Indexing Time Series Data

Slides based on previous slides of Prof. Christos Faloutsos and Prof. Eamonn Keogh

Outline

- Spatial Databases
- Temporal Databases
- Spatio-temporal Databases
- Multimedia Databases
  - Time Series databases
  - Text databases
  - Image and video databases

Time Series Databases

- A time series is a sequence of real numbers, representing the measurements of a real variable at equal time intervals
  - Stock prices
  - Volume of sales over time
  - Daily temperature readings
  - ECG (electrocardiogram) data

- A time series database is a large collection of time series

Time Series Data

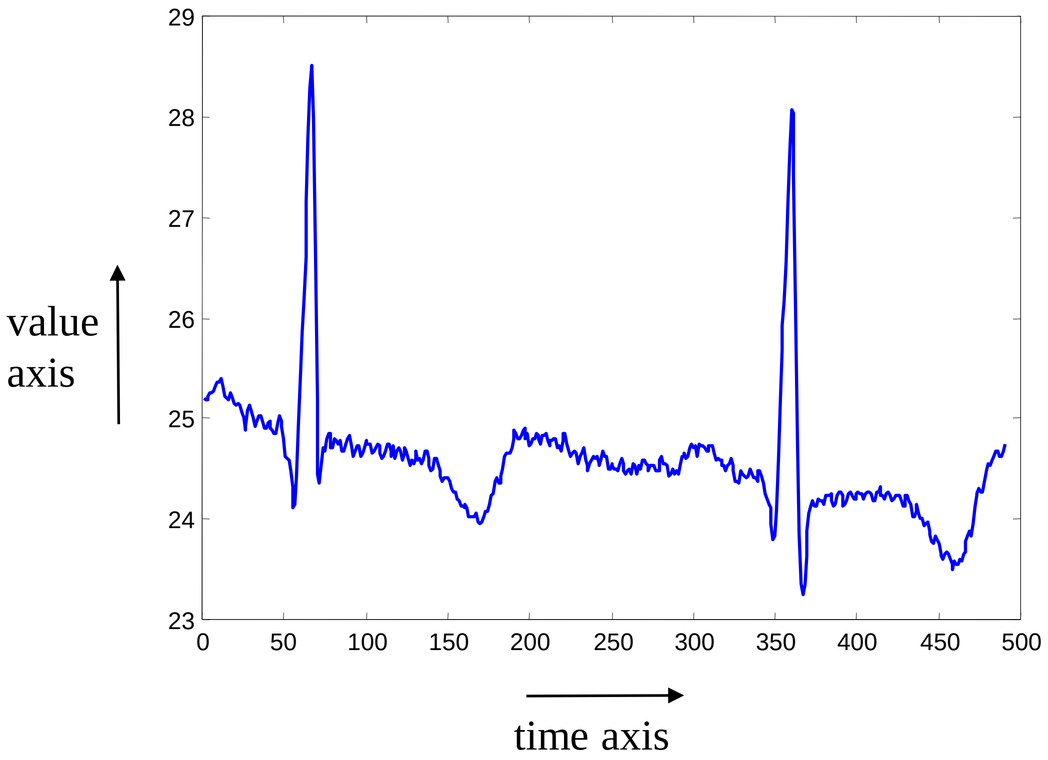
A time series is a collection of observations made sequentially in time.

25.1750
25.1750
25.2250
25.2500
25.2500
25.2750
25.3250
25.3500
25.3500
25.4000
25.4000
25.3250
25.2250
25.2000
25.1750

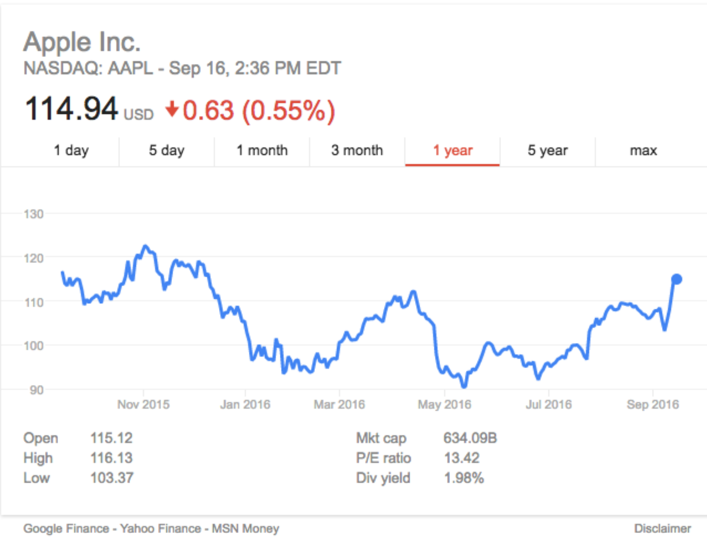
..

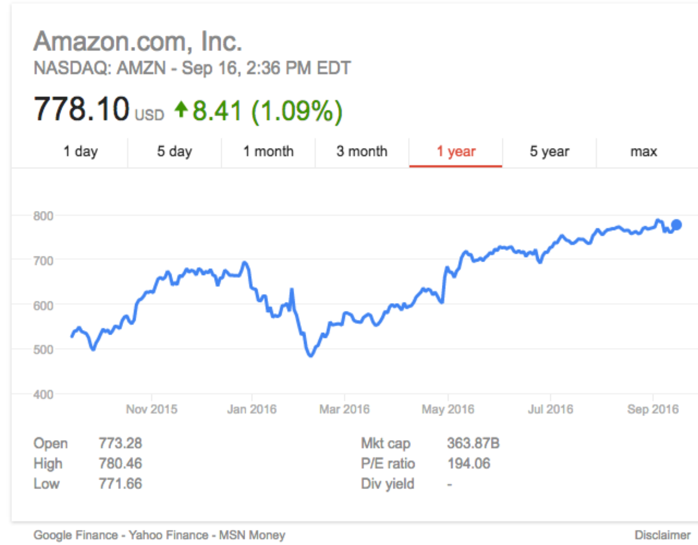
..

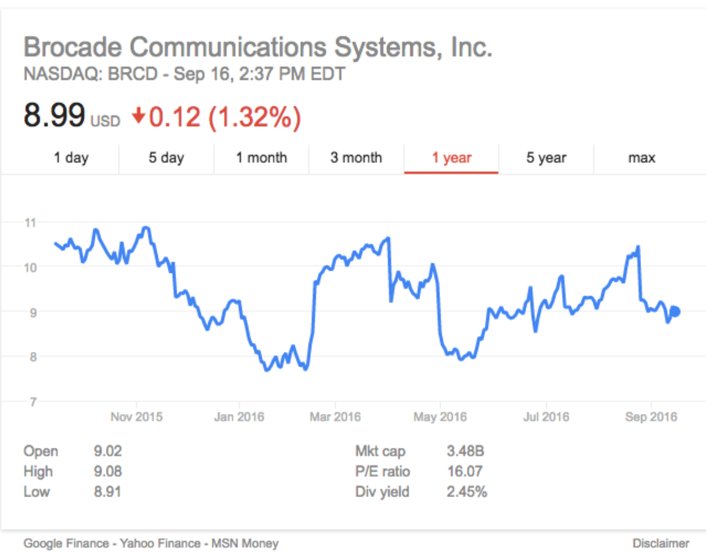
24.6250
24.6750
24.6750
24.6250
24.6250
24.6250
24.6750
24.7500



Stock example







Time Series Problems
(from a database perspective)

- The Similarity Problem

  $X = x_1, x_2, \ldots, x_n$ and $Y = y_1, y_2, \ldots, y_n$

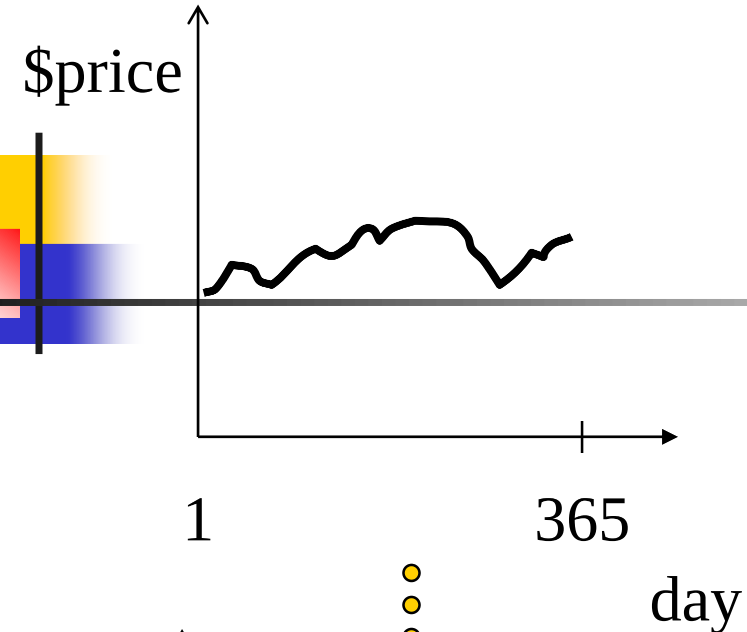
- Define and compute Sim(X, Y)
  - E.g. do stocks X and Y have similar movements?
- Retrieve efficiently similar time series (Indexing for Similarity Queries)

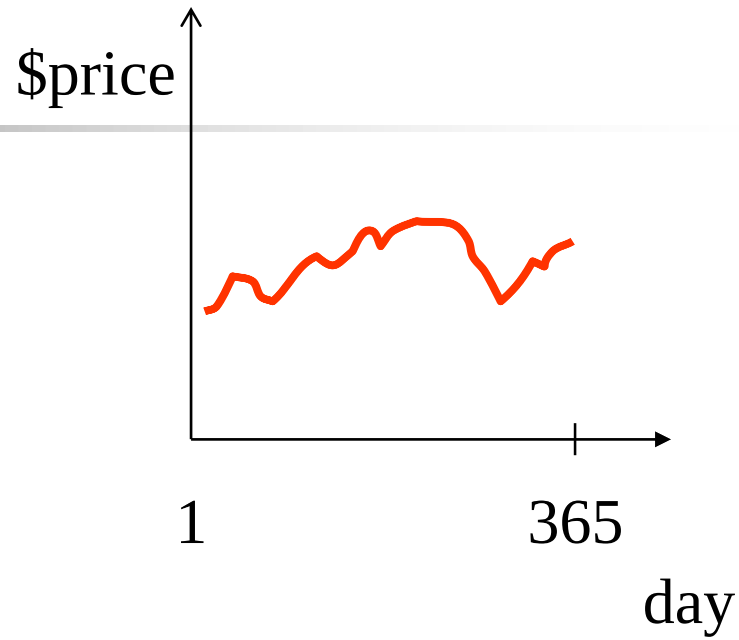
Types of queries

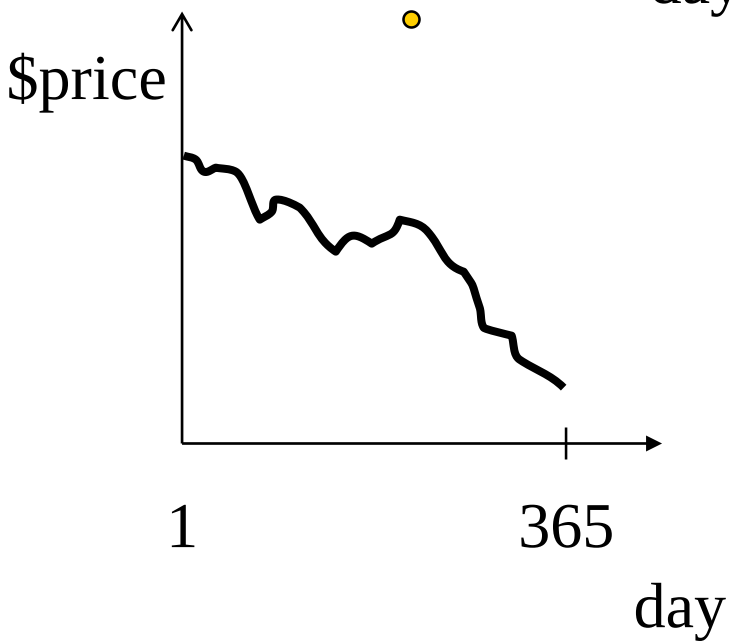
- whole match vs sub-pattern match (sub-sequence)
- range query vs nearest neighbors
- all-pairs query

Examples

- Find companies with similar stock prices over a time interval
- Find products with similar sell cycles
- Cluster users with similar credit card utilization
- Find similar subsequences in DNA sequences
- Find scenes in video streams







distance function: by expert

(eg, Euclidean distance)

Problems

- Define the similarity (or distance) function
- Find an efficient algorithm to retrieve similar time series from a database
  - (Faster than sequential scan)

The Similarity function depends on the Application

Metric Distances

- What properties should a similarity distance have (ideally)?

- $D(A,B) = D(B,A)$ *Symmetry*
- $D(A,A) = 0$ *Constancy of Self-Similarity*
- $D(A,B) >= 0$ *Positivity*
- $D(A,B) \leq D(A,C) + D(B,C)$ *Triangular Inequality*

"Euclidean" Similarity Measure

- View each sequence as a point in n-dimensional Euclidean space (n = length of each sequence)
- Define (dis-)similarity between sequences X and Y as

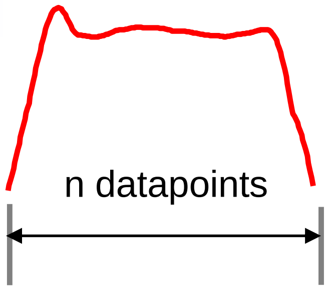
$$\boxed{L_p = \left(\sum_{i=1}^{n} | x_i - y_i |^p\right)^{1/p}}$$

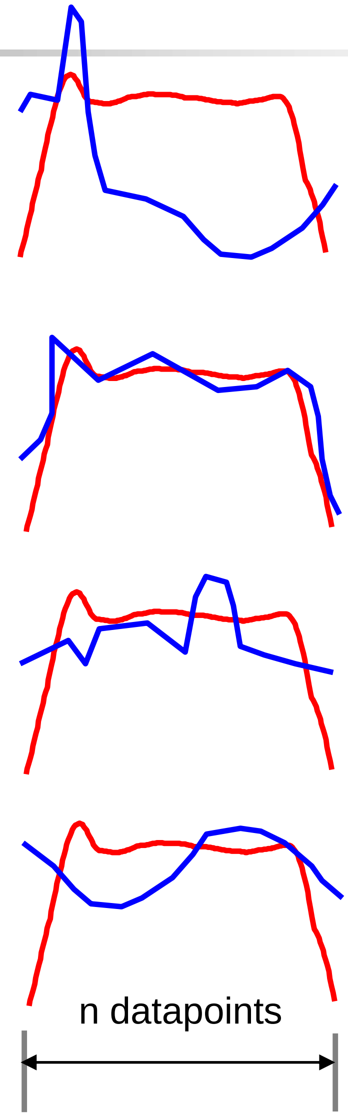
p=1 Manhattan distance

p=2 Euclidean distance

Euclidean model

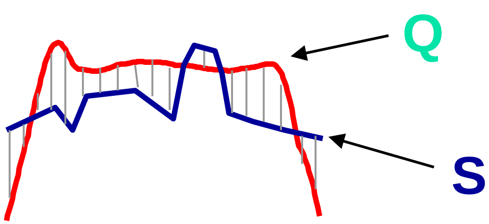
Query Q





Database	Distance	Rank
	0.98	4
	0.07	1
	0.21	2
	0.43	3

Euclidean Distance between two time series $Q = \{q_1, q_2, \ldots, q_n\}$ and $S = \{s_1, s_2, \ldots, s_n\}$



$$D(Q,S) \equiv \sqrt{\sum_{i=1}^{n} (q_i - s_i)^2}$$

Advantages

- Easy to compute: O(n)
- Allows scalable solutions to other problems, such as
  - indexing
  - clustering
  - etc...

Similarity Retrieval

- Range Query
  - Find all time series S where $D(Q,S) \leq \varepsilon$
- Nearest Neighbor query
  - Find all the k most similar time series to Q
- A method to answer the above queries: Linear scan … very slow

- A better approach GEMINI

GEMINI

Solution: Quick-and-dirty' filter:

- extract *m* features (numbers, eg., avg., etc.)
- map into a point in *m*-d feature space
- organize points with off-the-shelf spatial access method (‘SAM’, e.g., R-tree)
- retrieve the answer using a NN query
- discard false alarms

GEMINI Range Queries

Build an index for the database in a feature space using an R-tree

Algorithm RangeQuery(Q, ε)

1. Project the query Q into a point q in the feature space
2. Find all candidate objects in the index within ε
3. Retrieve from disk the actual sequences
4. Compute the actual distances and discard false alarms

GEMINI NN Query

Algorithm K_NNQuery(Q, K)

1. Project the query Q in the same feature space
2. Find the candidate K nearest neighbors in the index
3. Retrieve from disk the actual sequences pointed to by the candidates
4. Compute the actual distances and record the maximum
5. Issue a RangeQuery(Q, εmax)
6. Compute the actual distances, return best K

GEMINI

- GEMINI works when:

  $D_{feature}(F(x), F(y)) <= D(x, y)$

- *Note that, the closer the feature distance to the actual one, the better.*

Problem

- How to extract the features? How to define the feature space?

- Fourier transform
- Wavelets transform
- Averages of segments (Histograms or APCA)
- Chebyshev polynomials
- .... your favorite curve approximation...

Fourier transform

- DFT (Discrete Fourier Transform)
- Transform the data from the time domain to the frequency domain
- highlights the periodicities
- SO?

DFT

A: several real sequences are periodic

Q: Such as?

A:

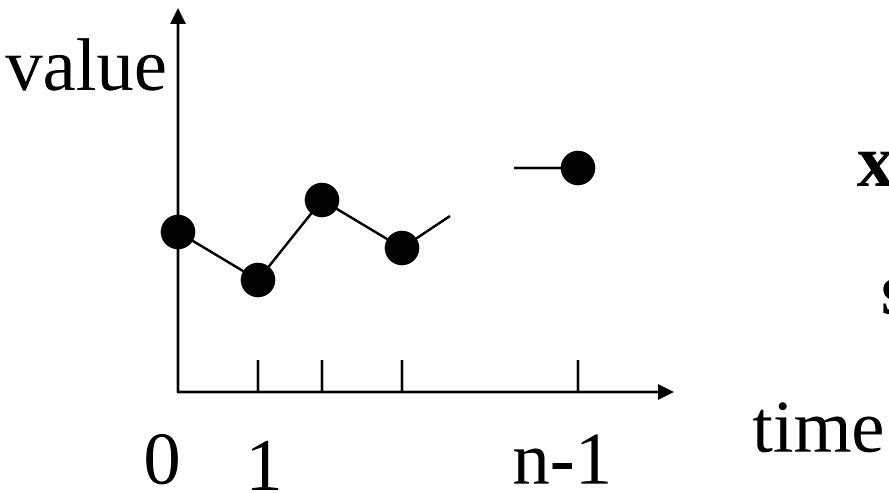
- sales patterns follow seasons;
- economy follows 50-year cycle (or 10?)
- temperature follows daily and yearly cycles

Many real signals follow (multiple) cycles

How does it work?

Decomposes signal to a sum of sine and cosine waves.

Q:How to assess 'similarity' of **x** with a (discrete) wave?



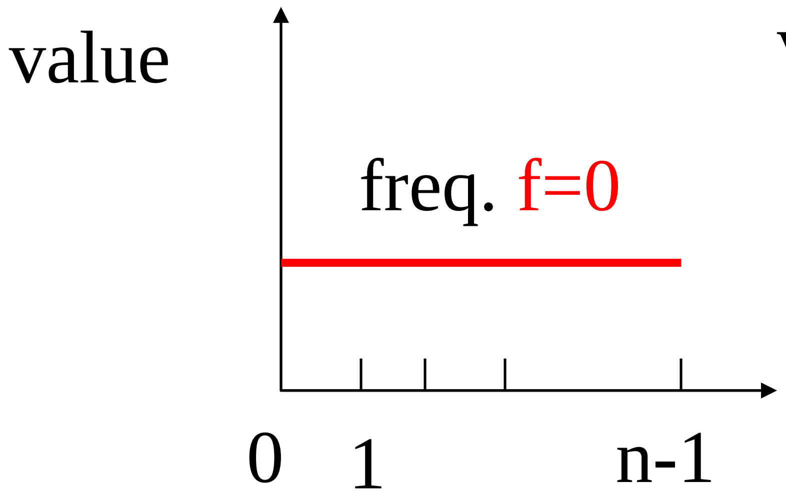
$$\mathbf{x} = \{x_0, x_1, \ldots x_{n-1}\}$$

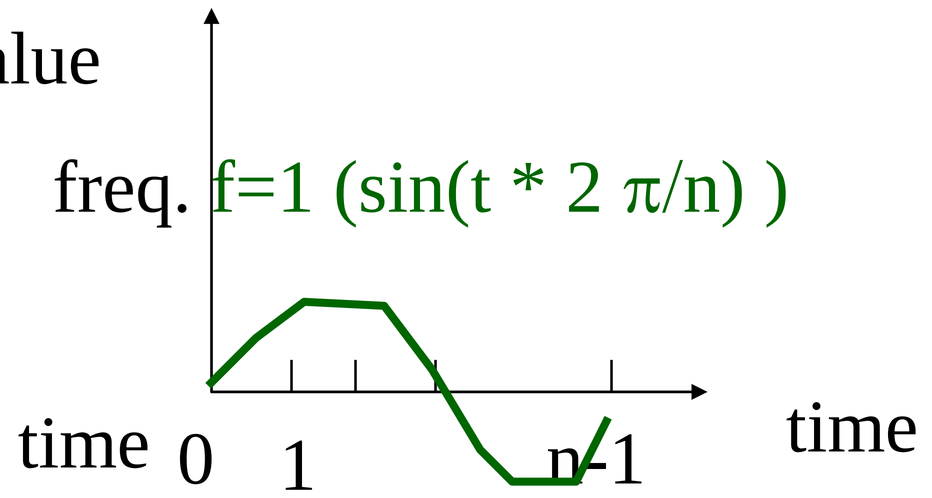
$$\mathbf{s} = \{s_0, s_1, \ldots s_{n-1}\}$$

How does it work?

A: consider the waves with frequency 0, 1, ...; use the inner-product (~cosine similarity)

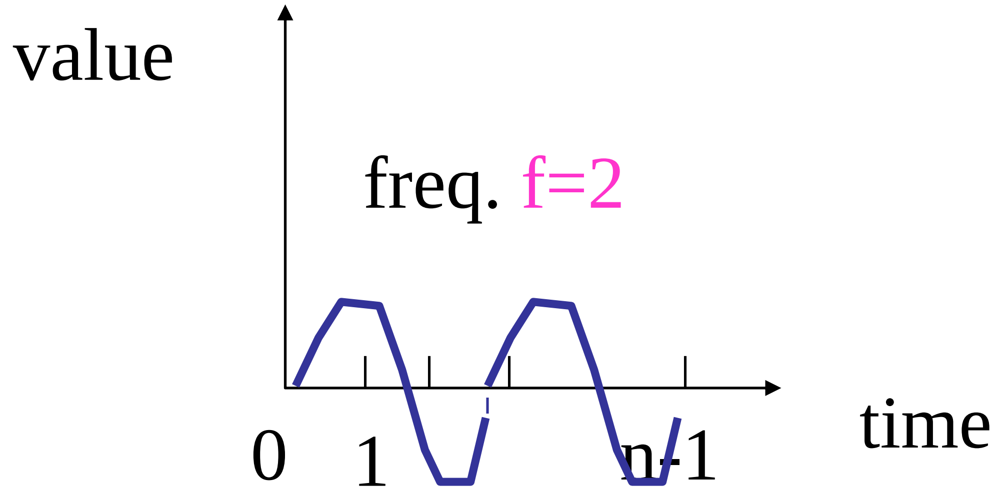
Freq=1/period





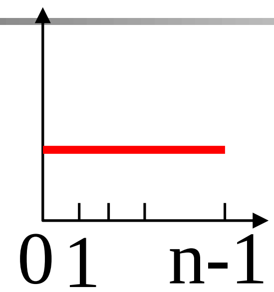
How does it work?

A: consider the waves with frequency 0, 1, ...; use the inner-product (~cosine similarity)

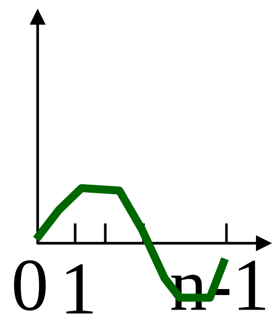


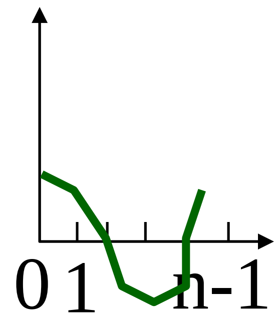
How does it work?

<u>'basis' functions</u>



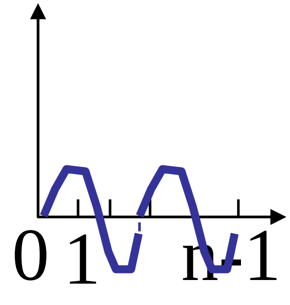
sine, freq =1

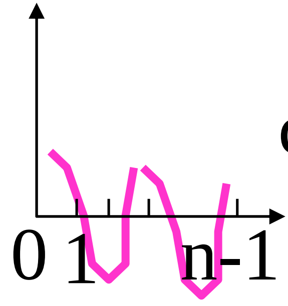




cosine, f=1

sine, freq = 2





cosine, f=2

How does it work?

- Basis functions are actually n-dim vectors, **orthogonal** to each other
- 'similarity' of **x** with each of them: inner product
- DFT: ~ all the similarities of **x** with the basis functions

How does it work?

Since $e^{jf} = cos(f) + j\ sin(f)$ (j=$sqrt$(-1)),

we finally have:

DFT: definition

- Discrete Fourier Transform (n-point):

$$X_f = 1/\sqrt{n} \sum_{t=0}^{n-1} x_t * \exp(-j 2\pi\, tf / n)$$

$$(j = \sqrt{-1})$$

inverse DFT

$$x_t = 1/\sqrt{n} \sum_{t=0}^{n-1} X_f * \exp(+j 2\pi\, tf / n)$$

DFT: properties

Observation - SYMMETRY property:

$$X_f = (X_{n-f})^*$$

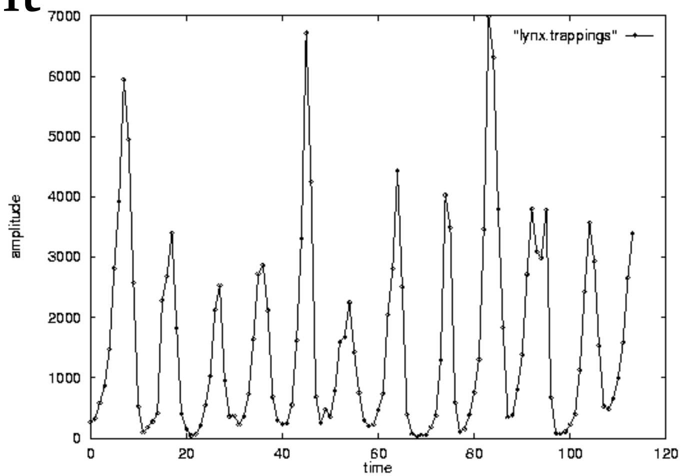
("*": complex conjugate: $(a + b\,j)^* = a - b\,j$)

Thus we use only the first half numbers

DFT: Amplitude spectrum

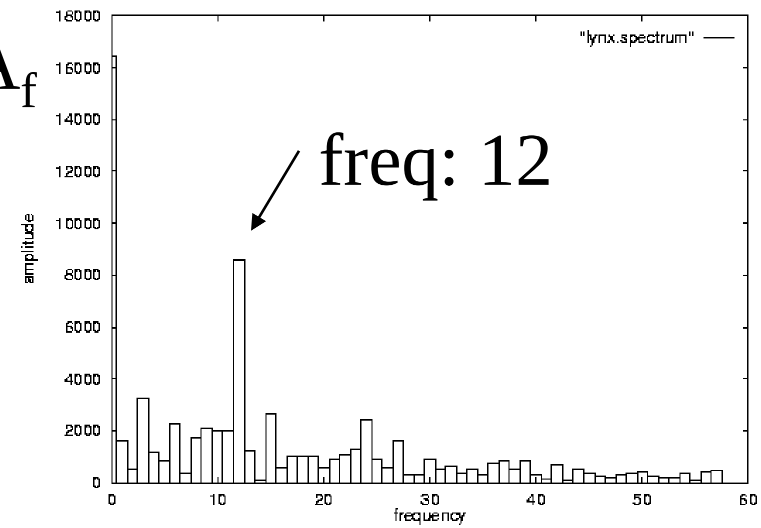
- Amplitude $$A_f^{\,2} = \mathrm{Re}^2(X_f) + \mathrm{Im}^2(X_f)$$
- Intuition: strength of frequency '*f*'

count



time

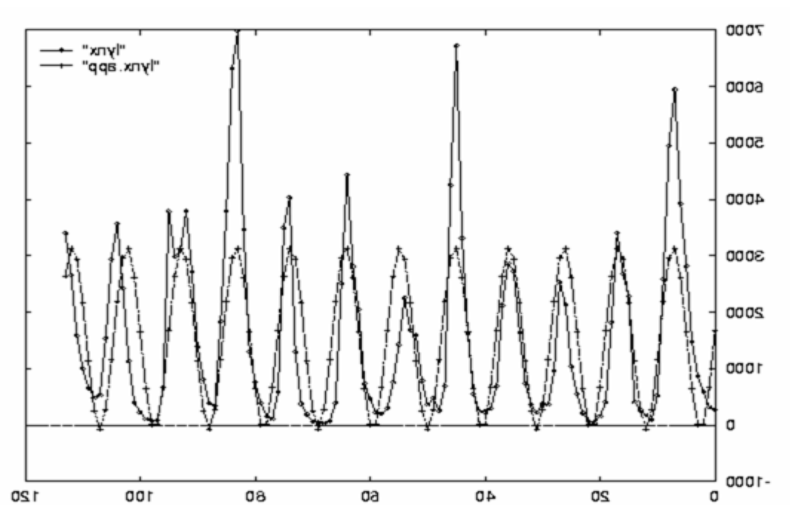
A_f

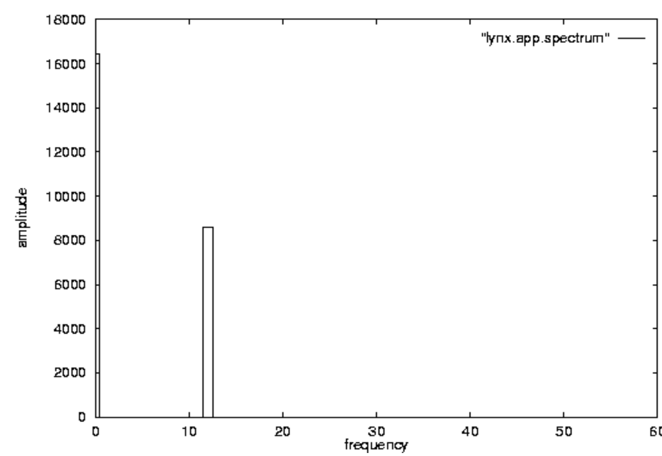


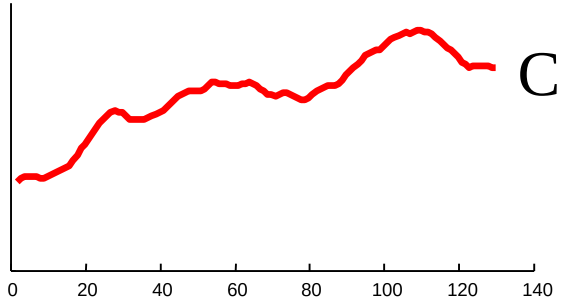
freq. *f*

DFT: Amplitude spectrum

- excellent approximation, with only 2 frequencies!
- so what?







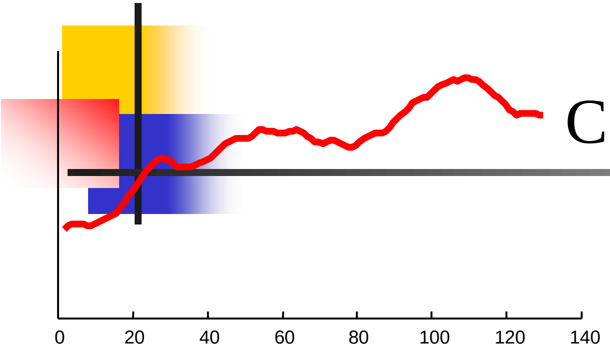
$n = 128$

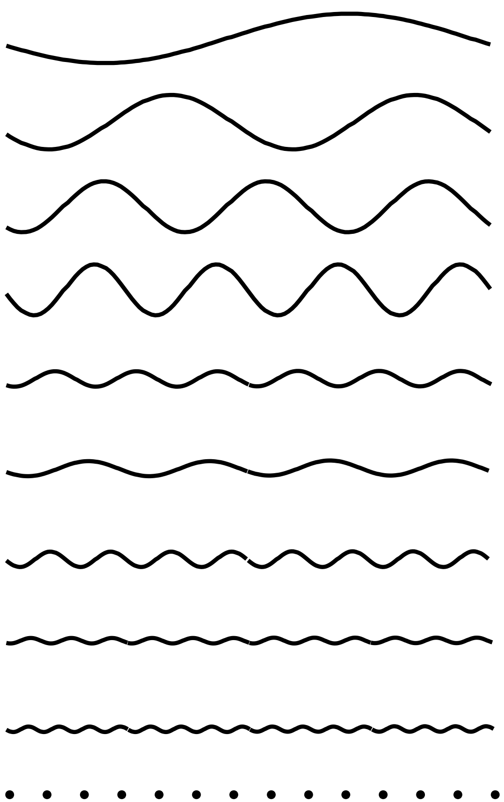
Raw Data

0.4995
0.5264
0.5523
0.5761
0.5973
0.6153
0.6301
0.6420
0.6515
0.6596
0.6672
0.6751
0.6843
0.6954
0.7086
0.7240
0.7412
0.7595
0.7780
0.7956
0.8115
0.8247
0.8345
0.8407
0.8431
0.8423
0.8387
…

The graphic shows a time series with 128 points.

The raw data used to produce the graphic is also reproduced as a column of numbers (just the first 30 or so points are shown).

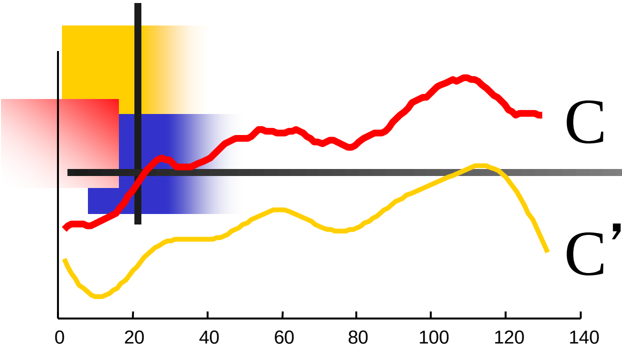


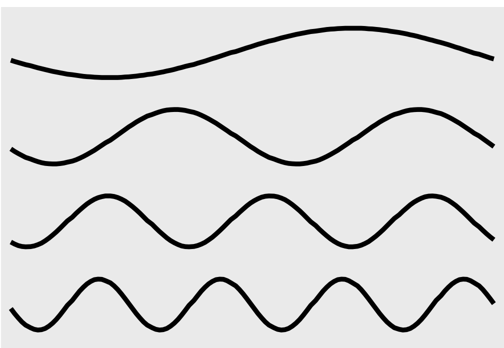


Raw Data	Fourier Coefficients
0.4995	1.5698
0.5264	1.0485
0.5523	0.7160
0.5761	0.8406
0.5973	0.3709
0.6153	0.4670
0.6301	0.2667
0.6420	0.1928
0.6515	0.1635
0.6596	0.1602
0.6672	0.0992
0.6751	0.1282
0.6843	0.1438
0.6954	0.1416
0.7086	0.1400
0.7240	0.1412
0.7412	0.1530
0.7595	0.0795
0.7780	0.1013
0.7956	0.1150
0.8115	0.1801
0.8247	0.1082
0.8345	0.0812
0.8407	0.0347
0.8431	0.0052
0.8423	0.0017
0.8387	0.0002
…	...

We can decompose the data into 64 pure sine waves using the Discrete Fourier Transform (just the first few sine waves are shown).

The Fourier Coefficients are reproduced as a column of numbers (just the first 30 or so coefficients are shown).





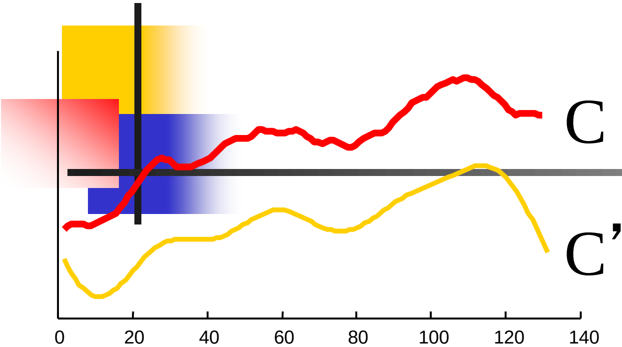
Raw Data	Fourier Coefficients	Truncated Fourier Coefficients
0.4995	1.5698	1.5698
0.5264	1.0485	1.0485
0.5523	0.7160	0.7160
0.5761	0.8406	0.8406
0.5973	0.3709	0.3709
0.6153	0.4670	0.4670
0.6301	0.2667	0.2667
0.6420	0.1928	0.1928
0.6515	0.1635	
0.6596	0.1602	
0.6672	0.0992	
0.6751	0.1282	
0.6843	0.1438	
0.6954	0.1416	
0.7086	0.1400	
0.7240	0.1412	
0.7412	0.1530	
0.7595	0.0795	
0.7780	0.1013	
0.7956	0.1150	
0.8115	0.1801	
0.8247	0.1082	
0.8345	0.0812	
0.8407	0.0347	
0.8431	0.0052	
0.8423	0.0017	
0.8387	0.0002	
…	...	

$n = 128$

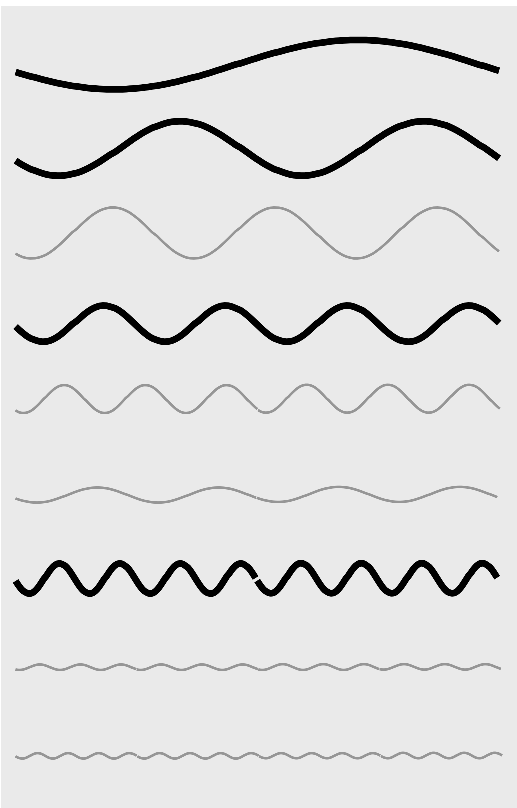
$N = 8$

$C_{\text{ratio}} = 1/16$

We have discarded $\frac{15}{16}$ of the data.



Raw Data	Fourier Coefficients	Sorted Truncated Fourier Coefficients
0.4995	1.5698	1.5698
0.5264	1.0485	1.0485
0.5523	0.7160	0.7160
0.5761	0.8406	0.8406
0.5973	0.3709	0.2667
0.6153	0.1670	0.1928
0.6301	0.4667	0.1438
0.6420	0.1928	0.1416
0.6515	0.1635	
0.6596	0.1302	
0.6672	0.0992	
0.6751	0.1282	
0.6843	0.2438	
0.6954	0.2316	
0.7086	0.1400	
0.7240	0.1412	
0.7412	0.1530	
0.7595	0.0795	
0.7780	0.1013	
0.7956	0.1150	
0.8115	0.1801	
0.8247	0.1082	
0.8345	0.0812	
0.8407	0.0347	
0.8431	0.0052	
0.8423	0.0017	
0.8387	0.0002	
…	…	



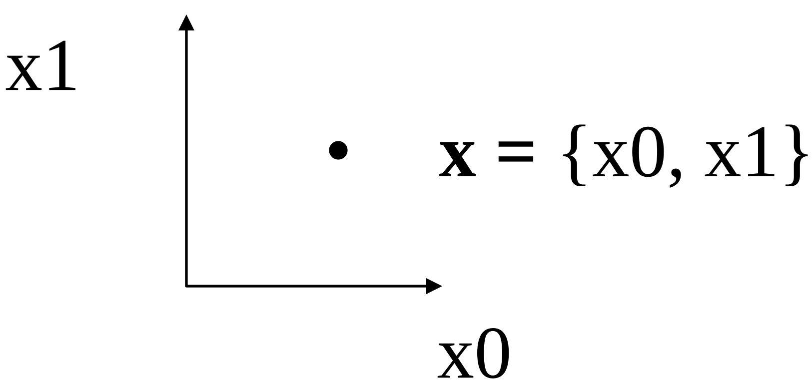
Instead of taking the first few coefficients, we could take the *best* coefficients

DFT: Parseval's theorem

$$\text{sum}(x_t^{\,2}) = \text{sum} (| X_f |^2)$$

Ie., DFT preserves the 'energy'

or, alternatively: it does an axis rotation:



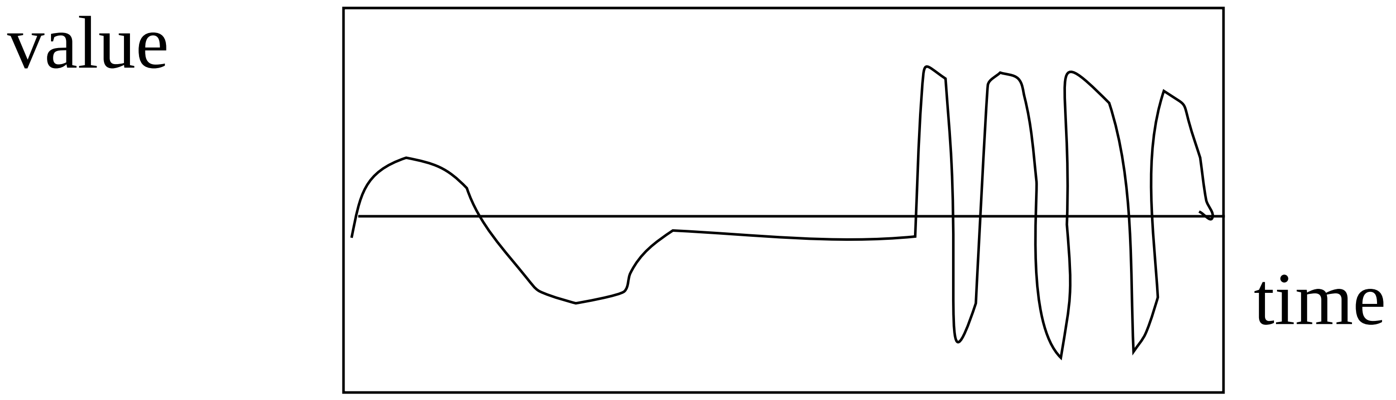
Lower Bounding lemma

- Using Parseval's theorem we can prove the lower bounding property!

- So, apply DFT to each time series, keep first 3-10 coefficients as a vector and use an R-tree to index the vectors
- R-tree works with euclidean distance, OK.

Wavelets - DWT

- DFT is great - but, how about compressing opera? (baritone, silence, soprano?)

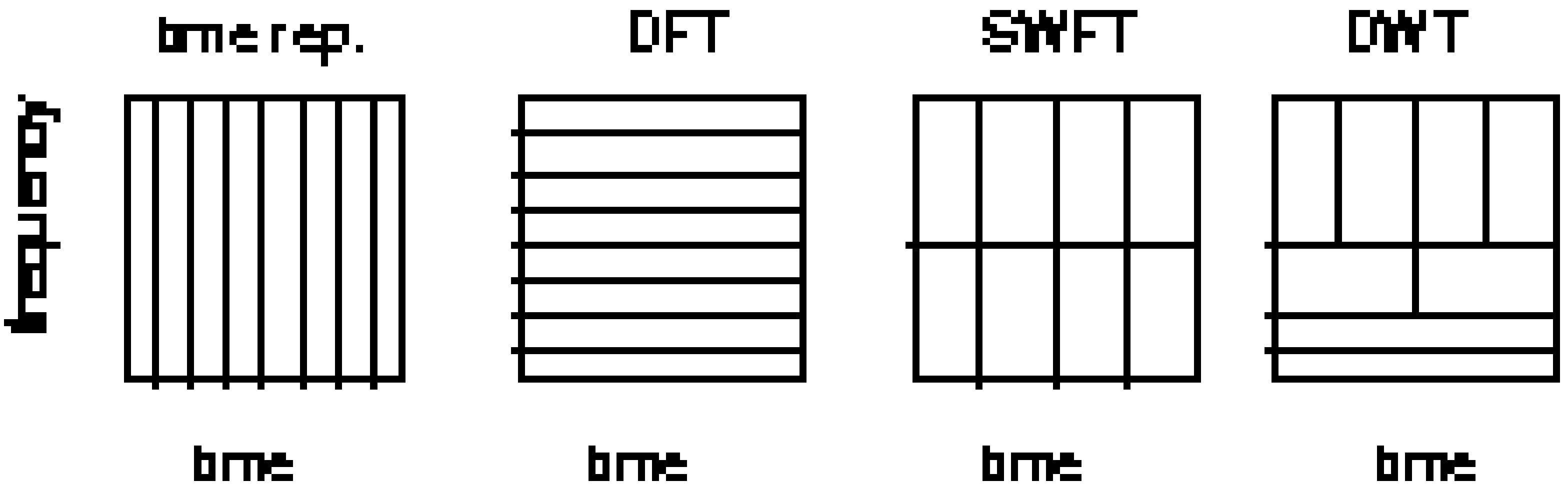


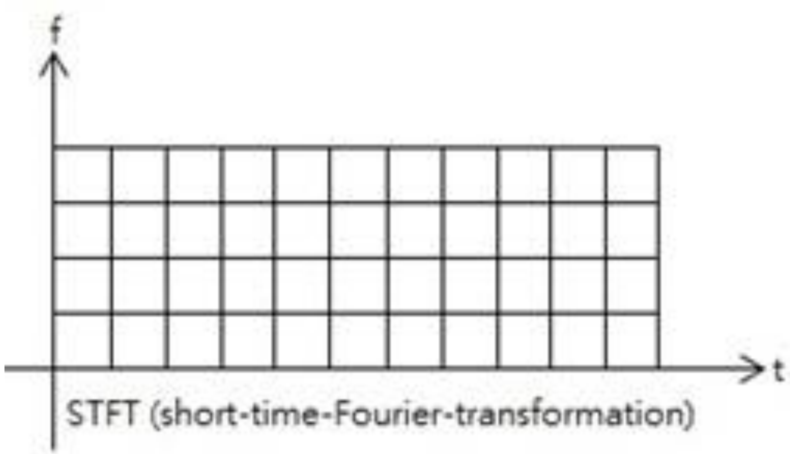
Wavelets - DWT

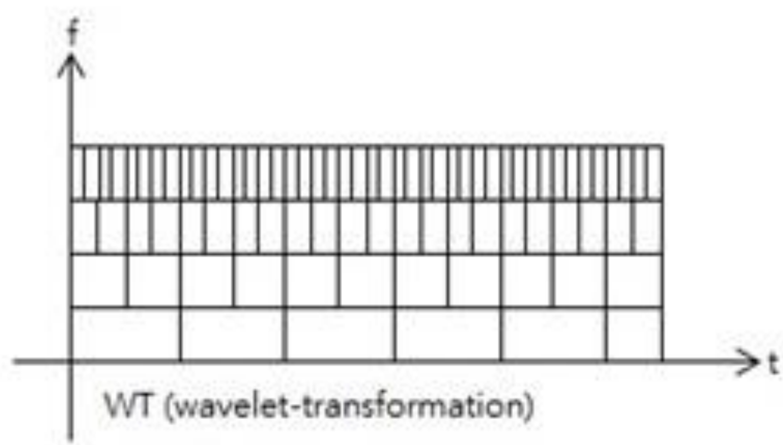
- Solution#1: Short window Fourier transform
- But: how short should be the window?

Wavelets - DWT

- Answer: **multiple** window sizes! -> DWT

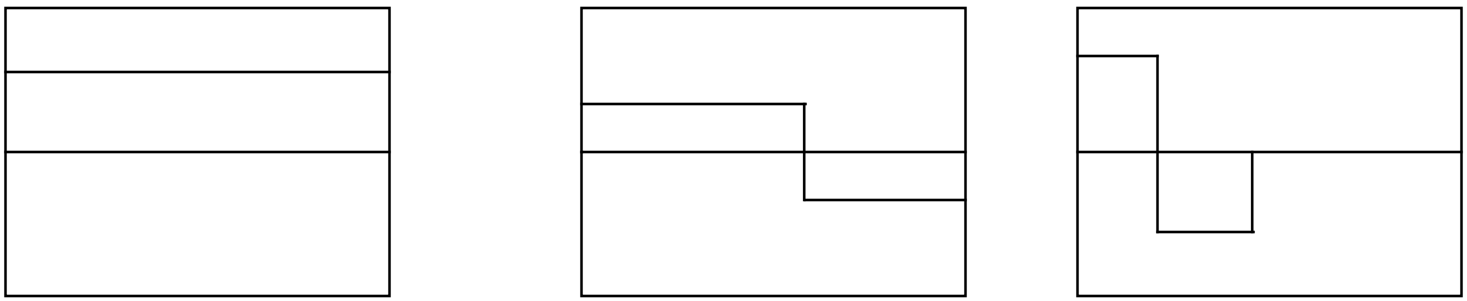



f
t
STFT (short-time-Fourier-transformation)


f
t
WT (wavelet-transformation)

Haar Wavelets

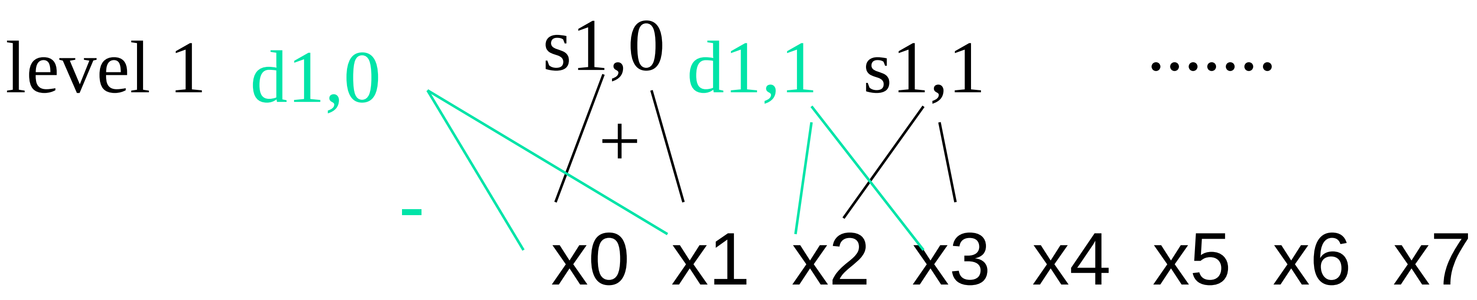
- subtract sum of left half from right half
- repeat recursively for quarters, eightths ...
- Basis functions are step functions with different lenghts



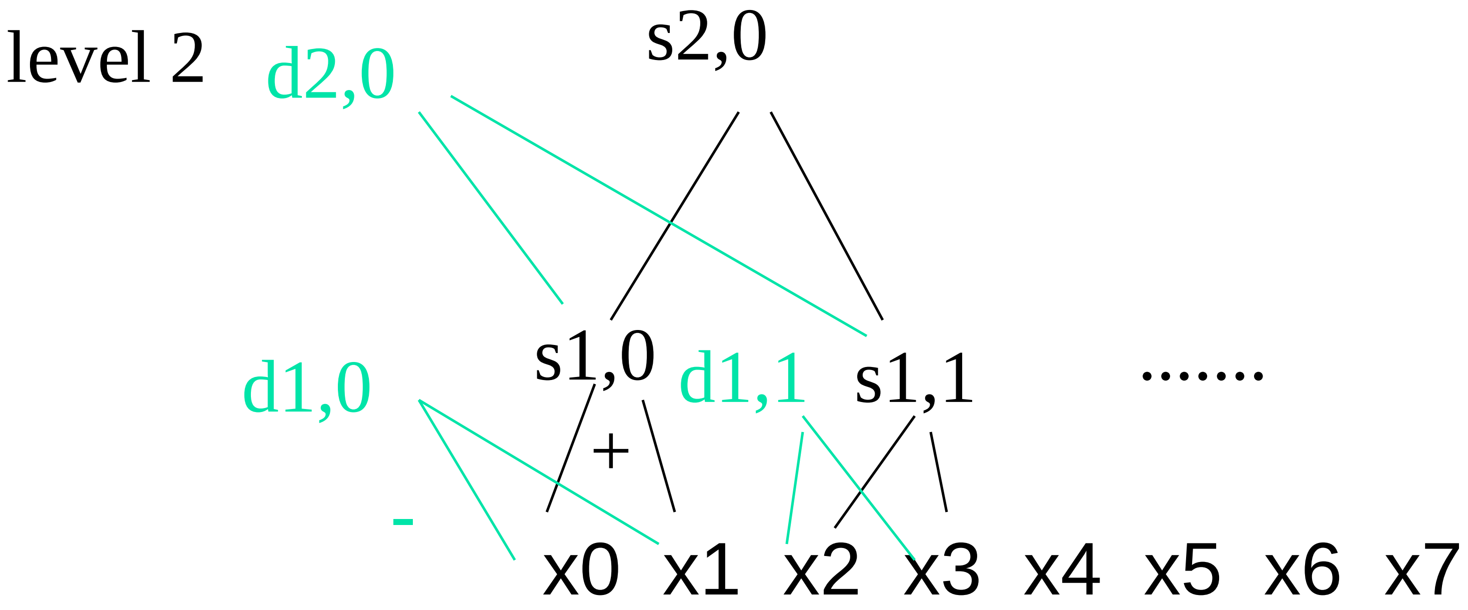
Wavelets - construction

x0 x1 x2 x3 x4 x5 x6 x7

Wavelets - construction

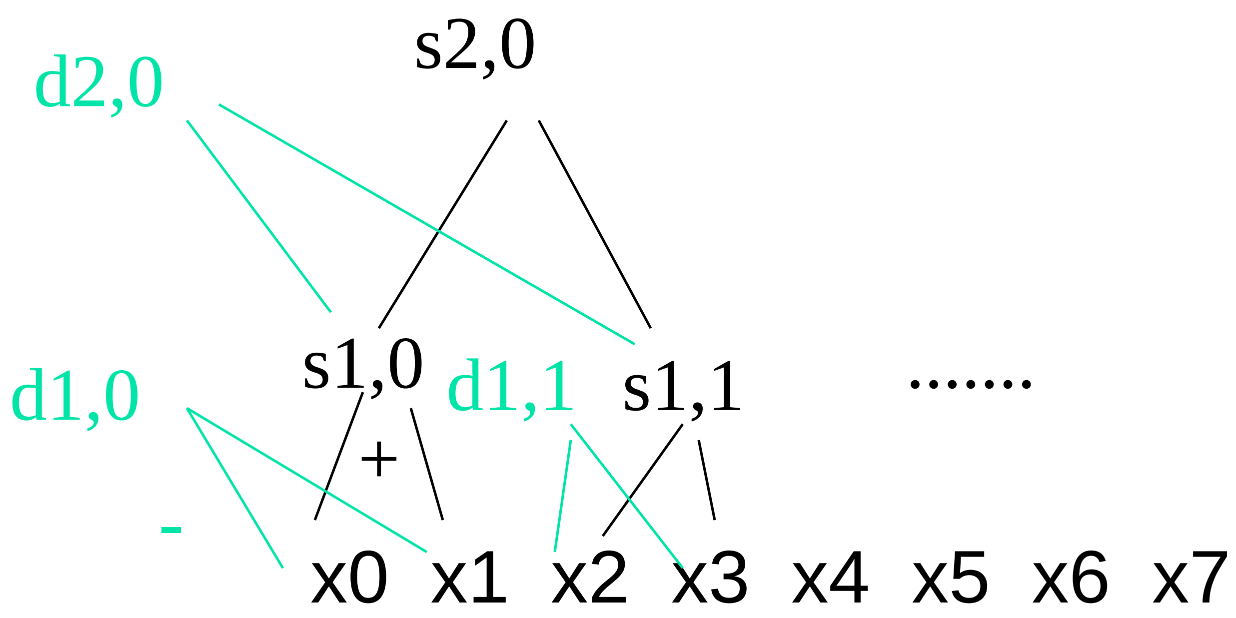


Wavelets - construction



Wavelets - construction

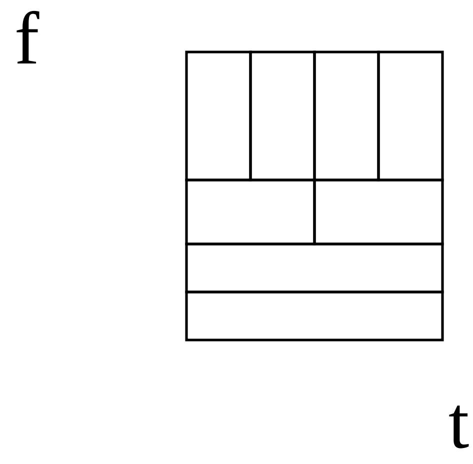
etc ...

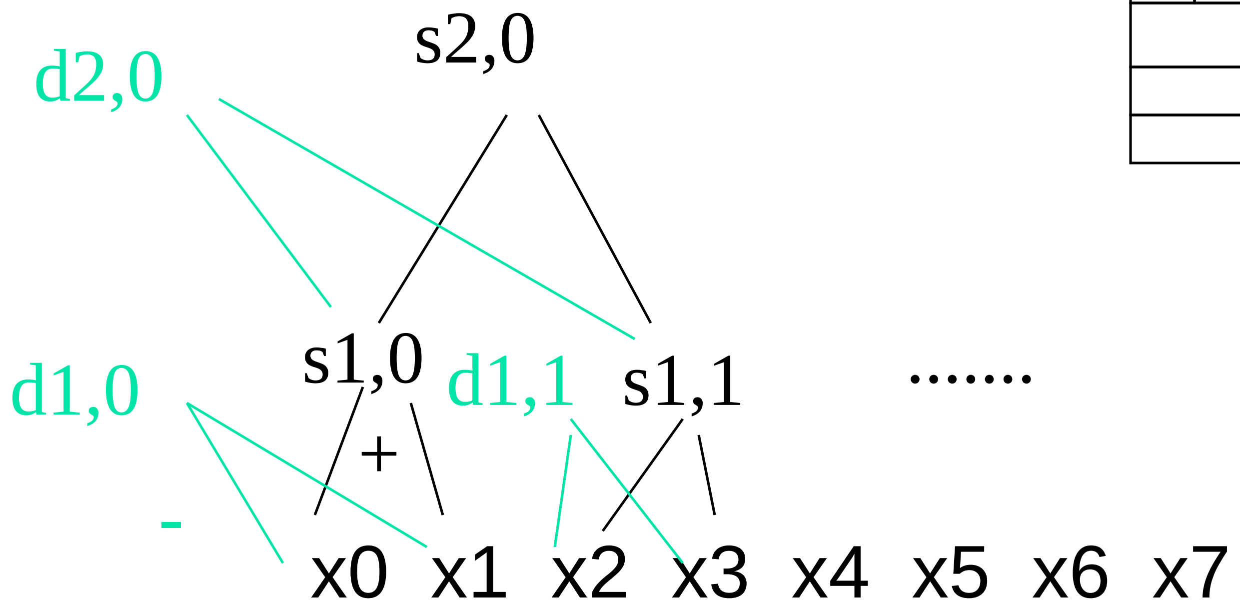


Wavelets - construction

Q: map each coefficient

on the time-freq. plane

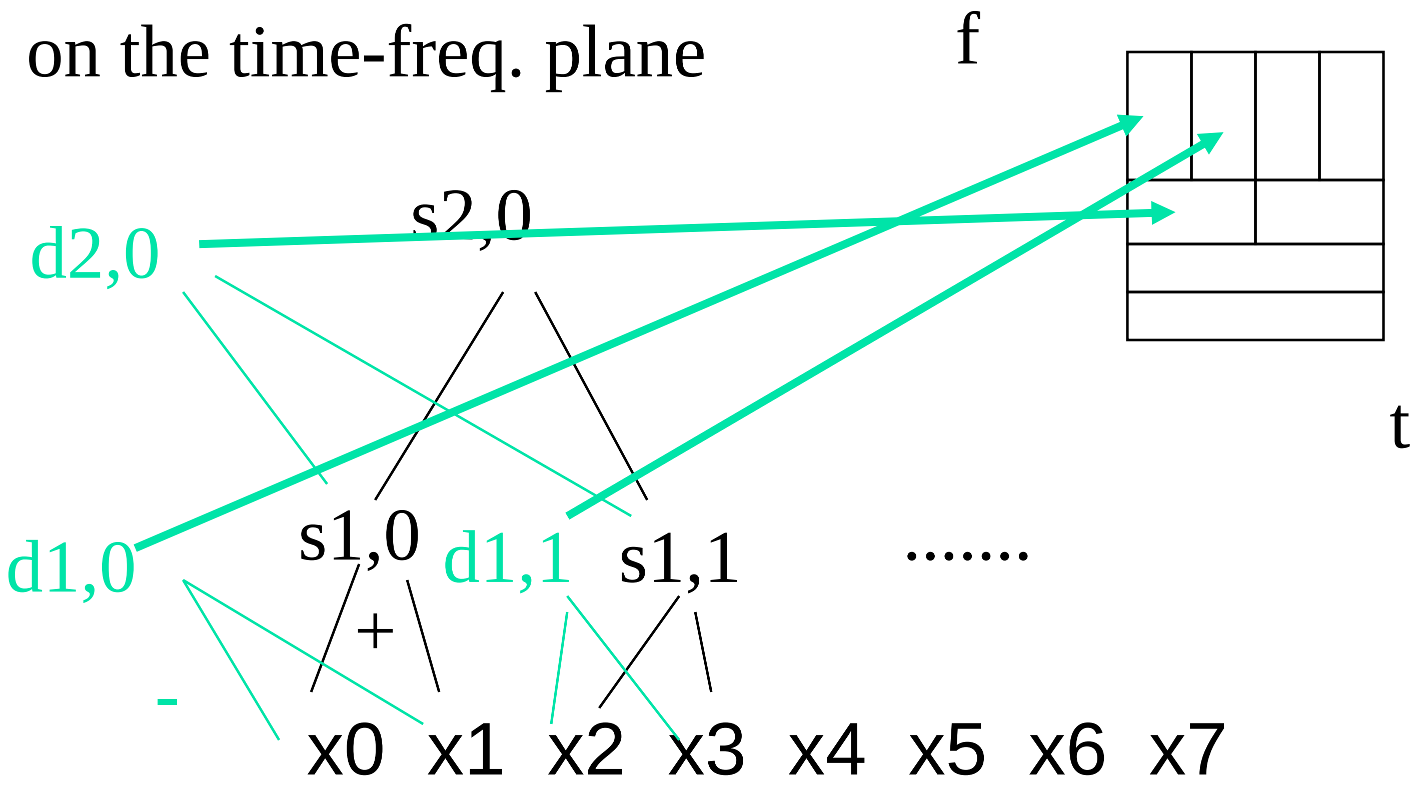




Wavelets - construction

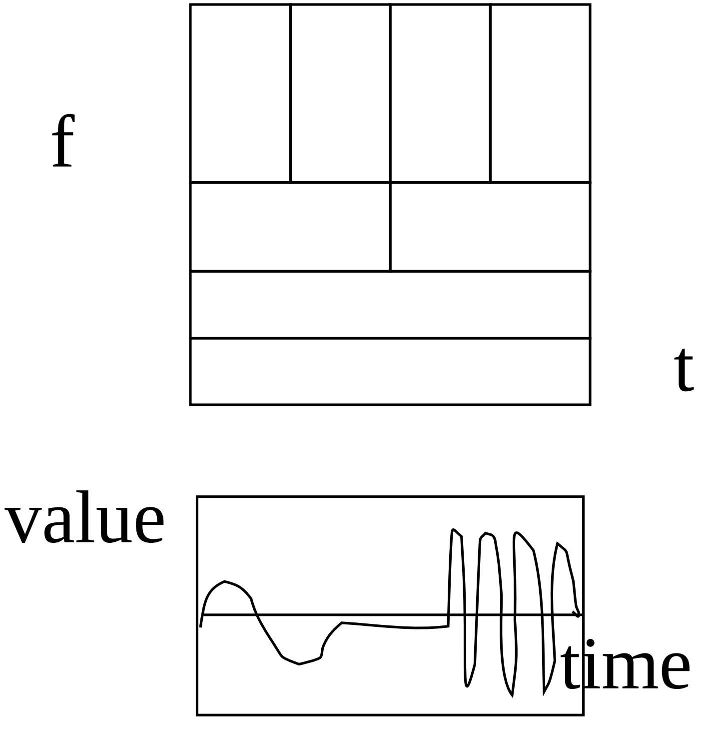
Q: map each coefficient

on the time-freq. plane



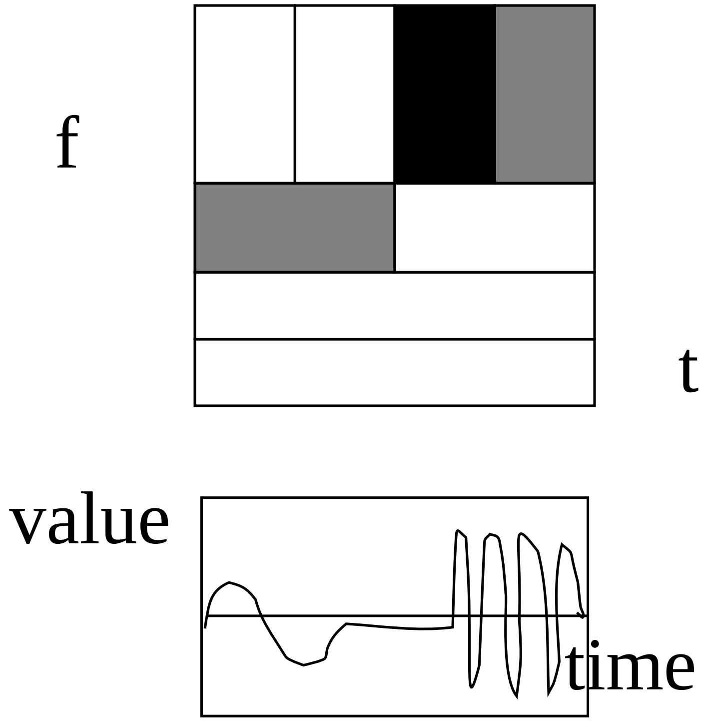
Wavelets - Drill:

- Q: baritone/silence/soprano - DWT?



Wavelets - Drill:

- Q: baritone/soprano - DWT?



Wavelets - construction

Observation1:

- ‘+’ can be some weighted addition
- ‘-’ is the corresponding weighted difference (‘Quadrature mirror filters’)

Observation2: unlike DFT/DCT,

- there are *many* wavelet bases: Haar, Daubechies-4, Daubechies-6, ...

Advantages of Wavelets

- Better compression (better RMSE with same number of coefficients)
- closely related to the processing of the mammalian eye and ear
- Good for progressive transmission
- handle spikes well
- usually, fast to compute (O(*n*)!)

Feature space

- Keep the d most “important” wavelets coefficients
- Normalize and keep the largest
- Lower bounding lemma: the same as DFT