
-1.0000	0.0000	-0.0000
-0.8090	-1.0000	0.8090
-0.3090	-0.0000	0.0000
0.3090	1.0000	0.3090
0.8090	0.0000	0.0000
1.0000	-1.0000	-1.0000
0.8090	-0.0000	-0.0000
0.3090	1.0000	0.3090
-0.3090	-0.0000	0.0000
-0.8090	-1.0000	0.8090
-1.0000	-0.0000	0.0000
-0.8090	1.0000	-0.8090
-0.3090	-0.0000	0.0000
0.3090	-1.0000	-0.3090
0.8090	-0.0000	-0.0000

Sum = 0

Fourier Analysis via inner product

- Other frequencies will also give 0 as the IP, so this our resulting spectrum:



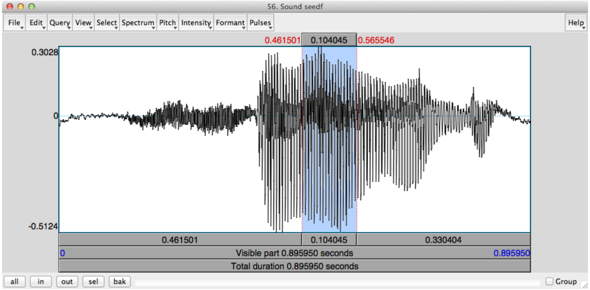
- This will also work with more complex signals!

Example recognizer: Match unknown (test) vowels from one speaker to template vowels from another speaker

1. Select a short waveform chunk from each template and test vowel.
2. Save **waveform** chunks to file and import into a matrix in Matlab
3. Derive the **spectrum** of each vowel by using the Fourier Transform: take the inner product of the vowel waveform with the waveforms of a series of tones with different frequencies.
4. Compare the **similarity** of each test vowel to each template vowel by taking the inner product of the spectrum of the test vowel with each template vowel:
   - **Recognition:** The largest IP value obtained for a given test vowel is the vowel that our recognizer selects as the one spoken.

unknown (test) vowels and template vowels

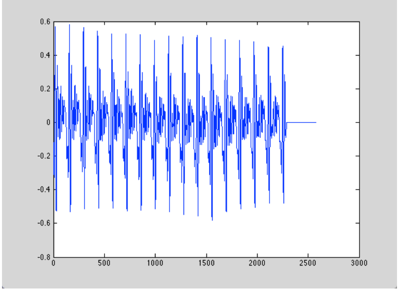
- words from a data base: "seed, said, sod, sud" produced by a male and a female speaker.
- Female speaker will be template, male will be unknown test.
- open waveforms in Praat and select chunk of about 100ms from vowel



- save chunk in its own .wav file.

import template and test vowels into Matlab as vectors

```
% get_vowels

iyf = wavread('iyf');
ehf = wavread('ehf');
aaf = wavread('aaf');
ahf = wavread('ahf');


iym = wavread('iym');
ehm = wavread('ehm');
aam = wavread('aam');
ahm = wavread('ahm');
```



- Vowels can be played in Matlab: `soundsc(aaf, 20000)`
- or plotted: `plot (aaf)`

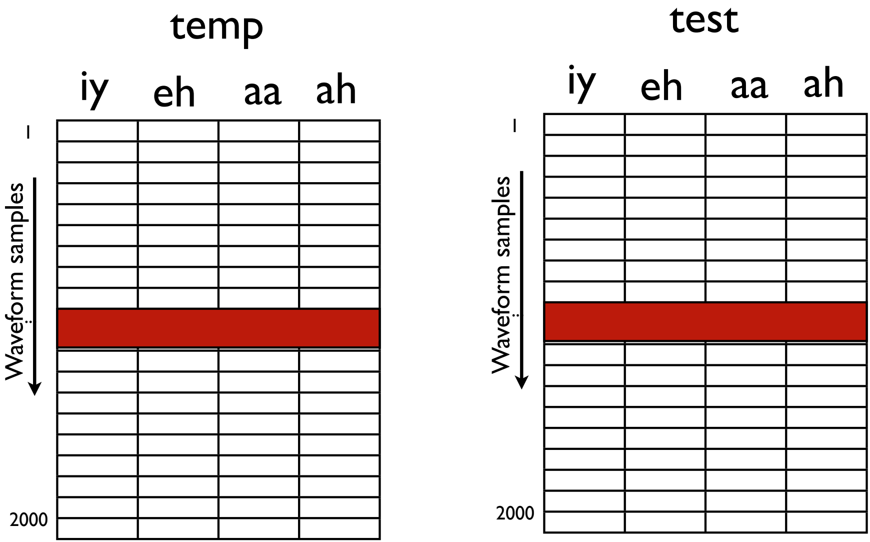
Make matrix with template vowels and with test vowels

```
%make_template

clear temp

temp(:,1) = iyf(1:2000)';
temp(:,2) = ehf(1:2000)';
temp(:,3) = aaf(1:2000)';
temp(:,4) = ahf(1:2000)';
```

```
%make_test

clear test

test(:,1) = iym(1:2000)';
test(:,2) = ehm(1:2000)';
test(:,3) = aam(1:2000)';
test(:,4) = ahm(1:2000)';
```

- Use only 2000 samples as our frequencies have 2000 samples.

Matrix with template vowels and with test vowels



Perform Fourier Transform for each vowel

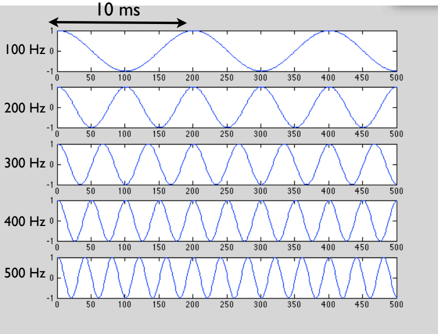
- Take the inner product of each vowel waveform (vector) with the waveform of tones from 100 Hz to 3000 Hz, in steps of 100 Hz.
- The resulting vector of inner products for each of the 30 frequencies is the spectrum.
- The tone waveforms are stored in a matrix called **freqs**.
- Each column is a different frequency. The rows contain the successive time samples.

```
>>size (freqs)
ans =
2000 30
```

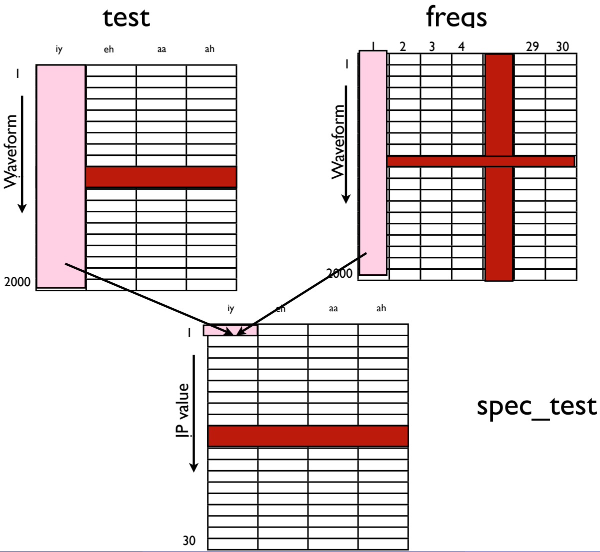
Perform Fourier Transform for each vowel

- Plot the waveforms of the first five frequencies in different panels of a figure.
- `subplot (m,n,p)` plots in an arrangement of *m* rows and *n* columns of plots. *p* is the current one will be plotted, numbered from top to bottom.

```
%plot_freqs

load freqs

for i = 1:5
    subplot (5,1,i), plot (freqs(:,i))
    xlim([0 500]);
end
```



Find IP of each vowel with each frequency



Nested loops

- In a loop, we can change one variable, the index.
- Suppose we want to change two variables, and repeat the computation with all combinations of the values of the two variables.
- We can put one loop inside another, so the inner loop goes through all of its values for the first value of the outer loop.
- Then the outer loop value changes and we go through all the values of the inner loop again.

Nested loops

- Example: Suppose type out the values of this matrix (X), one cell at a time

  X =

  1 7 2

  3 2 9

  7 0 1
- Start with the first row and type out the values from left to right (columns)
- Then go to the next row
- Code that will do this:

```
X = [ 1 7 2; 3 2 9; 7 0 1]

for irow = 1:3
    irow
    for icol = 1:3
        icol
        X(irow, icol)
        pause
    end
end
```

Perform Fourier Transform for each vowel

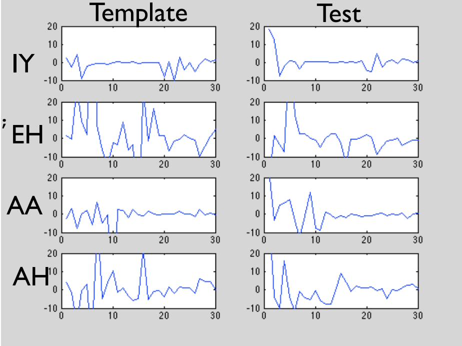
- Nested loop:
- Calculate spectrum of the vowels in the four columns of the **temp** and **test** matrices, one vowel per pass through the loop.
- For each vowel, loop through the 30 frequencies and calculated the IP of the vowel with each frequency.
- Store the results in spec_test and spec_temp matrices, one vowel in each column and 30 frequencies in the rows.

```
% do_ft

for v = 1:4
    for f = 1:30
        spec_temp(f,v) = sum(freqs(:,f).*(temp(:,v)-mean(temp(:,v))));
    end
end

for v = 1:4
    for f = 1:30
        spec_test(f,v) = sum(freqs(:,f).*(test(:,v)-mean(test(:,v))));
    end
end
```

Perform Fourier Transform for each vowel

- Plot the resulting spectra

```
%plot_spec

for i = 1:4
    subplot (4,2,2*i-1), plot(spec_temp(:,i));
    subplot (4,2,2*i-1), ylim([-10 20]);
    subplot (4,2,2*i), plot(spec_test(:,i));
    subplot (4,2,2*i),ylim([-10 20]);
end
```



Compare each unknown vowels to all the templates

- Go thru each test vowel in a loop.
- For each test vowel, go through each template vowel in a loop and take the IP.
- Save the IP in a matrix in which the rows correspond to the test vowels and the columns the template vowels.

```
%do_test

for j = 1:4
    for i=1:4
        IP(j,i) = sum((spec_test(:,j)-mean(spec_test(:,j))).*(spec_temp(:,i)-mean(spec_temp(:,i))));
    end
end
```

```
 >>IP

 IP =

    1.0e+03 *

     0.0023   -0.2601    0.0546    0.0495
     0.0855    1.1562    0.1832   -0.5468
    -0.0204   -0.4350    0.2976   -0.3989
    -0.0648   -1.5917    0.0845    0.7764
```

Results

- How do we interpret the IP matrix?
- Each row has the results for one test vowel, comparing it to all template vowels, across the columns

Template (columns) / Test (rows)

	IY	EH	AA	AH
IY	0.0023	-0.2601	0.0546	0.0495
EH	0.0855	1.1562	0.1832	-0.5468
AA	-0.0204	-0.4350	0.2976	-0.3989
AH	-0.0648	-1.5917	0.0845	0.7764

- The largest IP for each test vowel is the vowel category our system has recognized for that vowel
- If our recognizer worked, the diagonal cells of the matrix should be larger than the off-diagonal cells.
- 3 vowels were correctly recognized. One was not (/IY/).