Lecture 16: Discrete Fourier Transform, Spherical Harmonics

COMPSCI/MATH 290-04

Chris Tralie, Duke University

3/8/2016

Announcements

- Mini Assignment 3 due Sunday 3/13 11:55PM
- Group Assignment 2 Out around Friday/Saturday, due Monday 3/28
- Final projects assigned, first milestone due Wednesday 4/6
- Midterm Thursday

Table of Contents

Numpy: More Broadcasting

```
import numpy as np
import matplotlib.pyplot as plt
X = np.arange(4)
Y = np.arange(6)
Z = X[:, None] + Y[None, :]
print Z
```

Squared Euclidean Distances in Matrix Form

Notice that

$$||\vec{a} - \vec{b}||^2 = (\vec{a} - \vec{b}) \cdot (\vec{a} - \vec{b})$$

$$||\vec{a} - \vec{b}||^2 = \vec{a} \cdot \vec{a} + \vec{b} \cdot \vec{b} - 2\vec{a} \cdot \vec{b}$$

Squared Euclidean Distances in Matrix Form

Notice that

$$||\vec{a} - \vec{b}||^2 = (\vec{a} - \vec{b}) \cdot (\vec{a} - \vec{b})$$

$$||\vec{a} - \vec{b}||^2 = \vec{a} \cdot \vec{a} + \vec{b} \cdot \vec{b} - 2\vec{a} \cdot \vec{b}$$

Given points clouds X and Y expressed as $2 \times M$ and $2 \times N$ matrices, respectively, write code to compute an $M \times N$ matrix D so that

$$D[i, j] = ||X[:, i] - Y[:, j]||^2$$

Without using any for loops! Can use for ranking with Euclidean distance or D2 shape histograms, for example

Brute Force Nearest Neighbors

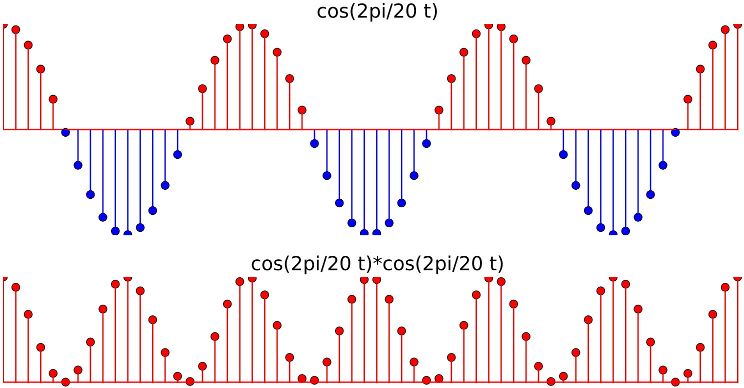
```
import numpy as np
import matplotlib.pyplot as plt
t = np.linspace(0, 2*np.pi, 100)
X = np.zeros((2, len(t)))
X[0, :] = t
X[1, :] = np.cos(t)
Y = np.zeros((2, len(t)))
Y[0, :] = t
Y[1, :] = np.sin(t**1.2)
##FILL THIS IN TO COMPUTE DISTANCE MATRIX D
idx = np.argmin(D, 1) #Find index of closest point in Y
    to point in X
plt.plot(X[0, :], X[1, :], '.')
plt.hold(True)
plt.plot(Y[0, :], Y[1, :], '.', color = 'red')
for i in range(len(idx)):
    plt.plot([X[0, i], Y[0, idx[i]]], [X[1, i], Y[1, idx
        [i]]], 'b')
plt.axes().set_aspect('equal', 'datalim'); plt.show()
```

Mini Assignment 3 API

```
def getCentroid(PC):
    return np.zeros((3, 1)) #Dummy value

def getCorrespondences(X, Y, Cx, Cy, Rx):
    return np.zeros(X.shape[1], dtype=np.int64) #dummy
        value

def getProcrustesAlignment(X, Y, idx):
    return (Cx, Cy, R)

#what the ICP algorithm did
def doICP(X, Y, MaxIters):
    CxList = [np.zeros((3, 1))]
    CyList = [np.zeros((3, 1))]
    RxList = [np.eye(3)]
    #TODO: Fill the rest of this in
    return (CxList, CyList, RxList)
```

Table of Contents

Discrete Cosine Integration



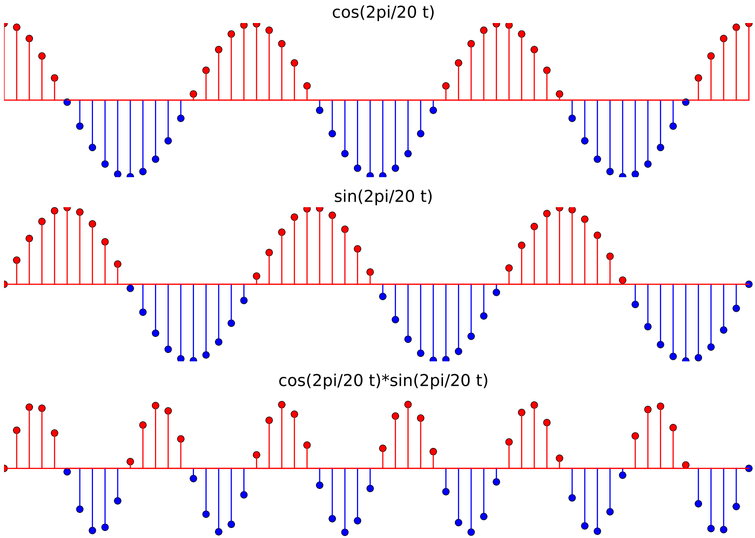
Discrete Cosine Integration

What is

$$\sum_{t=1}^{N=kT} \cos^2(\frac{2\pi t}{T})$$

?
Note that $\cos^2(A) = \frac{1+\cos(2A)}{2}$

Discrete Cosine Integration

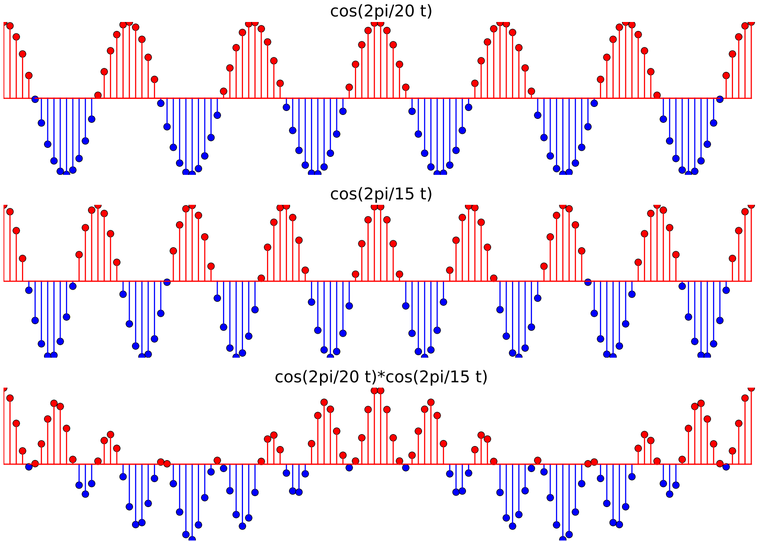


Discrete Cosine Integration

Why is $\sum_{t=1}^{N=kT} \cos(\frac{2\pi t}{T}) \sin(\frac{2\pi t}{T})$ zero?
Note that

$$\cos(A)\sin(A) = \frac{1}{2}\sin(2A)$$

Discrete Cosine Integration



Discrete Cosine Integration

Why does the product of two different cosines integrate to zero?

$$\cos(A + B) = \cos(A)\cos(B) - \sin(A)\sin(B)$$

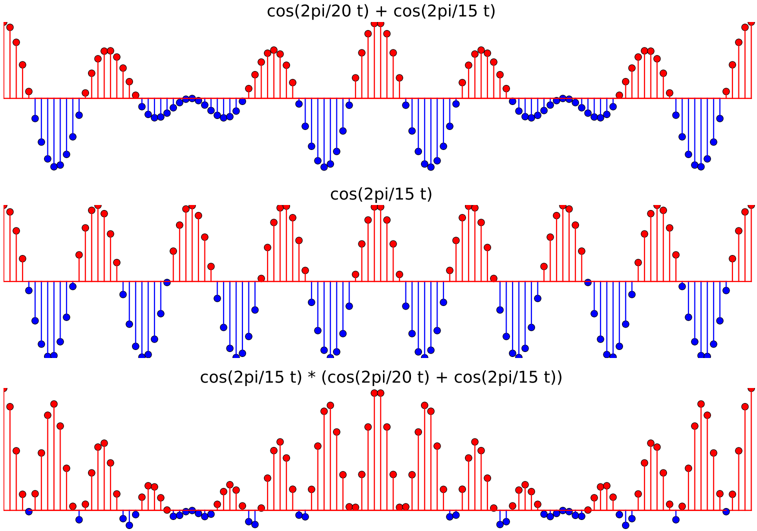
Discrete Cosine Integration

Why does the product of two different cosines integrate to zero?

$$\cos(A+B) = \cos(A)\cos(B) - \sin(A)\sin(B)$$

$$\text{Hint:}\,\cos(A+B) + \cos(A-B) = ?$$

Discrete Cosine Integration



Discrete Cosine Integration

Dot product interpretation

$$\vec{s_1} = \left[1, \cos\left(\frac{2\pi}{T_1}\right), \cos\left(\frac{2\pi 2}{T_1}\right) \ldots, \cos\left(\frac{2\pi N}{T_1}\right)\right]$$

$$\vec{s_2} = \left[1, \cos\left(\frac{2\pi}{T_2}\right), \cos\left(\frac{2\pi 2}{T_2}\right) \ldots, \cos\left(\frac{2\pi N}{T_2}\right)\right]$$

Dot product is linear!

$$\vec{s_2} \cdot (a\vec{s_1} + b\vec{s_2}) = a\vec{s_1} \cdot \vec{s_2} + b\vec{s_2} \cdot \vec{s_2}$$

Can use this type of dot product / integration to detect / test for the presence of different frequencies

Cosine Phase Separation

The general form of a sinusoid is $A\cos(\omega t + \phi)$

$$A\cos(\omega t + \phi) = A\cos(\phi)\cos(\omega t) - A\sin(\phi)\sin(\omega t)$$

Cosine Phase Separation

The general form of a sinusoid is $A\cos(\omega t + \phi)$

$$A\cos(\omega t + \phi) = A\cos(\phi)\cos(\omega t) - A\sin(\phi)\sin(\omega t)$$

$$\cos(\omega t)(A\cos(\omega t+\phi)) = A\cos(\phi)\cos^2(\omega t) - A\cos(\omega t)\sin(\phi)\sin(\omega t)$$

Cosine Phase Separation

The general form of a sinusoid is $A\cos(\omega t + \phi)$

$$A\cos(\omega t + \phi) = A\cos(\phi)\cos(\omega t) - A\sin(\phi)\sin(\omega t)$$

$$\cos(\omega t)(A\cos(\omega t+\phi)) = A\cos(\phi)\cos^2(\omega t) - A\cos(\omega t)\sin(\phi)\sin(\omega t)$$

- The “cosine component” of the sinusoid is $A\cos(\phi)$
- The “sinusoid component” of the sinusoid is $A\sin(\phi)$

Cosine Phase Separation

The general form of a sinusoid is $A\cos(\omega t + \phi)$

$$A\cos(\omega t + \phi) = A\cos(\phi)\cos(\omega t) - A\sin(\phi)\sin(\omega t)$$

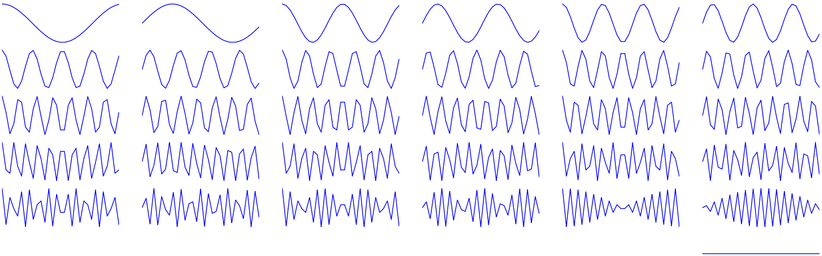
$$\cos(\omega t)(A\cos(\omega t+\phi)) = A\cos(\phi)\cos^2(\omega t) - A\cos(\omega t)\sin(\phi)\sin(\omega t)$$

- The “cosine component” of the sinusoid is $A\cos(\phi)$
- The “sinusoid component” of the sinusoid is $A\sin(\phi)$
- The *magnitude* A of this sinusoid is $\sqrt{(A\cos(\phi))^2 + (A\sin(\phi))^2} = A$ (sanity check)

Sinusoidal Coordinates Basis

For a signal of length N (let N be odd for the moment), define the following matrix

$$F = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \cos(\frac{2\pi}{N}) & \cos(2\frac{2\pi}{N}) & \dots & \cos((N-1)\frac{2\pi}{N}) \\ 0 & \sin(\frac{2\pi}{N}) & \sin(2\frac{2\pi}{N}) & \dots & \sin((N-1)\frac{2\pi}{N}) \\ 1 & \cos(\frac{4\pi}{N}) & \cos(2\frac{4\pi}{N}) & \dots & \cos((N-1)\frac{4\pi}{N}) \\ 0 & \sin(\frac{4\pi}{N}) & \sin(2\frac{4\pi}{N}) & \dots & \sin((N-1)\frac{4\pi}{N}) \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & \cos(\frac{((N-1)\pi}{N}) & \cos(2\frac{(N-1)\pi}{N}) & \dots & \cos((N-1)\frac{(N-1)\pi}{N}) \\ 0 & \sin(\frac{(N-1)\pi}{N}) & \sin(2\frac{(N-1)\pi}{N}) & \dots & \sin((N-1)\frac{(N-1)\pi}{N}) \end{bmatrix}$$

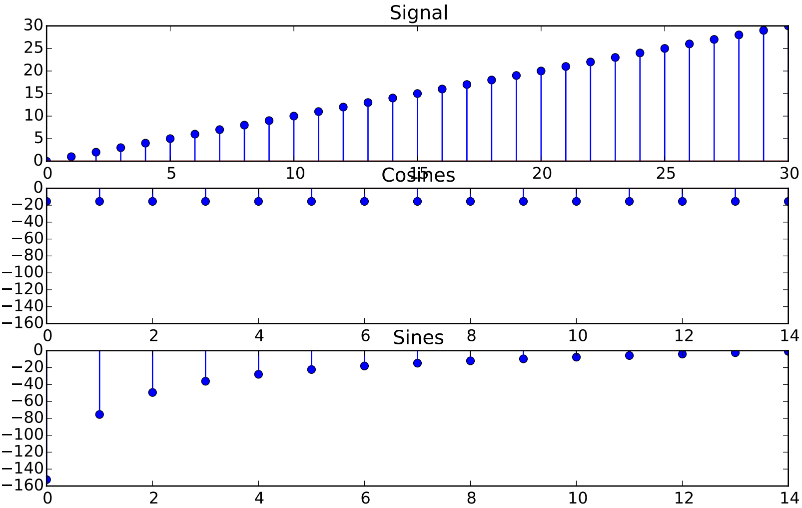


Each row is perpendicular to each other row. This means that

$$Fx$$

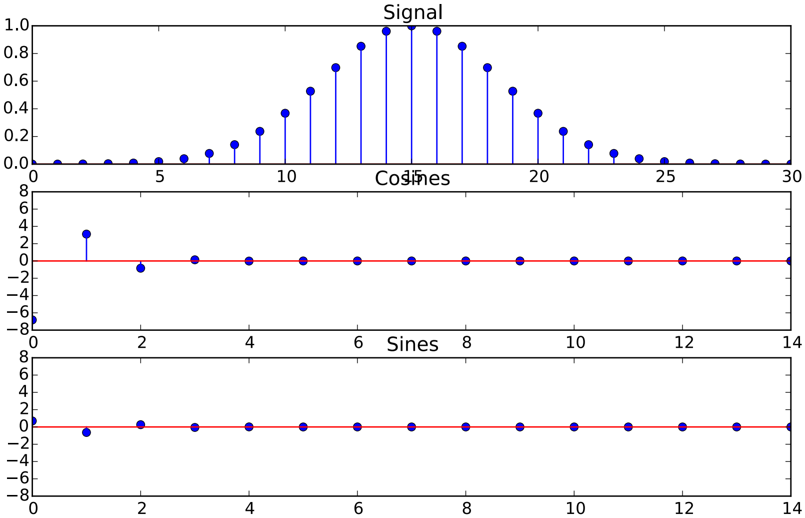
Is a high dimensional rotation! This means these sinusoids form a *basis for all signals of length N. This is known as the* Discrete Fourier Transform

Discrete Fourier Transform: Example



Show sinusoidal addition

Discrete Fourier Transform: Example



Show sinusoidal addition

Discrete Fourier Transform: Formal Definition

The formal definition of the Discrete Fourier Transform uses complex numbers to store the phase, for convenience

$$F[k] = \sum_{t=0}^{N-1} X[n] e^{-i2\pi \frac{k}{N} n}$$

$$F[k] = \sum_{t=0}^{N-1} X[n] \cos\left(2\pi \frac{k}{N} n\right) + i \sum_{t=0}^{N-1} X[n] \sin\left(-2\pi \frac{k}{N} n\right)$$

$$F[k] = \mathrm{Re}(F[k]) + i\mathrm{Imag}(F[k])$$

Inverse Discrete Fourier Transform

DFT decomposed X into a bunch of sinusoids, now add them back together

$$X[n] = \sum_{k=0}^{N-1} F[k] e^{i2\pi \frac{k}{N} n}$$

Functions on The Circle