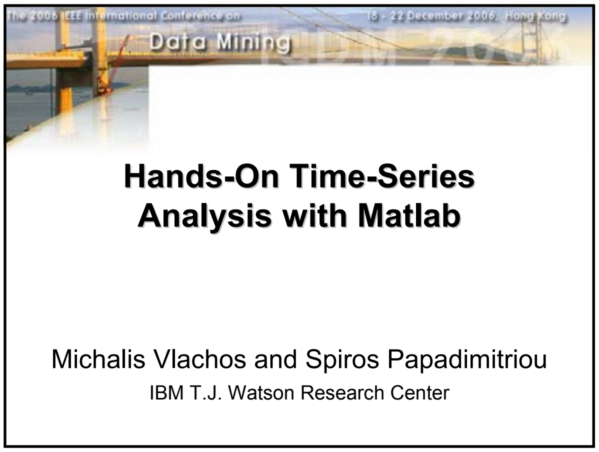
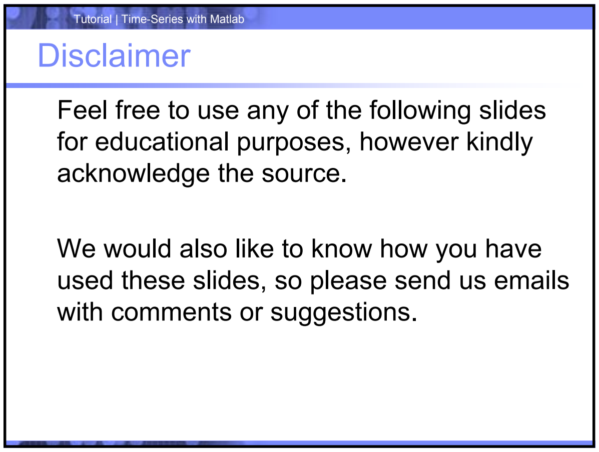

The 2006 IEEE International Conference on
Data Mining
18 - 22 December 2006, Hong Kong

Hands-On Time-Series Analysis with Matlab

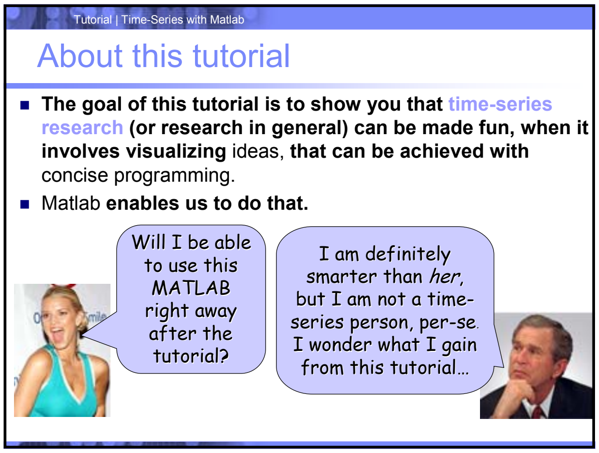
Michalis Vlachos and Spiros Papadimitriou
IBM T.J. Watson Research Center



Disclaimer

Feel free to use any of the following slides for educational purposes, however kindly acknowledge the source.

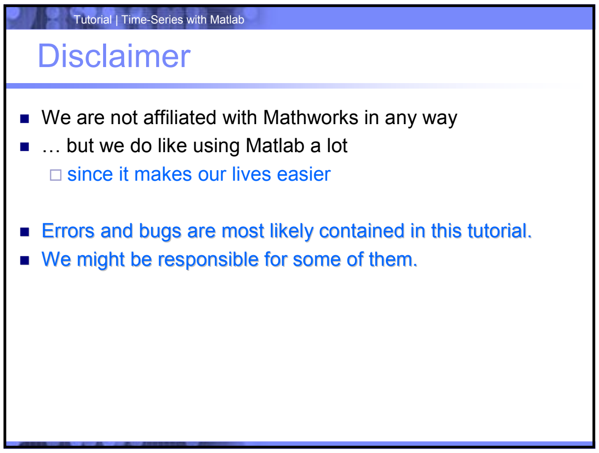
We would also like to know how you have used these slides, so please send us emails with comments or suggestions.



About this tutorial

- **The goal of this tutorial is to show you that** time-series research **(or research in general) can be made fun, when it involves visualizing** ideas, **that can be achieved with** concise programming.
- Matlab **enables us to do that.**

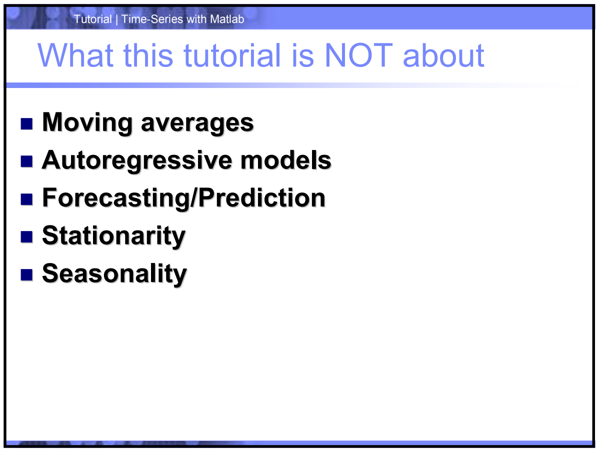
Will I be able to use this MATLAB right away after the tutorial?
I am definitely smarter than *her*, but I am not a time-series person, per-se. I wonder what I gain from this tutorial…



Disclaimer

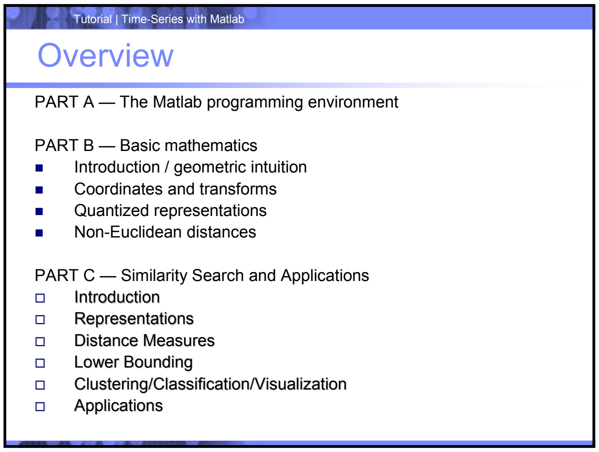
- We are not affiliated with Mathworks in any way
- … but we do like using Matlab a lot
  - □ since it makes our lives easier

- Errors and bugs are most likely contained in this tutorial.
- We might be responsible for some of them.



What this tutorial is NOT about

- **Moving averages**
- **Autoregressive models**
- **Forecasting/Prediction**
- **Stationarity**
- **Seasonality**

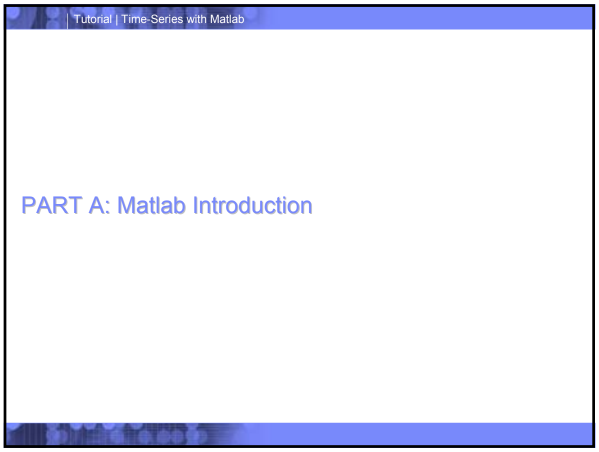


Overview

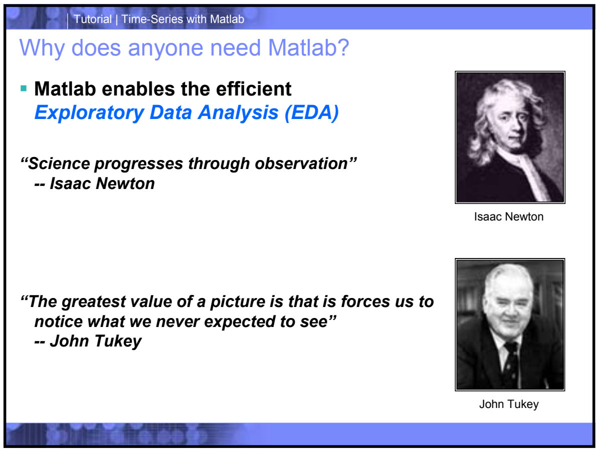
PART A — The Matlab programming environment

PART B — Basic mathematics
- Introduction / geometric intuition
- Coordinates and transforms
- Quantized representations
- Non-Euclidean distances

PART C — Similarity Search and Applications
- □ Introduction
- □ Representations
- □ Distance Measures
- □ Lower Bounding
- □ Clustering/Classification/Visualization
- □ Applications



PART A: Matlab Introduction



Why does anyone need Matlab?

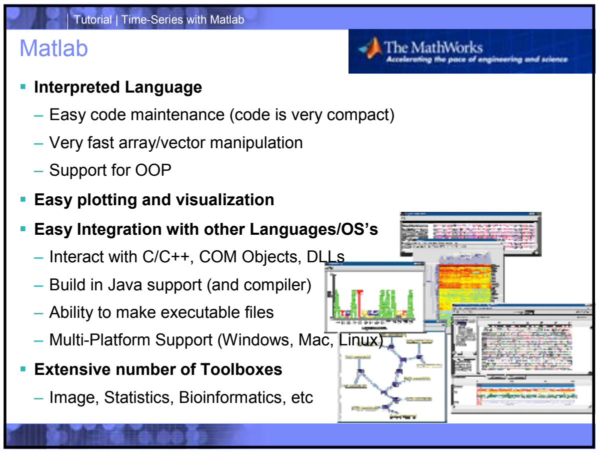
- **Matlab enables the efficient** ***Exploratory Data Analysis (EDA)***

"Science progresses through observation"
-- Isaac Newton

Isaac Newton

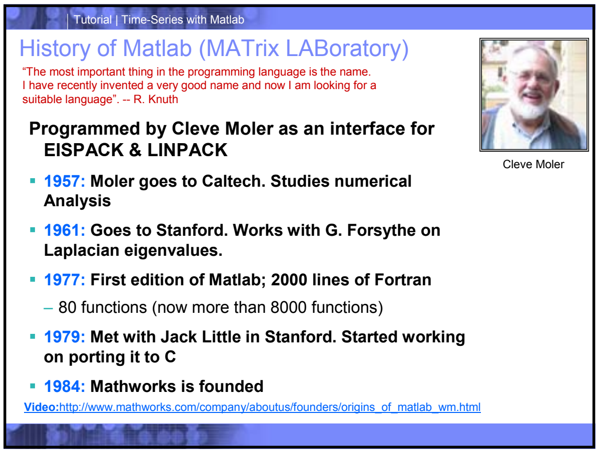
"The greatest value of a picture is that is forces us to notice what we never expected to see"
-- John Tukey

John Tukey



Matlab

- **Interpreted Language**
  - Easy code maintenance (code is very compact)
  - Very fast array/vector manipulation
  - Support for OOP
- **Easy plotting and visualization**
- **Easy Integration with other Languages/OS's**
  - Interact with C/C++, COM Objects, DLLs
  - Build in Java support (and compiler)
  - Ability to make executable files
  - Multi-Platform Support (Windows, Mac, Linux)
- **Extensive number of Toolboxes**
  - Image, Statistics, Bioinformatics, etc



History of Matlab (MATrix LABoratory)

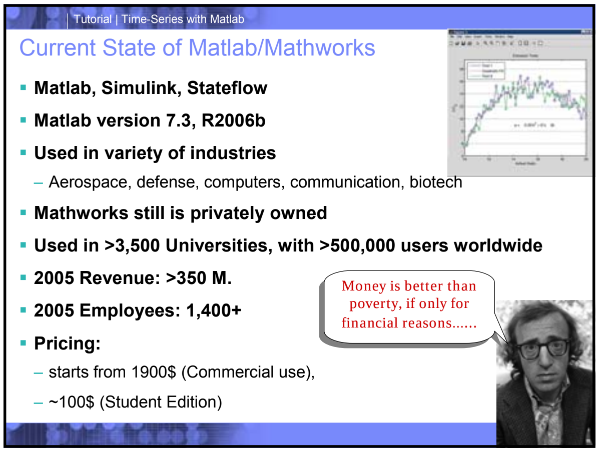
"The most important thing in the programming language is the name.
I have recently invented a very good name and now I am looking for a suitable language". -- R. Knuth

Cleve Moler

Programmed by Cleve Moler as an interface for EISPACK & LINPACK

- **1957: Moler goes to Caltech. Studies numerical Analysis**
- **1961: Goes to Stanford. Works with G. Forsythe on Laplacian eigenvalues.**
- **1977: First edition of Matlab; 2000 lines of Fortran**
  - 80 functions (now more than 8000 functions)
- **1979: Met with Jack Little in Stanford. Started working on porting it to C**
- **1984: Mathworks is founded**

Video:http://www.mathworks.com/company/aboutus/founders/origins_of_matlab_wm.html



Current State of Matlab/Mathworks

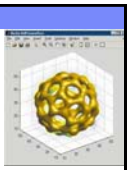
- **Matlab, Simulink, Stateflow**
- **Matlab version 7.3, R2006b**
- **Used in variety of industries**
  - Aerospace, defense, computers, communication, biotech
- **Mathworks still is privately owned**
- **Used in >3,500 Universities, with >500,000 users worldwide**
- **2005 Revenue: >350 M.**
- **2005 Employees: 1,400+**
- **Pricing:**
  - starts from 1900$ (Commercial use),
  - ~100$ (Student Edition)

Money is better than poverty, if only for financial reasons......

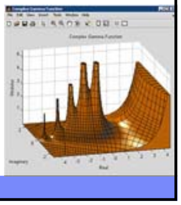
Matlab 7.3

- **R2006b, Released on Sept 1 2006**
  - Distributed computing
  - Better support for large files
  - New optimization Toolbox
  - Matlab builder for Java
    - *create Java classes from Matlab*
  - Demos, Webinars in Flash format
  - (http://www.mathworks.com/products/matlab/demos.html)

Who needs Matlab?



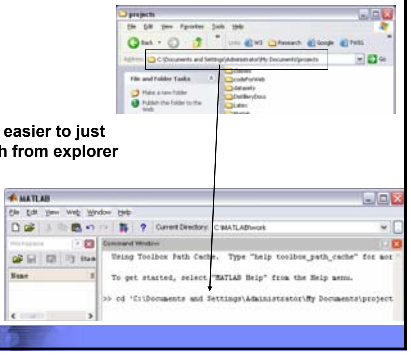
- **R&D companies for easy application deployment**
- **Professors**
  - Lab assignments
  - Matlab allows focus on *algorithms* not on language features
- **Students**
  - Batch processing of files
    - No more incomprehensible perl code!
  - Great environment for testing ideas
    - Quick coding of ideas, then porting to C/Java etc
  - Easy visualization
  - It's cheap! (for students at least…)



Starting up Matlab

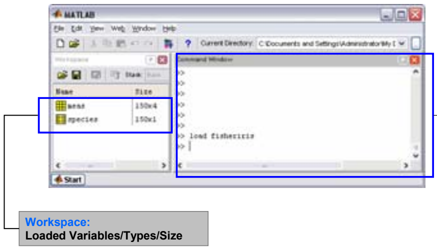
Personally I'm always ready to learn, although I do not always like being taught.
Sir Winston Churchill

- **Dos/Unix like directory navigation**
- **Commands like:**
  - cd
  - pwd
  - mkdir
- **For navigation it is easier to just copy/paste the path from explorer E.g.:**
  cd 'c:\documents\'


projects
Address C:\Documents and Settings\Administrator\My Documents\projects
File and Folder Tasks
MATLAB
Current Directory: C:\MATLAB\work
Command Window

```
Using Toolbox Path Cache.  Type "help toolbox_path_cache" for mor

To get started, select "MATLAB Help" from the Help menu.

>> cd 'C:\Documents and Settings\Administrator\My Documents\project
```

Matlab Environment


MATLAB
File Edit View Web Window Help
Current Directory: C:\Documents and Settings\Administrator\My I
Command Window
Start

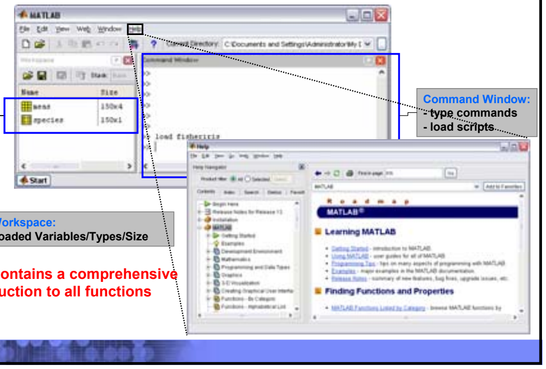
Name	Size
meas	150x4
species	150x1

```
>> load fisheriris
>>
```

Command Window:
- type commands
- load scripts

Workspace:
Loaded Variables/Types/Size

Matlab Environment

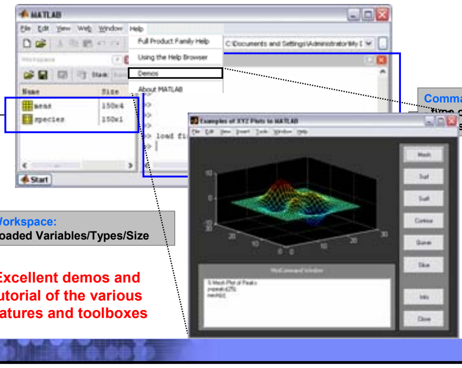

MATLAB
File Edit View Web Window Help
Help
Help Navigator
Learning MATLAB
Finding Functions and Properties

Command Window:
- type commands
- load scripts

Workspace:
Loaded Variables/Types/Size

Help contains a comprehensive introduction to all functions

Matlab Environment



Command Window:
- type commands
- load scripts

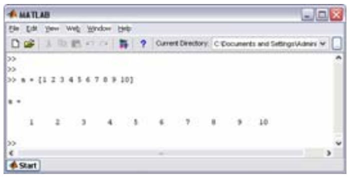
Workspace:
Loaded Variables/Types/Size

Excellent demos and tutorial of the various features and toolboxes

Starting with Matlab

- **Everything is arrays**
- **Manipulation of arrays is faster than regular manipulation with for-loops**

```
a = [1 2 3 4 5 6 7 9 10] % define an array
```

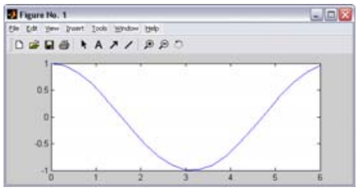

MATLAB
File Edit View Web Window Help
Current Directory: C:\Documents and Settings\Admini
Start

```
>>
>>
>> a = [1 2 3 4 5 6 7 8 9 10]

a =

     1     2     3     4     5     6     7     8     9    10

>>
```

Populating arrays

- **Plot sinusoid function**

```
a = [0:0.3:2*pi]  % generate values from 0 to 2pi (with step of 0.3)
b = cos(a)  % access cos at positions contained in array [a]
plot(a,b)  % plot a (x-axis) against b (y-axis)
```



```
Related:
linspace(-100,100,15); % generate 15 values between -100 and 100
```

Array Access

- **Access array elements**

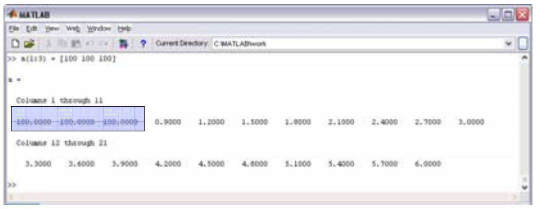
```
>> a(1)
ans =
         0
```

```
>> a(1:3)
ans =
         0    0.3000    0.6000
```

- **Set array elements**

```
>> a(1) = 100
```

```
>> a(1:3) = [100 100 100]
```


MATLAB
File Edit View Web Window Help
Current Directory: C:\MATLAB\work

```
>> a(1:3) = [100 100 100]

a =

  Columns 1 through 11

  100.0000  100.0000  100.0000    0.9000    1.2000    1.5000    1.8000    2.1000    2.4000    2.7000    3.0000

  Columns 12 through 21

    3.3000    3.6000    3.9000    4.2000    4.5000    4.8000    5.1000    5.4000    5.7000    6.0000

>>
```

2D Arrays

- **Can access whole columns or rows**
  - Let's define a 2D array

```
>> a = [1 2 3; 4 5 6]
a =

     1     2     3
     4     5     6

>> a(2,2)

ans =

     5
```

```
>> a(1,:)

ans =

     1     2     3

>> a(:,1)

ans =

     1
     4
```

Row-wise access

Column-wise access

A good listener is not only popular everywhere, but after a while he gets to know something. –Wilson Mizner

Column-wise computation

- **For arrays greater than 1D, all computations happen column-by-column**

```
>> a = [1 2 3; 3 2 1]
a =

     1     2     3
     3     2     1

>> mean(a)

ans =

    2.0000    2.0000    2.0000
```

```
>> max(a)

ans =

     3     2     3

>> sort(a)

ans =

     1     2     1
     3     2     3
```

Concatenating arrays

- **Column-wise or row-wise**

Row next to row

```
>> a = [1 2 3];
>> b = [4 5 6];
>> c = [a b]

c =

     1     2     3     4     5     6
```

Column next to column

```
>> a = [1;2];
>> b = [3;4];
>> c = [a b]
c =

     1     3
     2     4
```

Row below row

```
>> a = [1 2 3];
>> b = [4 5 6];
>> c = [a; b]

c =

     1     2     3
     4     5     6
```

Column below column

```
>> a = [1;2];
>> b = [3;4];
>> c = [a; b]

c =

     1
     2
     3
     4
```

Initializing arrays

- **Create array of ones [ones]**

```
>> a = ones(1,3)
a =

     1     1     1

>> a = ones(1,3)*inf
a =
     Inf  Inf  Inf
```

```
>> a = ones(2,2)*5;
a =

     5     5
     5     5
```

- **Create array of zeroes [zeros]**
  - Good for initializing arrays

```
>> a = zeros(1,4)
a =

     0     0     0     0
```

```
>> a = zeros(3,1) + [1 2 3]'
a =
     1
     2
     3
```

Reshaping and Replicating Arrays

- **Changing the array shape [reshape]**
  - (eg, for easier column-wise computation)

```
>> a = [1 2 3 4 5 6]'; % make it into a column
>> reshape(a,2,3)

ans =

     1     3     5
     2     4     6
```

reshape(X,[M,N]): [M,N] matrix of ***columnwise*** version of X

- **Replicating an array [repmat]**

```
>> a = [1 2 3];
>> repmat(a,1,2)

ans =    1     2     3     1     2     3

>> repmat(a,2,1)
ans =
         1     2     3
         1     2     3
```

repmat(X,[M,N]): make [M,N] tiles of X

Useful Array functions

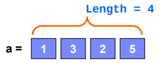
- **Last element of array [end]**

```
>> a = [1 3 2 5];
>> a(end)

ans =

     5
```

```
>> a = [1 3 2 5];
>> a(end-1)

ans =

     2
```

- **Length of array [length]**

```
>> length(a)

ans =

     4
```

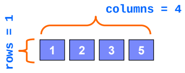


Length = 4

a = 1 3 2 5

- **Dimensions of array [size]**

```
>> [rows, columns] = size(a)
rows = 1

columns = 4
```



columns = 4

rows = 1

1 2 3 5

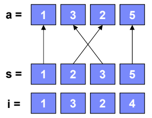
Useful Array functions

- **Find a specific element [find] ****

```
>> a = [1 3 2 5 10 5 2 3];
>> b = find(a==2)

b =

     3     7
```

- **Sorting [sort] *****

```
>> a = [1 3 2 5];
>> [s,i]=sort(a)

s =
     1     2     3     5

i =
     1     3     2     4
```



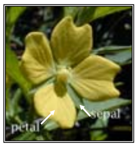
a = 1 3 2 5

s = 1 2 3 5

i = 1 3 2 4

Indicates the index where the element came from

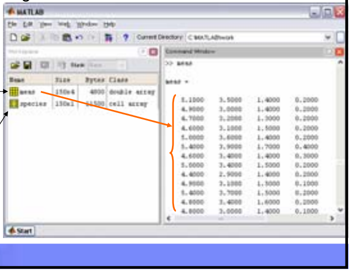
Visualizing Data and Exporting Figures



- **Use Fisher's Iris dataset**

```
>> load fisheriris
```

  - 4 dimensions, 3 species
  - Petal length & width, sepal length & width
  - Iris:
    - virginica/versicolor/setosa



MATLAB
File Edit View Web Window Help
Current Directory: C:\MATLAB\work
Workspace
Command Window
Start

Name	Size	Bytes	Class
meas	150x4	4800	double array
species	150x1	[illegible]	cell array

```
>> meas

meas =

    5.1000    3.5000    1.4000    0.2000
    4.9000    3.0000    1.4000    0.2000
    4.7000    3.2000    1.3000    0.2000
    4.6000    3.1000    1.5000    0.2000
    5.0000    3.6000    1.4000    0.2000
    5.4000    3.9000    1.7000    0.4000
    4.6000    3.4000    1.4000    0.3000
    5.0000    3.4000    1.5000    0.2000
    4.4000    2.9000    1.4000    0.2000
    4.9000    3.1000    1.5000    0.1000
    5.4000    3.7000    1.5000    0.2000
    4.8000    3.4000    1.6000    0.2000
    4.8000    3.0000    1.4000    0.1000
```

meas (150x4 array): Holds 4D measurements

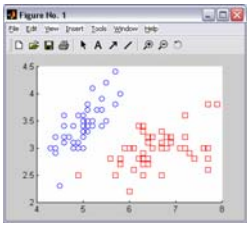
```
 . . .
'versicolor'
'versicolor'
'versicolor'
'versicolor'
'versicolor'
'virginica'
'virginica'
'virginica'
'virginica'
 . . .
```

species (150x1 cell array): Holds name of species for the specific measurement

strcmp, scatter, hold on

Visualizing Data (2D)

```
>> idx_setosa = strcmp(species, 'setosa'); % rows of setosa data
>> idx_virginica = strcmp(species, 'virginica'); % rows of virginica
>>
>> setosa = meas(idx_setosa,[1:2]);
>> virgin = meas(idx_virginica,[1:2]);
>> scatter(setosa(:,1), setosa(:,2)); % plot in blue circles by default
>> hold on;
>> scatter(virgin(:,1), virgin(:,2), 'rs');  % red[r] squares[s] for these
```



idx_setosa

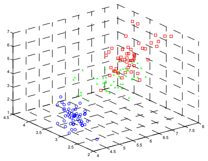
```
 . . .
 1
 1
 1
 0
 0
 0
 . . .
```

An array of zeros and ones indicating the positions where the keyword 'setosa' was found

The world is governed more by appearances rather than realities… --Daniel Webster scatter3

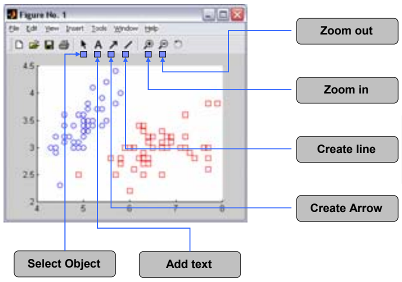
Visualizing Data (3D)

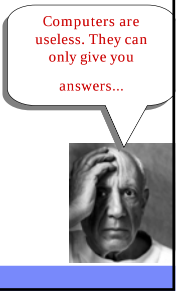
```
>> idx_setosa = strcmp(species, ‘setosa’); % rows of setosa data
>> idx_virginica = strcmp(species, ‘virginica’); % rows of virginica
>> idx_versicolor = strcmp(species, ‘versicolor’); % rows of versicolor

>> setosa = meas(idx_setosa,[1:3]);
>> virgin = meas(idx_virginica,[1:3]);
>> versi = meas(idx_versicolor,[1:3]);
>> scatter3(setosa(:,1), setosa(:,2),setosa(:,3)); % plot in blue circles by default
>> hold on;
>> scatter3(virgin(:,1), virgin(:,2),virgin(:,3), ‘rs’); % red[r] squares[s] for these
>> scatter3(versi(:,1), virgin(:,2),versi(:,3), ‘gx’); % green x’s
```



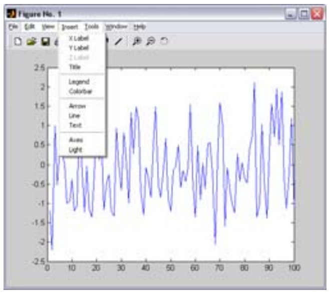
```
>> grid on; % show grid on axis
>> rotate3D on; % rotate with mouse
```

Changing Plots Visually



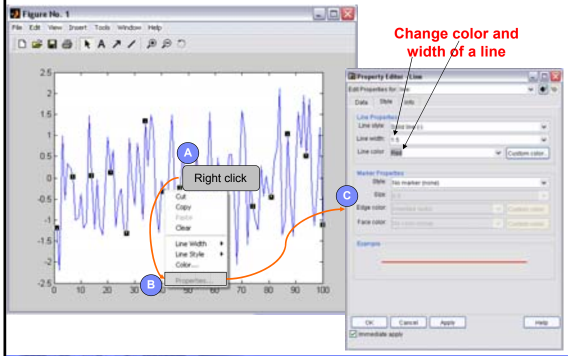


Changing Plots Visually

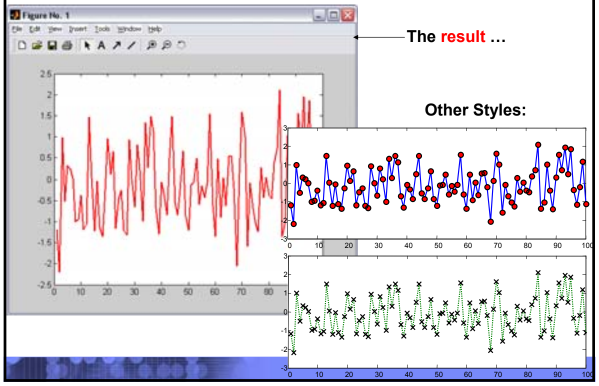


- *Add titles*
- *Add labels on axis*
- *Change tick labels*
- *Add grids to axis*
- *Change color of line*
- *Change thickness/ Linestyle*
- *etc*

Changing Plots Visually (Example)



Changing Plots Visually (Example)



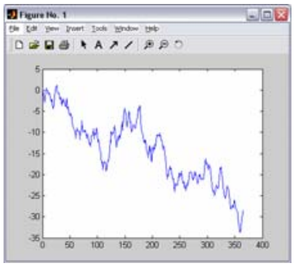
The **result** …

Other Styles:

Changing Figure Properties with Code

- **GUI’s are easy, but sooner or later we realize that coding is faster**

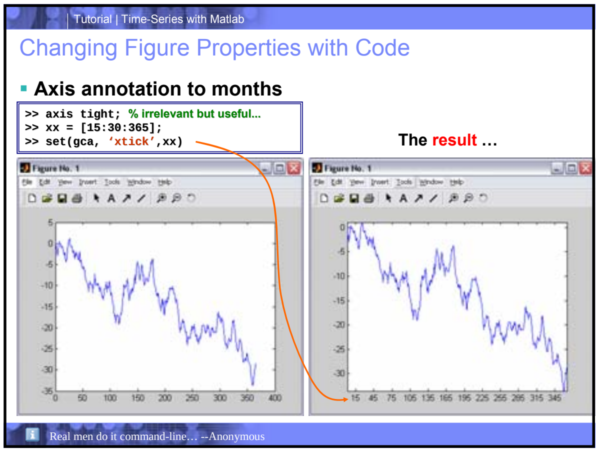
```
>> a = cumsum(randn(365,1)); % random walk of 365 values
```



If this represents a year’s worth of measurements of an imaginary quantity, we will change:

- **x-axis annotation to months**
- **Axis labels**
- **Put title in the figure**
- **Include some greek letters in the title *just for fun***

Real men do it command-line… --Anonymous


Tutorial | Time-Series with Matlab

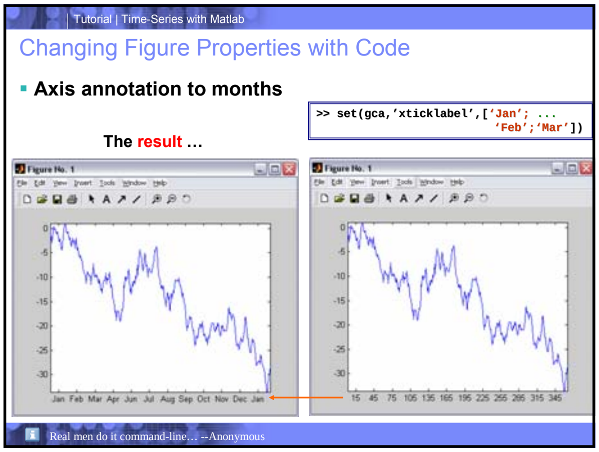
Changing Figure Properties with Code

- **Axis annotation to months**

```
>> axis tight; % irrelevant but useful...
>> xx = [15:30:365];
>> set(gca, 'xtick',xx)
```

The result …

Real men do it command-line… --Anonymous


Tutorial | Time-Series with Matlab

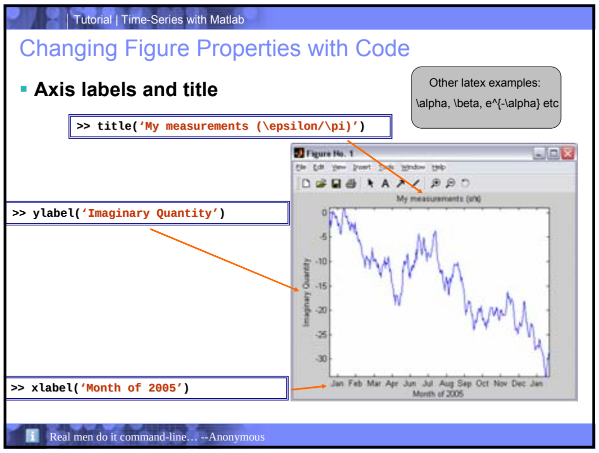
Changing Figure Properties with Code

- **Axis annotation to months**

```
>> set(gca,'xticklabel',['Jan'; ...
                         'Feb';'Mar'])
```

The result …

Real men do it command-line… --Anonymous


Tutorial | Time-Series with Matlab

Changing Figure Properties with Code

- **Axis labels and title**

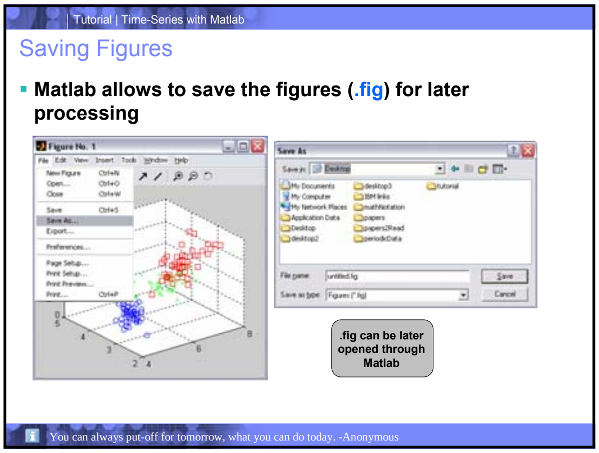
Other latex examples:
\alpha, \beta, e^{-\alpha} etc

```
>> title('My measurements (\epsilon/\pi)')
```

```
>> ylabel('Imaginary Quantity')
```

```
>> xlabel('Month of 2005')
```

Real men do it command-line… --Anonymous

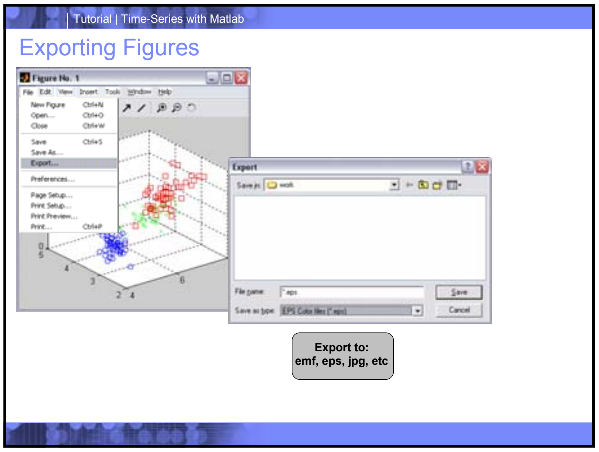

Tutorial | Time-Series with Matlab

Saving Figures

- **Matlab allows to save the figures (.fig) for later processing**

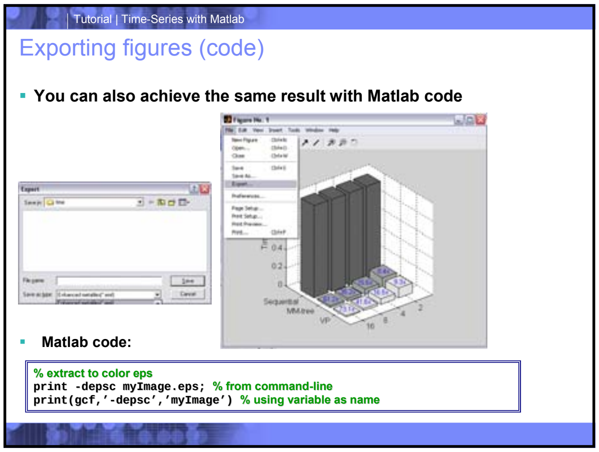
.fig can be later opened through Matlab

You can always put-off for tomorrow, what you can do today. -Anonymous


Tutorial | Time-Series with Matlab

Exporting Figures

Export to: emf, eps, jpg, etc

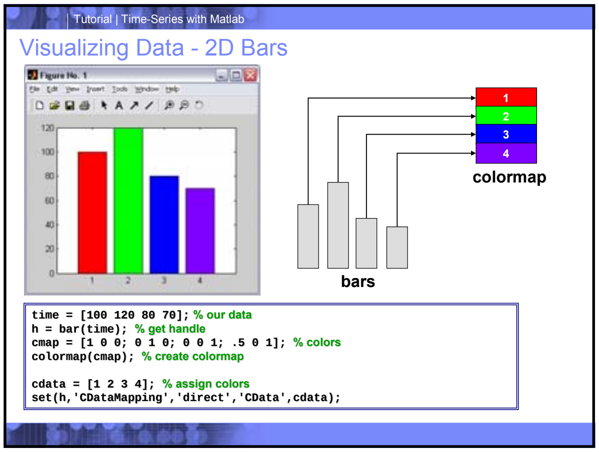

Tutorial | Time-Series with Matlab

Exporting figures (code)

- **You can also achieve the same result with Matlab code**

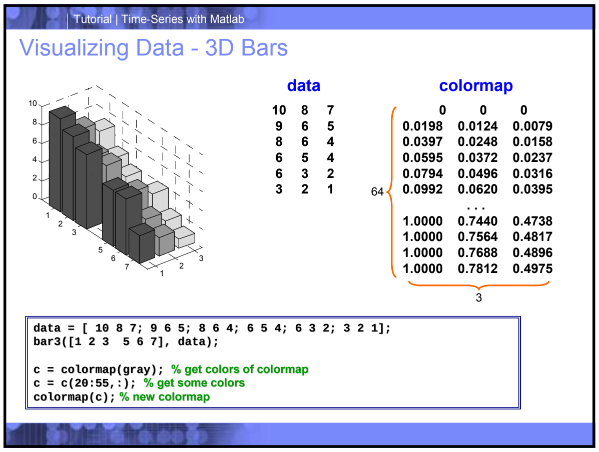
- **Matlab code:**

```
% extract to color eps
print -depsc myImage.eps; % from command-line
print(gcf,'-depsc','myImage') % using variable as name
```



Visualizing Data - 2D Bars

```
time = [100 120 80 70]; % our data
h = bar(time);  % get handle
cmap = [1 0 0; 0 1 0; 0 0 1; .5 0 1];  % colors
colormap(cmap);  % create colormap

cdata = [1 2 3 4];  % assign colors
set(h,'CDataMapping','direct','CData',cdata);
```



Visualizing Data - 3D Bars

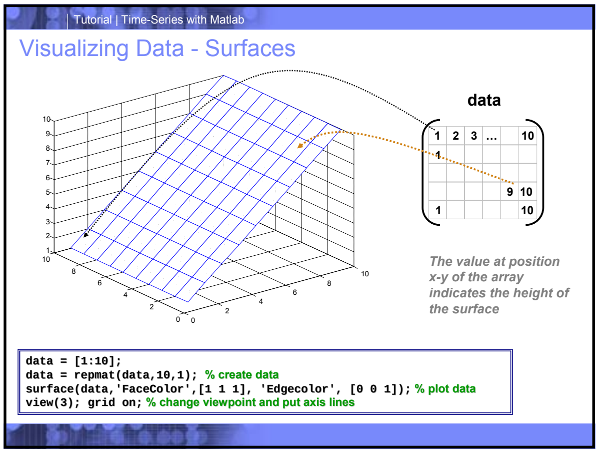
data

```
10   8   7
 9   6   5
 8   6   4
 6   5   4
 6   3   2
 3   2   1
```

colormap

```
         0        0        0
    0.0198   0.0124   0.0079
    0.0397   0.0248   0.0158
    0.0595   0.0372   0.0237
64  0.0794   0.0496   0.0316
    0.0992   0.0620   0.0395
              . . .
    1.0000   0.7440   0.4738
    1.0000   0.7564   0.4817
    1.0000   0.7688   0.4896
    1.0000   0.7812   0.4975
                3
```

```
data = [ 10 8 7; 9 6 5; 8 6 4; 6 5 4; 6 3 2; 3 2 1];
bar3([1 2 3  5 6 7], data);

c = colormap(gray);  % get colors of colormap
c = c(20:55,:);  % get some colors
colormap(c); % new colormap
```



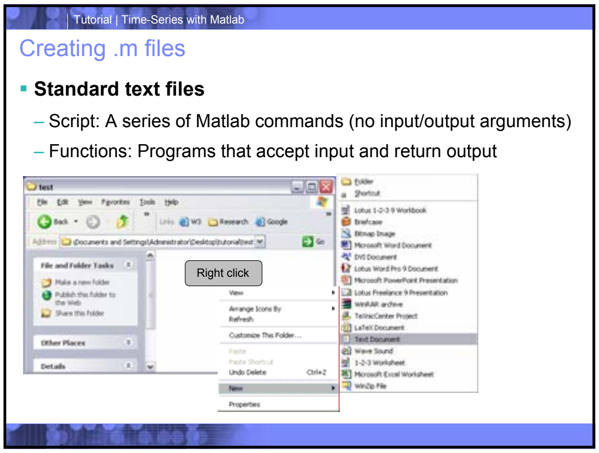
Visualizing Data - Surfaces

data

1	2	3	…	10
1				
			9	10
1				10

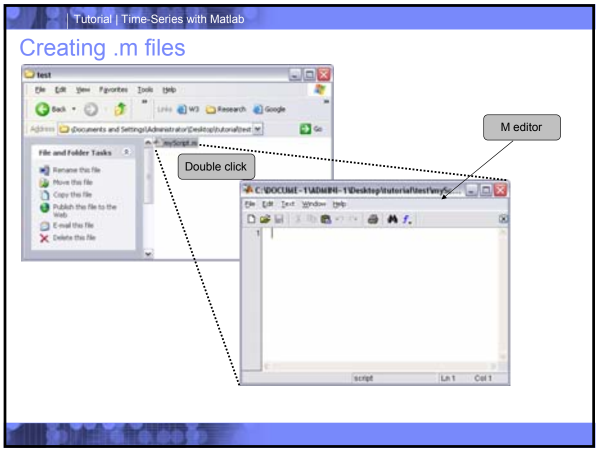
The value at position x-y of the array indicates the height of the surface

```
data = [1:10];
data = repmat(data,10,1);  % create data
surface(data,'FaceColor',[1 1 1], 'Edgecolor', [0 0 1]); % plot data
view(3); grid on; % change viewpoint and put axis lines
```



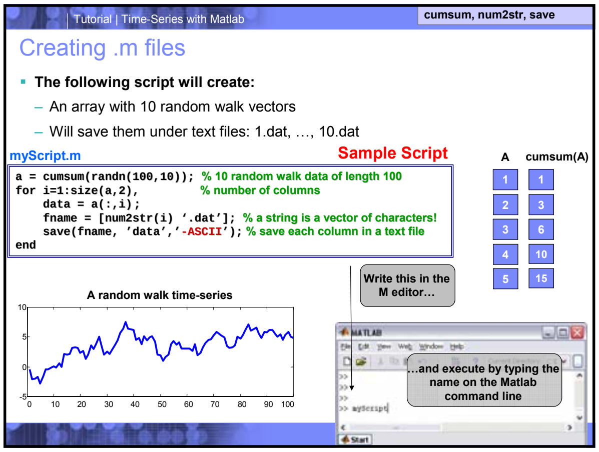
Creating .m files

- **Standard text files**
  - Script: A series of Matlab commands (no input/output arguments)
  - Functions: Programs that accept input and return output



Creating .m files

test
File Edit View Favorites Tools Help
Back
Links W3 Research Google
Address C:\Documents and Settings\Administrator\Desktop\tutorial\test
Go
File and Folder Tasks
Rename this file
Move this file
Copy this file
Publish this file to the Web
E-mail this file
Delete this file
myScript.m
Double click
M editor
C:\DOCUME~1\ADMINI~1\Desktop\tutorial\test\myS...
File Edit Text Window Help
1
script
Ln 1
Col 1



cumsum, num2str, save

Creating .m files

- **The following script will create:**
  - An array with 10 random walk vectors
  - Will save them under text files: 1.dat, …, 10.dat

myScript.m

Sample Script

```
a = cumsum(randn(100,10));  % 10 random walk data of length 100
for i=1:size(a,2),          % number of columns
    data = a(:,i);
    fname = [num2str(i) '.dat']; % a string is a vector of characters!
    save(fname, 'data','-ASCII'); % save each column in a text file
end
```

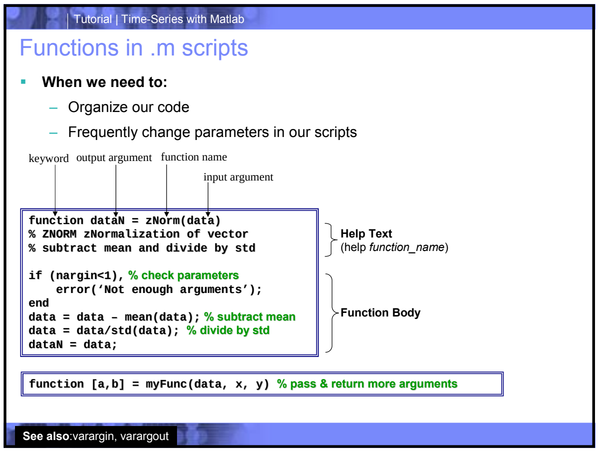
A	cumsum(A)
1	1
2	3
3	6
4	10
5	15

A random walk time-series

Write this in the M editor…

…and execute by typing the name on the Matlab command line

```
>>
>>
>>
>> myScript
```



Functions in .m scripts

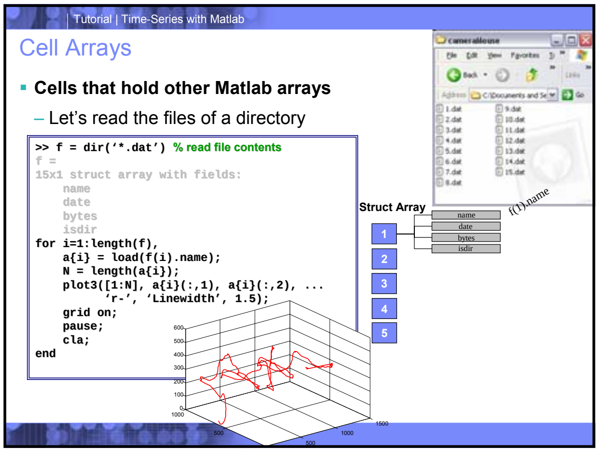
- **When we need to:**
  - Organize our code
  - Frequently change parameters in our scripts

keyword
output argument
function name
input argument
Help Text
(help *function_name*)
Function Body

```
function dataN = zNorm(data)
% ZNORM zNormalization of vector
% subtract mean and divide by std

if (nargin<1), % check parameters
    error('Not enough arguments');
end
data = data – mean(data); % subtract mean
data = data/std(data);  % divide by std
dataN = data;
```

```
function [a,b] = myFunc(data, x, y)  % pass & return more arguments
```

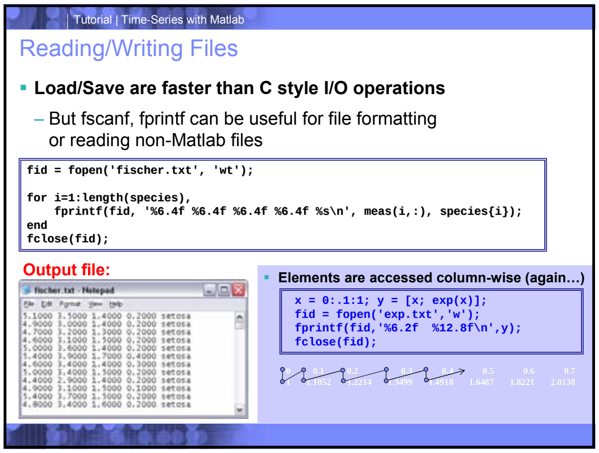
See also:varargin, varargout



Cell Arrays

- **Cells that hold other Matlab arrays**
  - Let's read the files of a directory

```
>> f = dir('*.dat') % read file contents
f =
15x1 struct array with fields:
    name
    date
    bytes
    isdir
for i=1:length(f),
    a{i} = load(f(i).name);
    N = length(a{i});
    plot3([1:N], a{i}(:,1), a{i}(:,2), ...
        'r-', 'Linewidth', 1.5);
    grid on;
    pause;
    cla;
end
```



Reading/Writing Files

- **Load/Save are faster than C style I/O operations**
  - But fscanf, fprintf can be useful for file formatting or reading non-Matlab files

```
fid = fopen('fischer.txt', 'wt');

for i=1:length(species),
    fprintf(fid, '%6.4f %6.4f %6.4f %6.4f %s\n', meas(i,:), species{i});
end
fclose(fid);
```

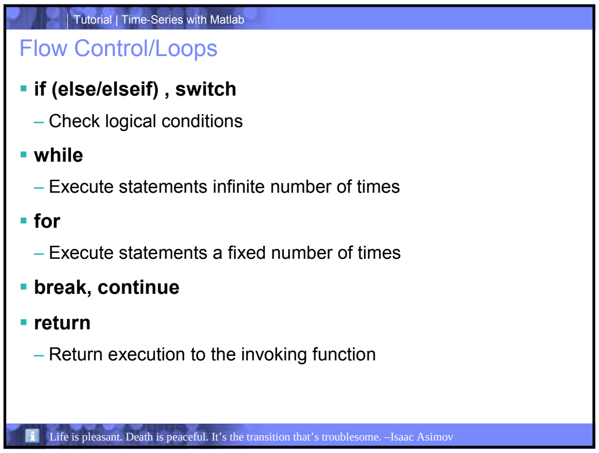
Output file:

```
5.1000 3.5000 1.4000 0.2000 setosa
4.9000 3.0000 1.4000 0.2000 setosa
4.7000 3.2000 1.3000 0.2000 setosa
4.6000 3.1000 1.5000 0.2000 setosa
5.0000 3.6000 1.4000 0.2000 setosa
5.4000 3.9000 1.7000 0.4000 setosa
4.6000 3.4000 1.4000 0.3000 setosa
5.0000 3.4000 1.5000 0.2000 setosa
4.4000 2.9000 1.4000 0.2000 setosa
4.9000 3.1000 1.5000 0.1000 setosa
5.4000 3.7000 1.5000 0.2000 setosa
4.8000 3.4000 1.6000 0.2000 setosa
```

- **Elements are accessed column-wise (again…)**

```
x = 0:.1:1; y = [x; exp(x)];
fid = fopen('exp.txt','w');
fprintf(fid,'%6.2f  %12.8f\n',y);
fclose(fid);
```

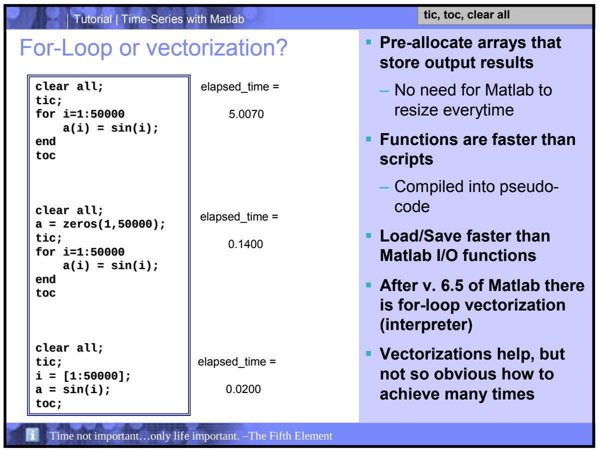
```
0       0.1     0.2     0.3     0.4     0.5     0.6     0.7
1       1.1052  1.2214  1.3499  1.4918  1.6487  1.8221  2.0138
```



Flow Control/Loops

- **if (else/elseif) , switch**
  - Check logical conditions
- **while**
  - Execute statements infinite number of times
- **for**
  - Execute statements a fixed number of times
- **break, continue**
- **return**
  - Return execution to the invoking function

Life is pleasant. Death is peaceful. It's the transition that's troublesome. –Isaac Asimov



tic, toc, clear all

For-Loop or vectorization?

```
clear all;
tic;
for i=1:50000
    a(i) = sin(i);
end
toc
```

elapsed_time =

5.0070

```
clear all;
a = zeros(1,50000);
tic;
for i=1:50000
    a(i) = sin(i);
end
toc
```

elapsed_time =

0.1400

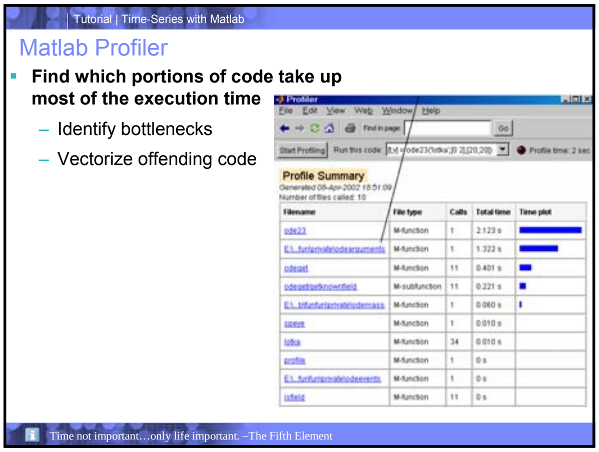
```
clear all;
tic;
i = [1:50000];
a = sin(i);
toc;
```

elapsed_time =

0.0200

- **Pre-allocate arrays that store output results**
  - No need for Matlab to resize everytime
- **Functions are faster than scripts**
  - Compiled into pseudo-code
- **Load/Save faster than Matlab I/O functions**
- **After v. 6.5 of Matlab there is for-loop vectorization (interpreter)**
- **Vectorizations help, but not so obvious how to achieve many times**

Time not important…only life important. –The Fifth Element



Matlab Profiler

- **Find which portions of code take up most of the execution time**
  - Identify bottlenecks
  - Vectorize offending code

[t,y] = ode23('lotka',[0 2],[20;20])

Profile Summary

Generated 08-Apr-2002 18:51:09

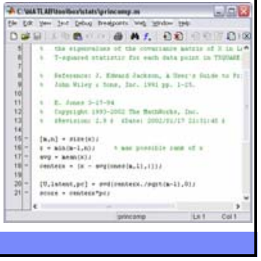
Number of files called: 10

Filename	File type	Calls	Total time	Time plot
ode23	M-function	1	2.123 s	▬▬▬▬▬
E:\...fun\private\odearguments	M-function	1	1.322 s	▬▬▬
odeget	M-function	11	0.401 s	▬
odeget/getknownfield	M-subfunction	11	0.221 s	■
E:\...b\funfun\private\odemass	M-function	1	0.060 s	▮
speye	M-function	1	0.010 s	
lotka	M-function	34	0.010 s	
profile	M-function	1	0 s	
E:\...funfun\private\odeevents	M-function	1	0 s	
isfield	M-function	11	0 s	

Time not important…only life important. –The Fifth Element

Hints &Tips

- **There is always an easier (and faster) way**
  - Typically there is a specialized function for what you want to achieve
- **Learn vectorization techniques, by 'peaking' at the actual Matlab files:**
  - edit [fname], eg
  - edit mean
  - edit princomp
- **Matlab Help contains many vectorization examples**

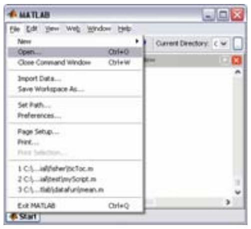

C:\MATLAB\toolbox\stats\princomp.m
File Edit View Text Debug Breakpoints Web Window Help
princomp
Ln 1
Col 1

```
%   the eigenvalues of the covariance matrix of X in L
%   T-squared statistic for each data point in TSQUARE

%   Reference: J. Edward Jackson, A User's Guide to Pr
%   John Wiley & Sons, Inc. 1991 pp. 1-25.

%   B. Jones 3-17-94
%   Copyright 1993-2002 The MathWorks, Inc.
%   $Revision: 2.9 $  $Date: 2002/01/17 21:31:45 $

[m,n] = size(x);
r = min(m-1,n);     % max possible rank of x
avg = mean(x);
centerx = (x - avg(ones(m,1),:));

[U,latent,pc] = svd(centerx./sqrt(m-1),0);
score = centerx*pc;
```

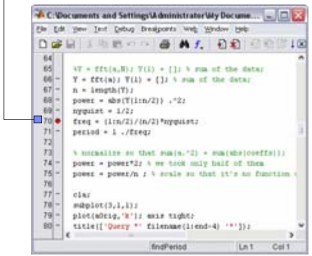
Debugging

Beware of bugs in the above code; I have only proved it correct, not tried it
-- R. Knuth

- **Not as frequently required as in C/C++**
  - Set breakpoints, step, step in, check variables values



Set breakpoints

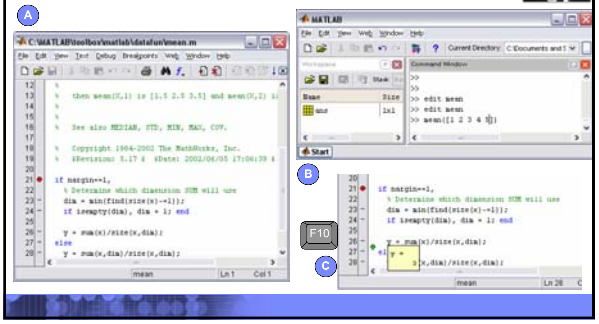

C:\Documents and Settings\Administrator\My Docume...
File Edit View Text Debug Breakpoints Web Window Help
findPeriod
Ln 1
Col 1

```
%Y = fft(a,N); Y(1) = []; % sum of the data;
Y = fft(a); Y(1) = []; % sum of the data;
n = length(Y);
power = abs(Y(1:n/2)) .^2;
nyquist = 1/2;
freq = (1:n/2)/(n/2)*nyquist;
period = 1 ./freq;

% normalize so that sum(a.^2) = sum(abs(coeffs));
power = power*2; % we took only half of them
power = power/n ; % scale so that it's an function o

clf;
subplot(3,1,1);
plot(aOrig,'k'); axis tight;
title(['Query "' filename(1:end-4) '"']);
```

Debugging

Either this man is dead or my watch has stopped.

- **Full control over variables and execution path**
  - F10: step, F11: step in (visit functions, as well)


A
C:\MATLAB\toolbox\matlab\datafun\mean.m
File Edit View Text Debug Breakpoints Web Window Help
mean
Ln 1
Col 1
MATLAB
File Edit View Web Window Help
Current Directory: C:\Documents and S
Workspace
Command Window
Name
Size
ans
1x1
Start
B
F10
C
mean
Ln 28

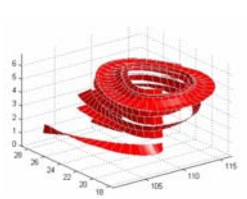
```
%
%   then mean(X,1) is [1.5 2.5 3.5] and mean(X,2) is
%
%
%   See also MEDIAN, STD, MIN, MAX, COV.

%   Copyright 1984-2002 The MathWorks, Inc.
%   $Revision: 5.17 $  $Date: 2002/06/05 17:06:39 $

if nargin==1,
  % Determine which dimension SUM will use
  dim = min(find(size(x)~=1));
  if isempty(dim), dim = 1; end

  y = sum(x)/size(x,dim);
else
  y = sum(x,dim)/size(x,dim);
```

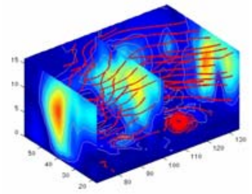
```
>>
>>
>> edit mean
>> edit mean
>> mean([1 2 3 4 5])
```

```
if nargin==1,
  % Determine which dimension SUM will use
  dim = min(find(size(x)~=1));
  if isempty(dim), dim = 1; end

  y = sum(x)/size(x,dim);
el  y =
       3 (x,dim)/size(x,dim);
```

Advanced Features – 3D modeling/Volume Rendering

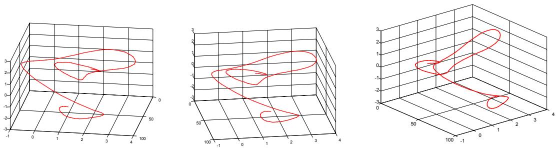
- **Very easy volume manipulation and rendering**





Advanced Features – Making Animations (Example)

- **Create animation by changing the camera viewpoint**

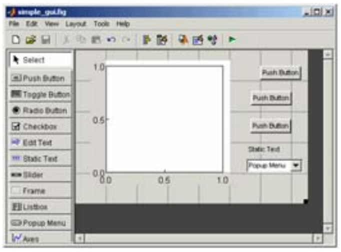


```
azimuth = [50:100 99:-1:50]; % azimuth range of values
for k = 1:length(azimuth),
    plot3(1:length(a), a(:,1), a(:,2), 'r', 'Linewidth',2);
    grid on;
    view(azimuth(k),30); % change new
    M(k) = getframe; % save the frame
end

movie(M,20); % play movie 20 times
```

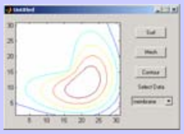
See also:movie2avi

Advanced Features – GUI's

- **Built-in Development Environment**
  - Buttons, figures, Menus, sliders, etc



- **Several Examples in Help**
  - Directory listing
  - Address book reader
  - GUI with multiple axis

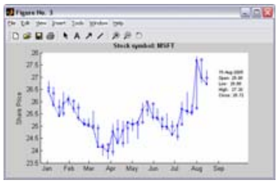


Advanced Features – Using Java

- **Matlab is shipped with Java Virtual Machine (JVM)**
- **Access Java API (eg I/O or networking)**
- **Import Java classes and construct objects**
- **Pass data between Java objects and Matlab variables**

Advanced Features – Using Java (Example)

- **Stock Quote Query**
  - Connect to Yahoo server
  - http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=4069&objectType=file



```
disp('Contacting YAHOO server using ...');
    disp(['url = java.net.URL(' urlString ')']);
end;
url = java.net.URL(urlString);

try
    stream = openStream(url);
    ireader = java.io.InputStreamReader(stream);
    breader = java.io.BufferedReader(ireader);
    connect_query_data= 1; %connect made;
catch
    connect_query_data= -1;  %could not connect
case;
    disp(['URL: ' urlString]);
    error(['Could not connect to server. It may
be unavailable. Try again later.']);
    stockdata={};
    return;
end
```

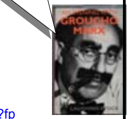
Matlab Toolboxes

- **You can buy many specialized toolboxes from Mathworks**
  - Image Processing, Statistics, Bio-Informatics, etc
- **There are many equivalent *free* toolboxes too:**
  - SVM toolbox
    - **http://theoval.sys.uea.ac.uk/~gcc/svm/toolbox/**
  - Wavelets
    - **http://www.math.rutgers.edu/~ojanen/wavekit/**
  - Speech Processing
    - **http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html**
  - Bayesian Networks
    - **http://www.cs.ubc.ca/~murphyk/Software/BNT/bnt.html**

In case I get stuck…

- **help [command] (on the command line)**
  eg. `help fft`
- **Menu: help -> matlab help**
  - Excellent introduction on various topics
- **Matlab webinars**
  - http://www.mathworks.com/company/events/archived_webinars.html?fp
- **Google groups**
  - **comp.soft-sys.matlab**
  - **You can find *anything* here**
  - **Someone else had the same problem before you!**

I've had a wonderful evening. But this wasn't it…



PART B: Mathematical notions

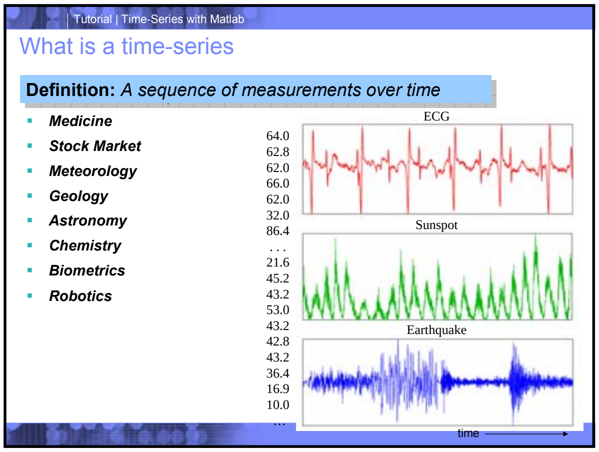
Eight percent of success is showing up.

Overview of Part B

1. **Introduction and geometric intuition**

➡ **Coordinates and transforms**
   - Fourier transform (DFT)
   - Wavelet transform (DWT)
   - Incremental DWT
   - Principal components (PCA)
   - Incremental PCA

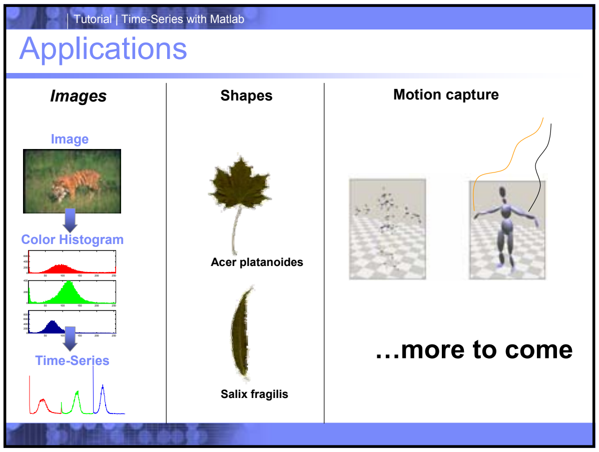
3. **Quantized representations**
   - Piecewise quantized / symbolic
   - Vector quantization (VQ) / K-means
4. **Non-Euclidean distances**
   - Dynamic time warping (DTW)



What is a time-series

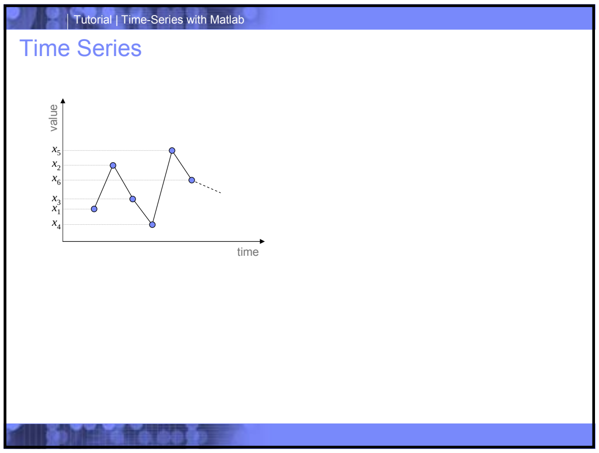
Definition: *A sequence of measurements over time*

- ***Medicine***
- ***Stock Market***
- ***Meteorology***
- ***Geology***
- ***Astronomy***
- ***Chemistry***
- ***Biometrics***
- ***Robotics***

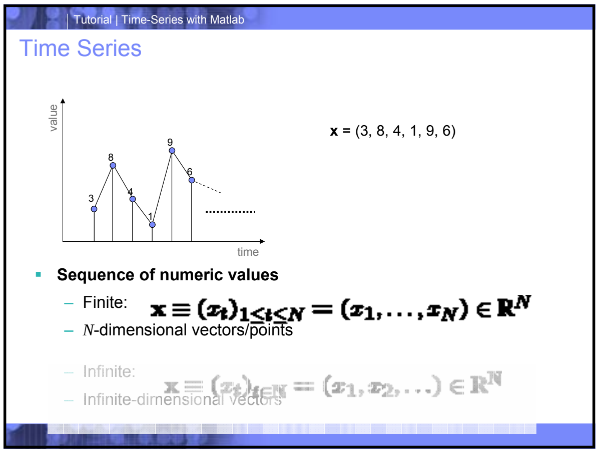


Applications

…more to come



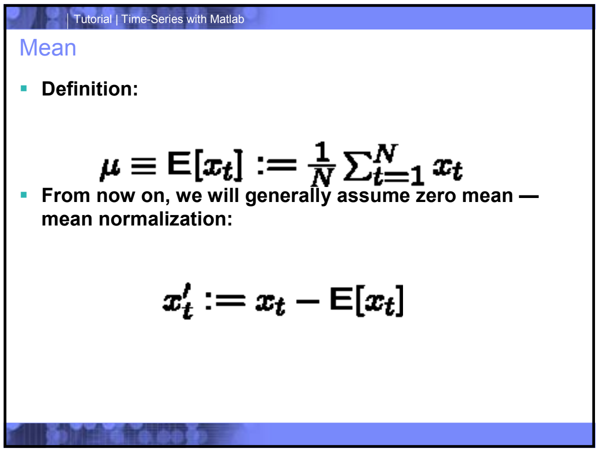
Time Series



Time Series

x = (3, 8, 4, 1, 9, 6)

- **Sequence of numeric values**
  - Finite: $\mathbf{x} \equiv (x_t)_{1 \le t \le N} = (x_1, \ldots, x_N) \in \mathbb{R}^N$
  - N-dimensional vectors/points
  - Infinite: $\mathbf{x} \equiv (x_t)_{t \in \mathbb{N}} = (x_1, x_2, \ldots) \in \mathbb{R}^{\mathbb{N}}$
  - Infinite-dimensional vectors



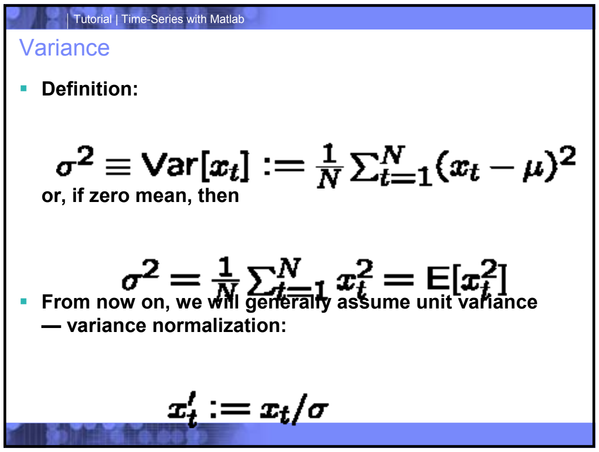
Mean

- **Definition:**

$$\mu \equiv \mathrm{E}[x_t] := \frac{1}{N}\sum_{t=1}^{N} x_t$$

- **From now on, we will generally assume zero mean — mean normalization:**

$$x'_t := x_t - \mathrm{E}[x_t]$$



Variance

- **Definition:**

$$\sigma^2 \equiv \mathrm{Var}[x_t] := \frac{1}{N}\sum_{t=1}^{N}(x_t - \mu)^2$$

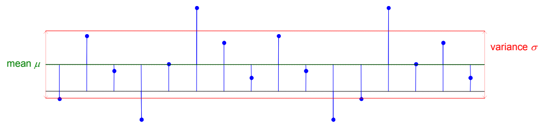
or, if zero mean, then

$$\sigma^2 = \frac{1}{N}\sum_{t=1}^{N} x_t^2 = \mathrm{E}[x_t^2]$$

- **From now on, we will generally assume unit variance — variance normalization:**

$$x'_t := x_t / \sigma$$

Mean and variance



Why and when to normalize

- **Intuitively, the notion of "shape" is *generally* independent of**
  - Average level (mean)
  - Magnitude (variance)
- **Unless otherwise specified, we normalize to zero mean and unit variance**

Variance "=" Length

- **Variance of zero-mean series:**

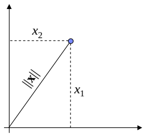
$$\sigma_x^2 = \frac{1}{N}\sum_t x_t^2$$

- **Length of N-dimensional vector (L2-norm):**

$$\|\mathbf{x}\| = \sqrt{\sum_i x_i^2}$$

- **So that:**

$$\boxed{\sigma_x\sqrt{N} = \|\mathbf{x}\|}$$



Covariance and correlation

- **Definition**

$$\mathrm{Cov}[x_t, y_t] := \frac{1}{N}\sum_{t=1}^{N}(x_t - \mu_x)(y_t - \mu_y)$$

$$\rho := \frac{\sum_{t=1}^{N}(x_t - \mu_x)(y_t - \mu_y)}{N\sigma_x\sigma_y}$$

or, if zero mean and unit variance, then

$$\rho = \mathrm{Cov}[x_t, y_t] = \frac{1}{N}\sum_{t=1}^{N} x_t y_t$$

Correlation and similarity

- **How "strong" is the linear relationship**

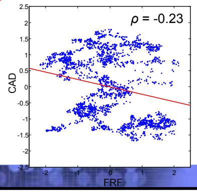
$$y_t = \alpha x_t + \epsilon_t, \quad \text{for } 1 \leq t \leq N$$

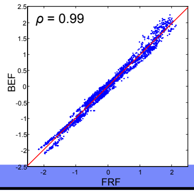
  between x_t and y_t ?
- **For normalized series,**

residual

$$\alpha = \rho_{x,y} \quad \text{and} \quad \sum_t \epsilon_t^2 / N = 1 - \rho_{x,y}^2$$

slope





Correlation "=" Angle

- **Correlation of normalized series:**

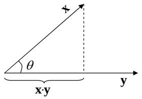
$$\rho_{x,y} = \frac{1}{N}\sum_t x_t y_t$$

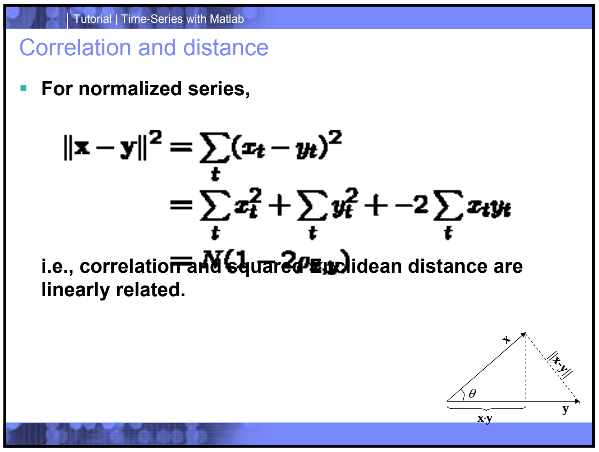
- **Cosine law:**

$$\sum_i x_i y_i = \mathbf{x} \cdot \mathbf{y} = \|\mathbf{x}\|\,\|\mathbf{y}\| \cos\theta_{x,y}$$

- **So that:**

$$\boxed{\rho_{x,y} = \cos\theta_{x,y}}$$



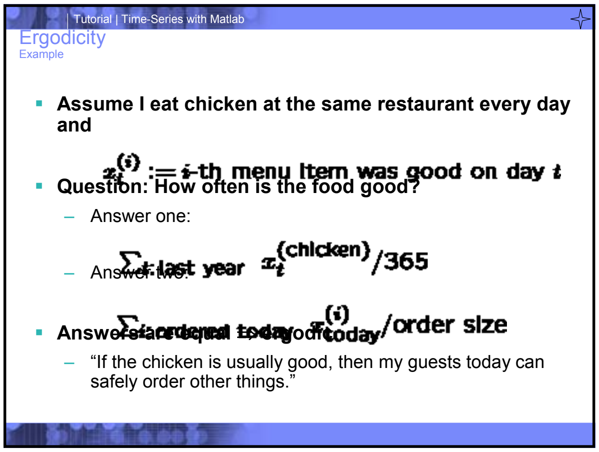


Correlation and distance

- **For normalized series,**

$$\|\mathbf{x}-\mathbf{y}\|^2 = \sum_t (x_t - y_t)^2$$
$$= \sum_t x_t^2 + \sum_t y_t^2 + -2\sum_t x_t y_t$$
$$= N(1 - 2\rho_{\mathbf{x},\mathbf{y}})$$

i.e., correlation and squared Euclidean distance are linearly related.



Ergodicity
Example

- **Assume I eat chicken at the same restaurant every day and**

$$x_t^{(i)} := i\text{-th menu item was good on day } t$$

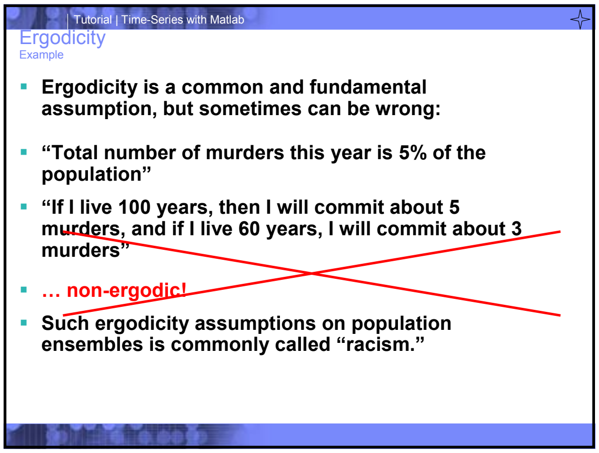
- **Question: How often is the food good?**
  - Answer one:

$$\textstyle\sum_{t:\text{ last year}} x_t^{(\text{chicken})}/365$$

  - Answer two:

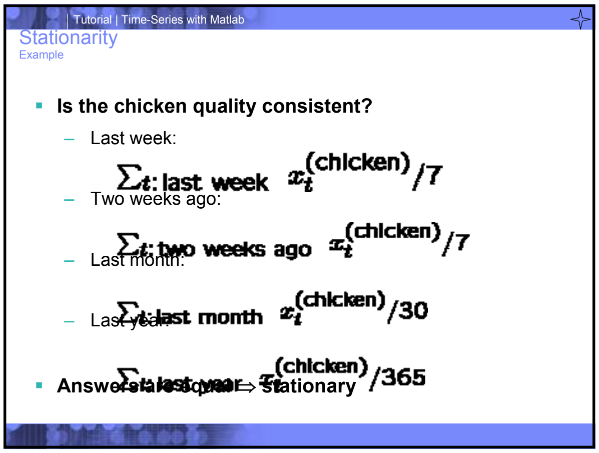
$$\textstyle\sum_{i:\text{ ordered today}} x_{\text{today}}^{(i)}/\text{order size}$$

- **Answers are equal ⇒ ergodic**
  - "If the chicken is usually good, then my guests today can safely order other things."



Ergodicity
Example

- **Ergodicity is a common and fundamental assumption, but sometimes can be wrong:**
- **"Total number of murders this year is 5% of the population"**
- **"If I live 100 years, then I will commit about 5 murders, and if I live 60 years, I will commit about 3 murders"**
- **… non-ergodic!**
- **Such ergodicity assumptions on population ensembles is commonly called "racism."**



Stationarity
Example

- **Is the chicken quality consistent?**
  - Last week:

$$\textstyle\sum_{t:\text{ last week}} x_t^{(\text{chicken})}/7$$

  - Two weeks ago:

$$\textstyle\sum_{t:\text{ two weeks ago}} x_t^{(\text{chicken})}/7$$

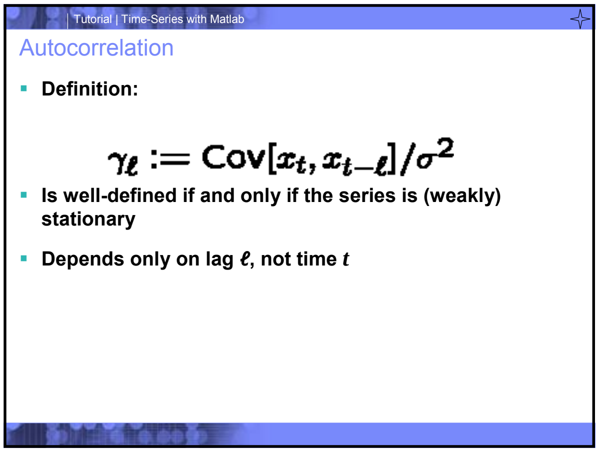
  - Last month:

$$\textstyle\sum_{t:\text{ last month}} x_t^{(\text{chicken})}/30$$

  - Last year:

$$\textstyle\sum_{t:\text{ last year}} x_t^{(\text{chicken})}/365$$

- **Answers are equal ⇒ stationary**

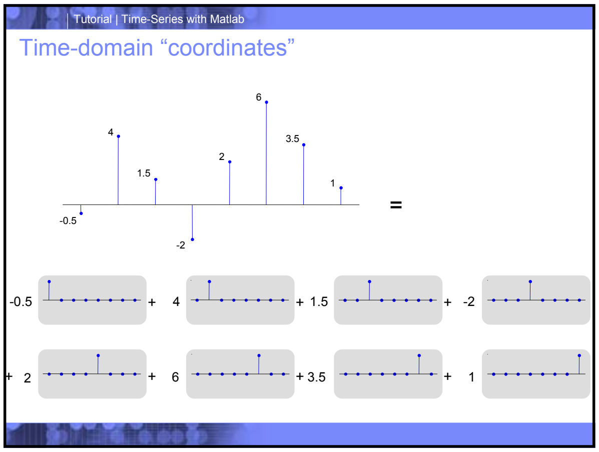


Autocorrelation

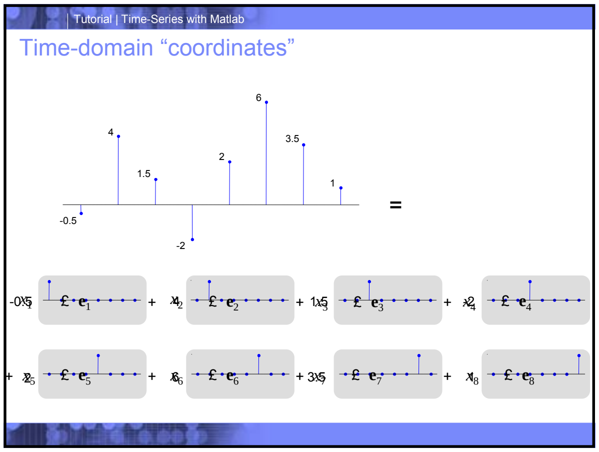
- **Definition:**

$$\gamma_\ell := \mathrm{Cov}[x_t, x_{t-\ell}]/\sigma^2$$

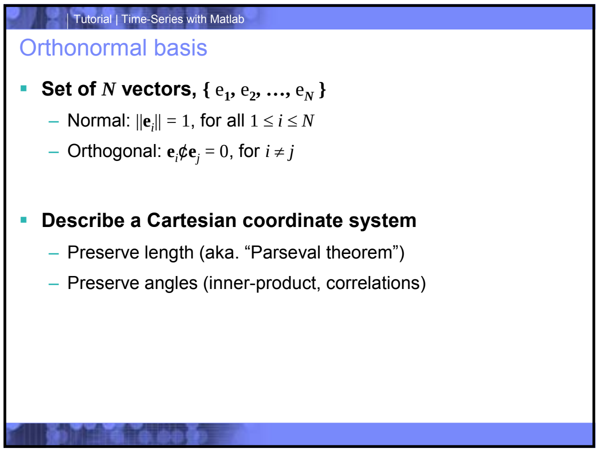
- **Is well-defined if and only if the series is (weakly) stationary**
- **Depends only on lag ℓ, not time t**



Time-domain "coordinates"

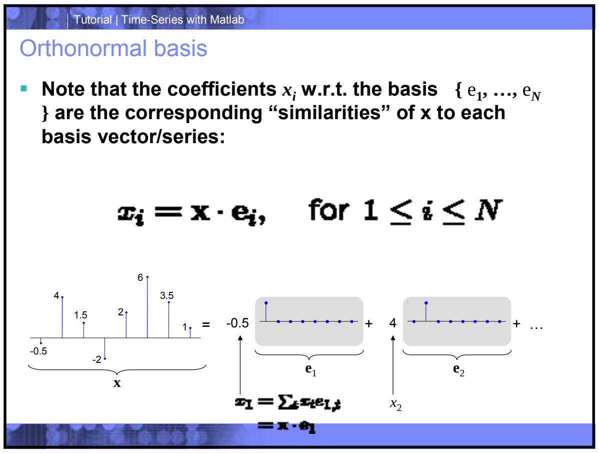

Tutorial | Time-Series with Matlab

Time-domain “coordinates”


Tutorial | Time-Series with Matlab

Orthonormal basis

- **Set of N vectors, { e_1, e_2, …, e_N }**
  - Normal: $\|\mathbf{e}_i\| = 1$, for all $1 \le i \le N$
  - Orthogonal: $\mathbf{e}_i$¢$\mathbf{e}_j = 0$, for $i \neq j$
- **Describe a Cartesian coordinate system**
  - Preserve length (aka. “Parseval theorem”)
  - Preserve angles (inner-product, correlations)

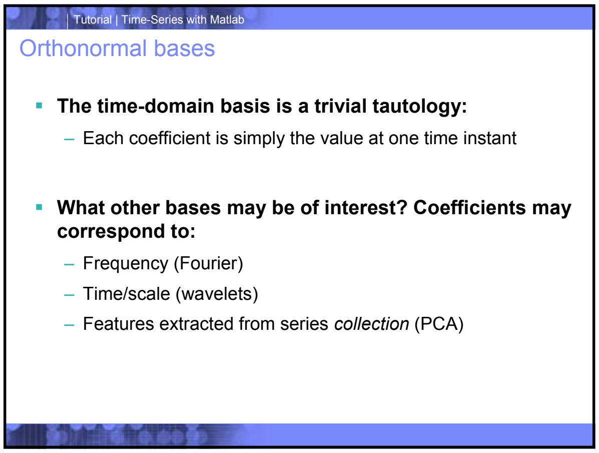

Tutorial | Time-Series with Matlab

Orthonormal basis

- **Note that the coefficients x_i w.r.t. the basis { e_1, …, e_N } are the corresponding “similarities” of x to each basis vector/series:**

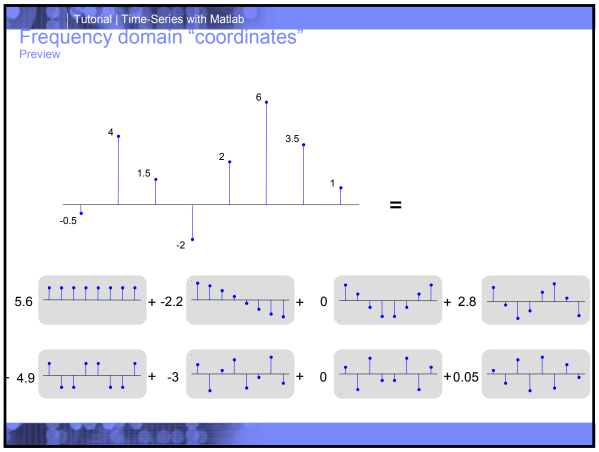
$$x_i = \mathbf{x} \cdot \mathbf{e}_i, \quad \text{for } 1 \le i \le N$$

$$x_1 = \sum_t x_t e_{1,t}$$
$$= \mathbf{x} \cdot \mathbf{e}_1$$


Tutorial | Time-Series with Matlab

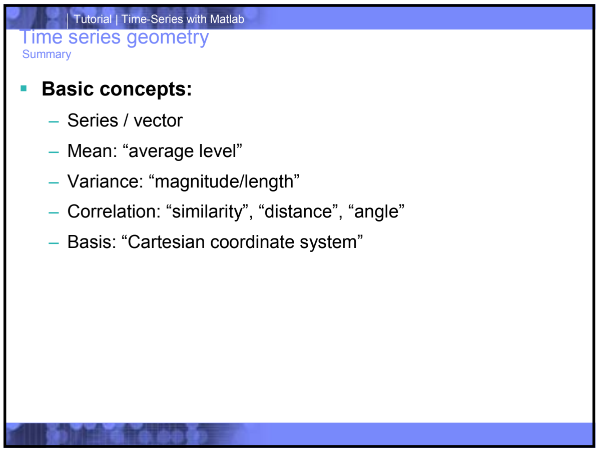
Orthonormal bases

- **The time-domain basis is a trivial tautology:**
  - Each coefficient is simply the value at one time instant
- **What other bases may be of interest? Coefficients may correspond to:**
  - Frequency (Fourier)
  - Time/scale (wavelets)
  - Features extracted from series *collection* (PCA)


Tutorial | Time-Series with Matlab

Frequency domain “coordinates”

Preview


Tutorial | Time-Series with Matlab

Time series geometry

Summary

- **Basic concepts:**
  - Series / vector
  - Mean: “average level”
  - Variance: “magnitude/length”
  - Correlation: “similarity”, “distance”, “angle”
  - Basis: “Cartesian coordinate system”

Time series geometry

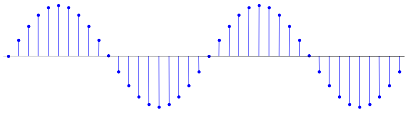
Preview — Applications

- **The quest for the right basis…**
- **Compression / pattern extraction**
  - Filtering
  - Similarity / distance
  - Indexing
  - Clustering
  - Forecasting
  - Periodicity estimation
  - Correlation

Overview

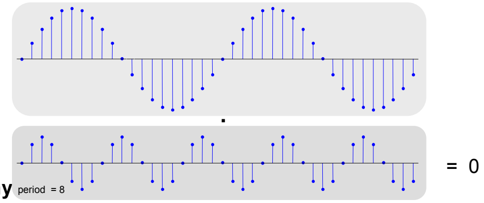
1. **Introduction and geometric intuition**
2. **Coordinates and transforms**
   - Fourier transform (DFT)
   - Wavelet transform (DWT)
   - Incremental DWT
   - Principal components (PCA)
   - Incremental PCA
3. **Quantized representations**
   - Piecewise quantized / symbolic
   - Vector quantization (VQ) / K-means
4. **Non-Euclidean distances**
   - Dynamic time warping (DTW)

Frequency



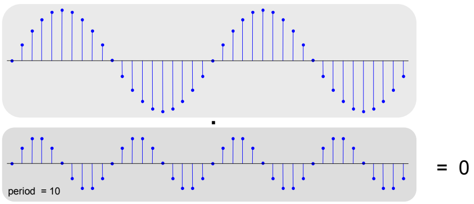
- **One cycle every 20 time units (period)**

Frequency and time



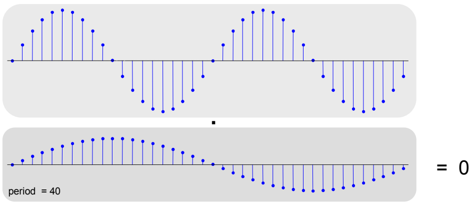
- **Why**
- **It's not 8, because its "similarity" (projection) to a period-8 series (of the same length) is zero.**

Frequency and time



- **Why is the cycle 20?**
- **It's not 10, because its "similarity" (projection) to a period-10 series (of the same length) is zero.**

Frequency and time



- **Why is the cycle 20?**
- **It's not 40, because its "similarity" (projection) to a period-40 series (of the same length) is zero.**

…and so on

Frequency

Fourier transform - Intuition

- **To find the period, we compared the time series with sinusoids of many different periods**
- **Therefore, a good “description” (or basis) would consist of all these sinusoids**
- **This is precisely the idea behind the discrete Fourier transform**
  - The coefficients capture the similarity (in terms of amplitude and phase) of the series with sinusoids of different periods

Frequency

Fourier transform - Intuition

- **Technical details:**
  - We have to ensure we get an orthonormal basis
  - Real form: sines and cosines at $N/2$ different frequencies
  - Complex form: exponentials at N different frequencies

Fourier transform

Real form

- **For odd-length series,**

$$x_t = \frac{\alpha_0}{\sqrt{N}} + \frac{1}{\sqrt{N}} \sum_{k=1}^{\lfloor N/2 \rfloor} \left(\alpha_k \cos(2\pi f_k t) + \beta_k \sin(2\pi f_k t)\right), \text{ where}$$

$$f_k := \frac{k}{N}, \text{ for } 1 \leq k \leq \lfloor N/2 \rfloor$$

- **The pair of bases at frequency f_k are**

$$\mathbf{a}_k := \frac{1}{\sqrt{N}}\left(\cos(2\pi f_k 1), \ldots, \cos(2\pi f_k t), \ldots, \cos(2\pi f_k N)\right) \in \mathbb{R}^N$$

$$\mathbf{b}_k := \frac{1}{\sqrt{N}}\left(\sin(2\pi f_k 1), \ldots, \sin(2\pi f_k t), \ldots, \sin(2\pi f_k N)\right) \in \mathbb{R}^N$$

plus the zero-frequency (mean) component

$$\mathbf{a}_0 := \left(\frac{1}{\sqrt{N}}, \ldots, \frac{1}{\sqrt{N}}\right) \in \mathbb{R}^N$$

Fourier transform

Real form — Amplitude and phase

- **Observe that, for any f_k, we can write**

$$\alpha_k \cos(2\pi f_k t) + \beta_k \sin(2\pi f_k t) = r_k cos(2\pi f_k t - \theta_k)$$

where

$$r_k := \sqrt{\alpha_k^2 + \beta_k^2}$$

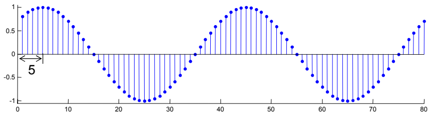
$$\theta_k := \arctan(\beta_k, \alpha_k)$$

are the amplitude and phase, respectively.

Fourier transform

Real form — Amplitude and phase

- **It is often easier to think in terms of amplitude r_k and phase θ_k – e.g.,**



$$\frac{\sqrt{2}}{2}\cos\left(\frac{2\pi}{40}t\right) + \frac{\sqrt{2}}{2}\sin\left(\frac{2\pi}{40}t\right) =$$

$$\cos\left(\frac{2\pi}{40}t - \frac{\pi}{4}\right) = \cos\left(\frac{2\pi}{40}(t-5)\right),$$

$$1 \leq t \leq 80$$

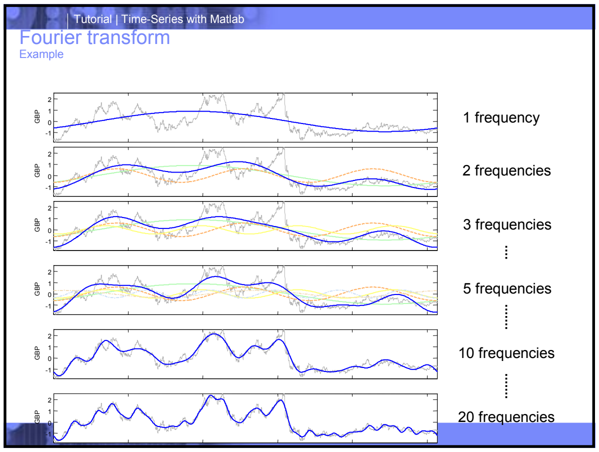
Fourier transform

Complex form

- **The equations become easier to handle if we allow the series and the Fourier coefficients X_k to take complex values:**

$$x_t = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X_k e^{-2\pi i f_k}$$

- **Matlab note:** `fft` **omits the** $1/\sqrt{N}$ **scaling factor and is not unitary—however,** `ifft` **includes an** $1/N$ **scaling factor, so always** `ifft(fft(x)) == x`



Fourier transform

Example

1 frequency

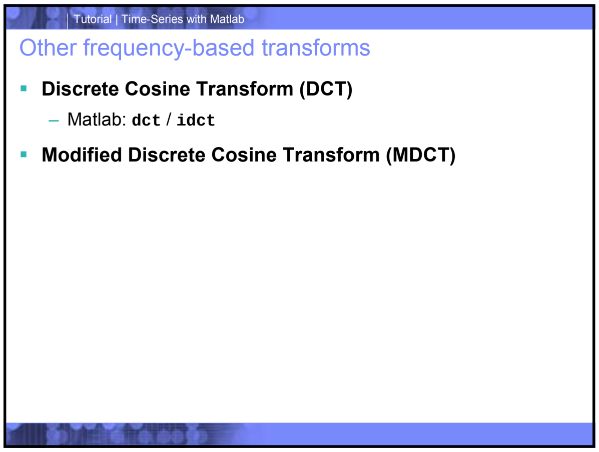
2 frequencies

3 frequencies

5 frequencies

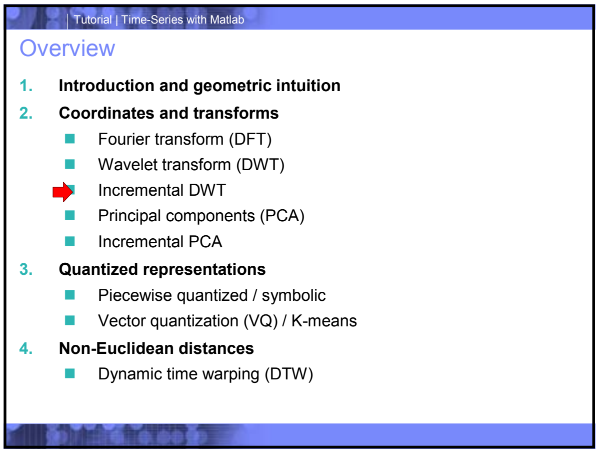
10 frequencies

20 frequencies



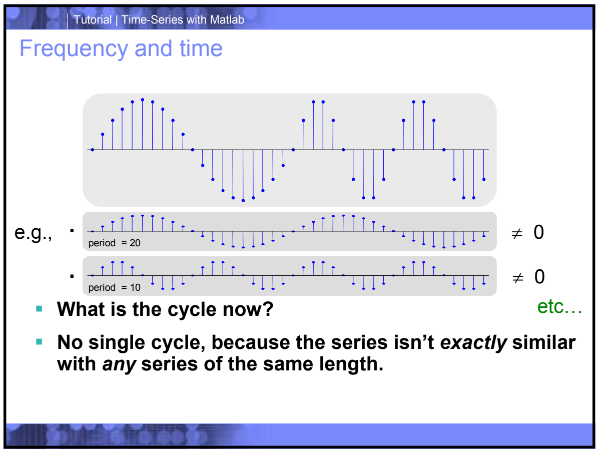
Other frequency-based transforms

- **Discrete Cosine Transform (DCT)**
  - Matlab: `dct` / `idct`
- **Modified Discrete Cosine Transform (MDCT)**



Overview

1. **Introduction and geometric intuition**
2. **Coordinates and transforms**
   - Fourier transform (DFT)
   - Wavelet transform (DWT)
   - Incremental DWT
   - Principal components (PCA)
   - Incremental PCA
3. **Quantized representations**
   - Piecewise quantized / symbolic
   - Vector quantization (VQ) / K-means
4. **Non-Euclidean distances**
   - Dynamic time warping (DTW)



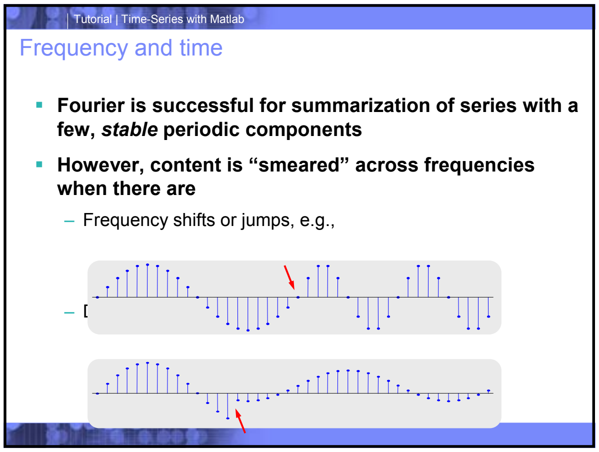
Frequency and time

e.g.,

- period = 20 ≠ 0
- period = 10 ≠ 0

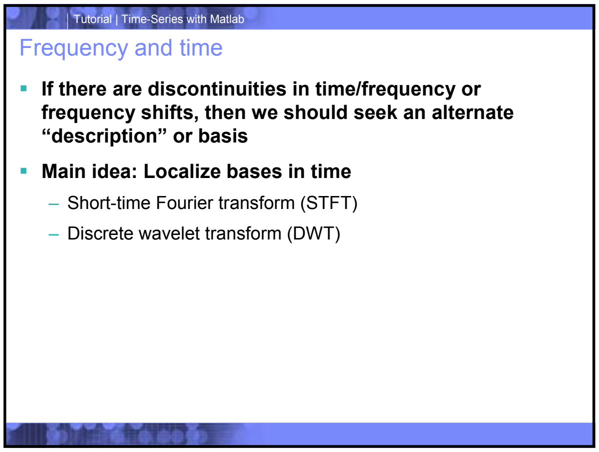
etc…

- **What is the cycle now?**
- **No single cycle, because the series isn't *exactly* similar with *any* series of the same length.**



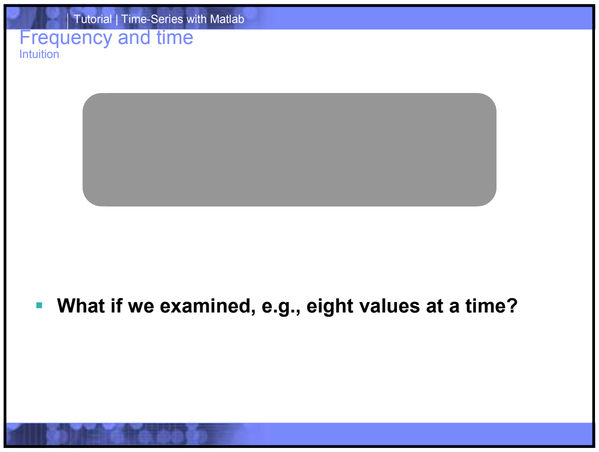
Frequency and time

- **Fourier is successful for summarization of series with a few, *stable* periodic components**
- **However, content is "smeared" across frequencies when there are**
  - Frequency shifts or jumps, e.g.,
  - [



Frequency and time

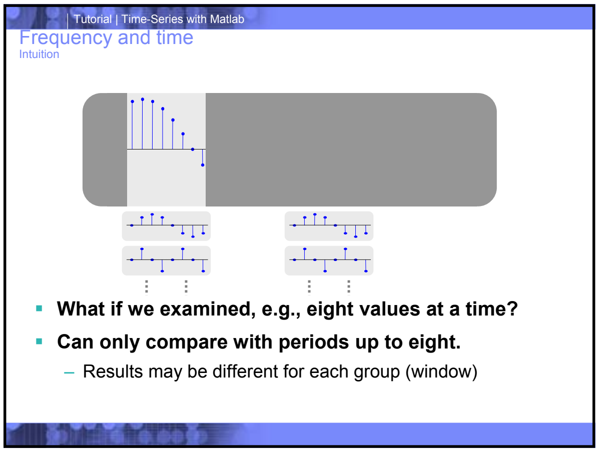
- **If there are discontinuities in time/frequency or frequency shifts, then we should seek an alternate "description" or basis**
- **Main idea: Localize bases in time**
  - Short-time Fourier transform (STFT)
  - Discrete wavelet transform (DWT)



Frequency and time

Intuition

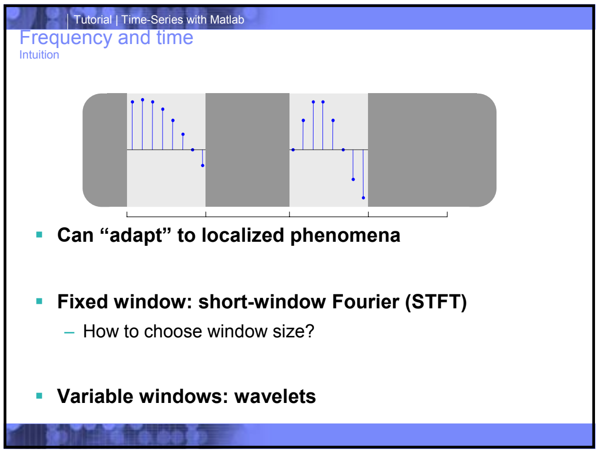
- **What if we examined, e.g., eight values at a time?**



Frequency and time

Intuition

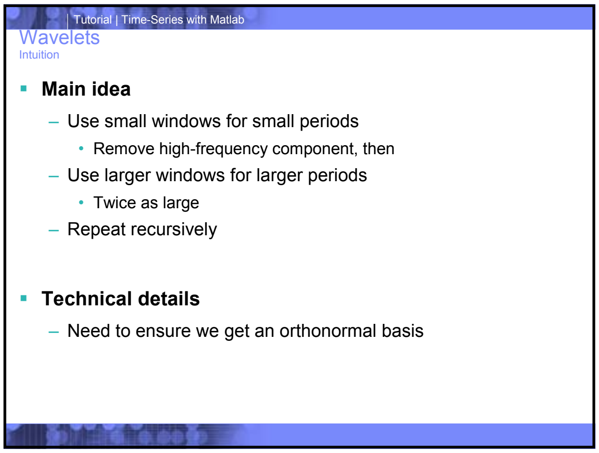
- **What if we examined, e.g., eight values at a time?**
- **Can only compare with periods up to eight.**
  - Results may be different for each group (window)



Frequency and time

Intuition

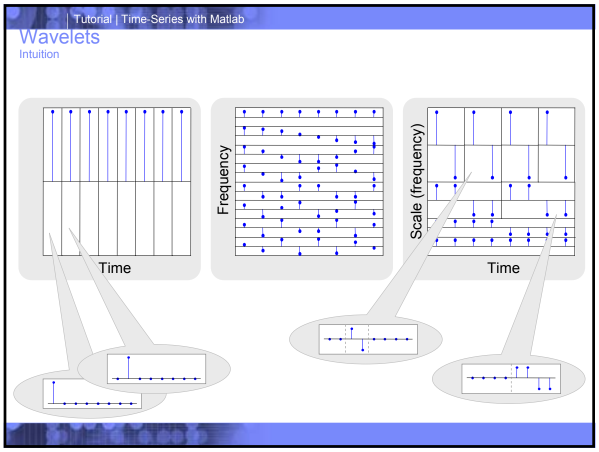
- **Can “adapt” to localized phenomena**
- **Fixed window: short-window Fourier (STFT)**
  - How to choose window size?
- **Variable windows: wavelets**



Wavelets

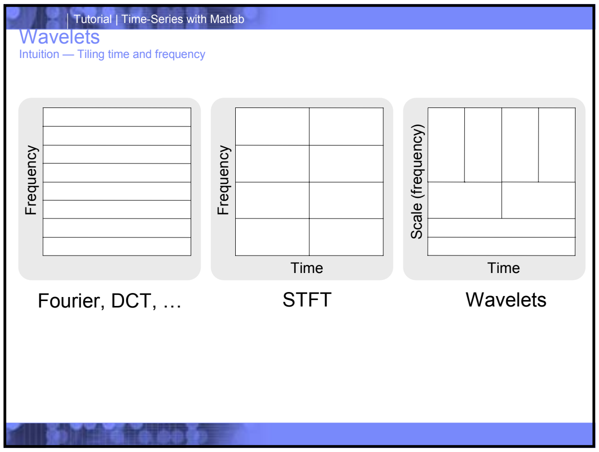
Intuition

- **Main idea**
  - Use small windows for small periods
    - Remove high-frequency component, then
  - Use larger windows for larger periods
    - Twice as large
  - Repeat recursively
- **Technical details**
  - Need to ensure we get an orthonormal basis



Wavelets

Intuition



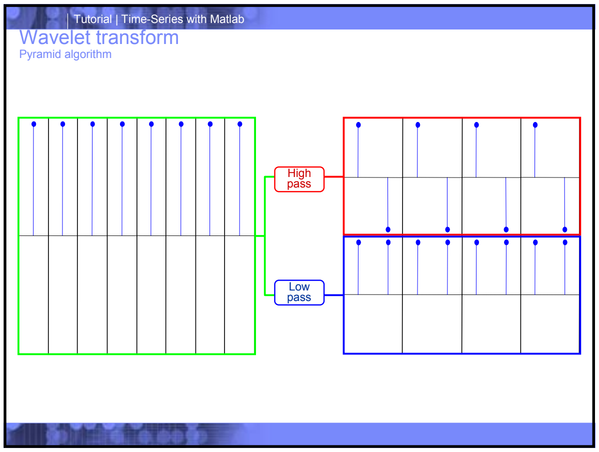
Wavelets

Intuition — Tiling time and frequency

Fourier, DCT, …

STFT

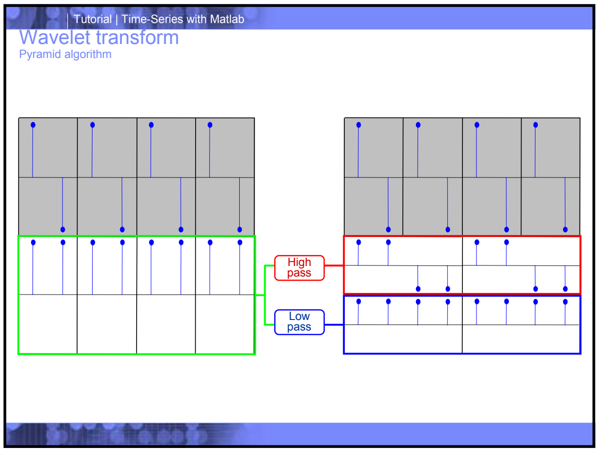
Wavelets



Tutorial | Time-Series with Matlab

Wavelet transform

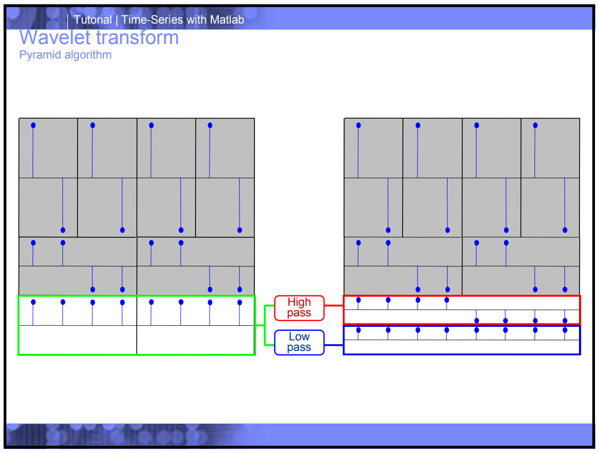
Pyramid algorithm



Tutorial | Time-Series with Matlab

Wavelet transform

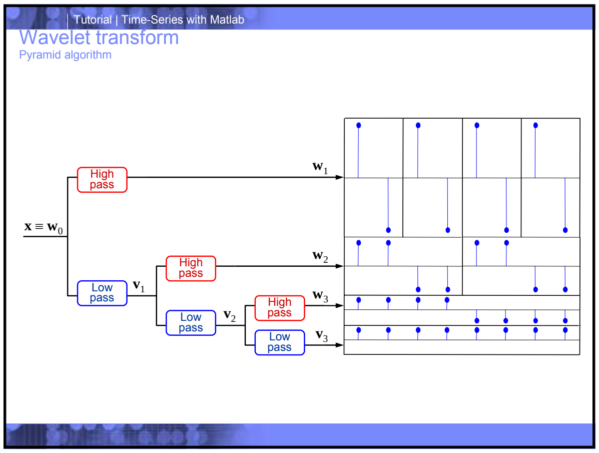
Pyramid algorithm



Tutorial | Time-Series with Matlab

Wavelet transform

Pyramid algorithm



Tutorial | Time-Series with Matlab

Wavelet transform

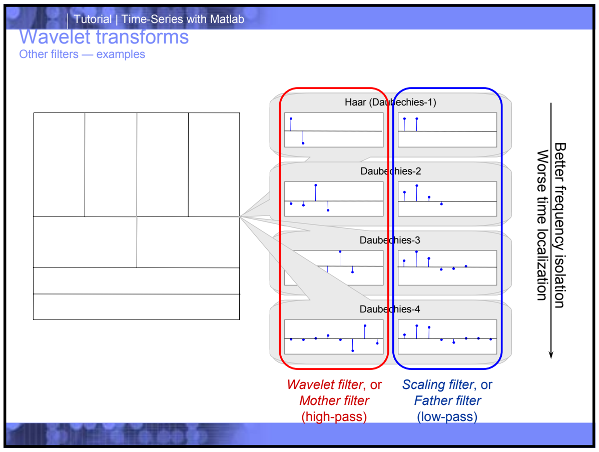
Pyramid algorithm

Tutorial | Time-Series with Matlab

Wavelet transforms

General form

- **A high-pass / low-pass filter pair**
  - Example: pairwise difference / average (Haar)
  - In general: Quadrature Mirror Filter (QMF) pair
    - Orthogonal spans, which cover the entire space
  - Additional requirements to ensure orthonormality of overall transform…
- **Use to recursively analyze into top / bottom half of frequency band**



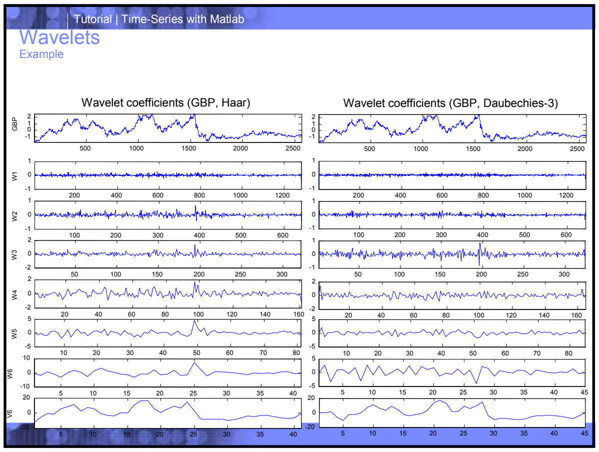
Tutorial | Time-Series with Matlab

Wavelet transforms

Other filters — examples

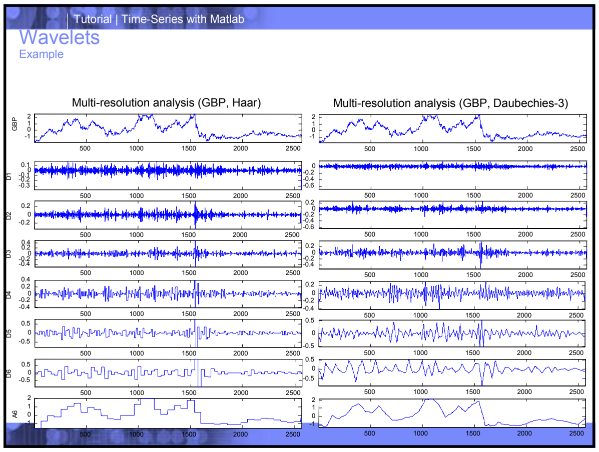
Wavelet filter, or *Mother filter* (high-pass)

Scaling filter, or *Father filter* (low-pass)



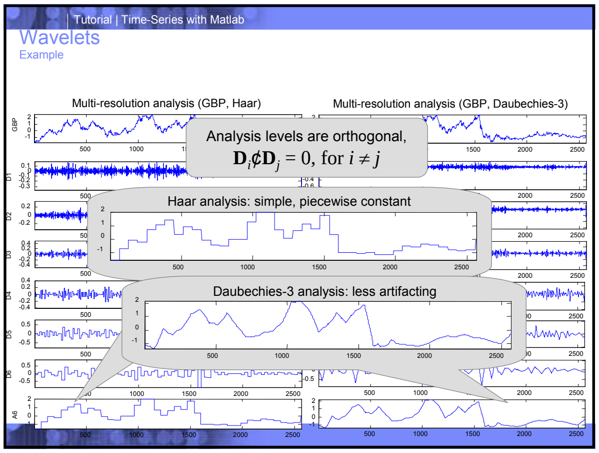
Wavelets

Example



Wavelets

Example



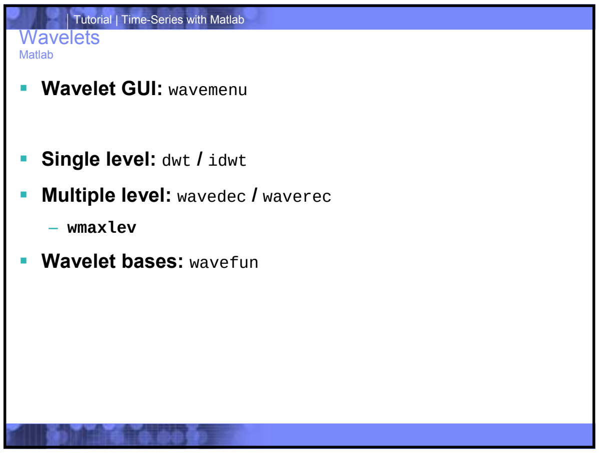
Wavelets

Example

Analysis levels are orthogonal,
$\mathbf{D}_i$¢$\mathbf{D}_j = 0$, for $i \neq j$

Haar analysis: simple, piecewise constant

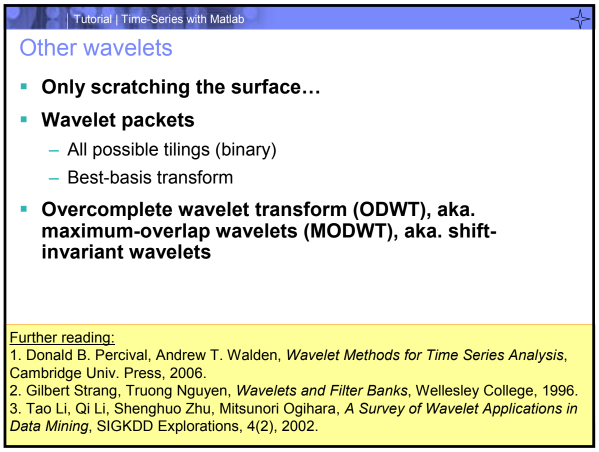
Daubechies-3 analysis: less artifacting



Wavelets

Matlab

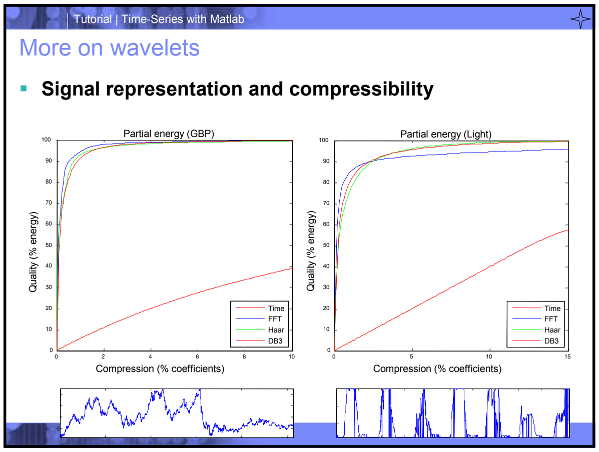
- **Wavelet GUI:** `wavemenu`
- **Single level:** `dwt` **/** `idwt`
- **Multiple level:** `wavedec` **/** `waverec`
  - **`wmaxlev`**
- **Wavelet bases:** `wavefun`



Other wavelets

- **Only scratching the surface…**
- **Wavelet packets**
  - All possible tilings (binary)
  - Best-basis transform
- **Overcomplete wavelet transform (ODWT), aka. maximum-overlap wavelets (MODWT), aka. shift-invariant wavelets**

Further reading:
1. Donald B. Percival, Andrew T. Walden, *Wavelet Methods for Time Series Analysis*, Cambridge Univ. Press, 2006.
2. Gilbert Strang, Truong Nguyen, *Wavelets and Filter Banks*, Wellesley College, 1996.
3. Tao Li, Qi Li, Shenghuo Zhu, Mitsunori Ogihara, *A Survey of Wavelet Applications in Data Mining*, SIGKDD Explorations, 4(2), 2002.



More on wavelets

- **Signal representation and compressibility**

More wavelets

- **Keeping the highest coefficients minimizes *total* error (L2-distance)**
- **Other coefficient selection/thresholding schemes for different error metrics (e.g., maximum per-instant error, or L^1-dist.)**
  - Typically use Haar bases

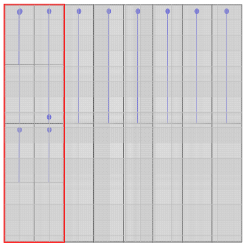
Further reading:
1. Minos Garofalakis, Amit Kumar, *Wavelet Synopses for General Error Metrics*, ACM TODS, 30(4), 2005.
2.Panagiotis Karras, Nikos Mamoulis, *One-pass Wavelet Synopses for Maximum-Error Metrics*, VLDB 2005.

Overview

1. **Introduction and geometric intuition**
2. **Coordinates and transforms**
   - Fourier transform (DFT)
   - Wavelet transform (DWT)
   - Incremental DWT
   - ➡ Principal components (PCA)
   - Incremental PCA
3. **Quantized representations**
   - Piecewise quantized / symbolic
   - Vector quantization (VQ) / K-means
4. **Non-Euclidean distances**
   - Dynamic time warping (DTW)

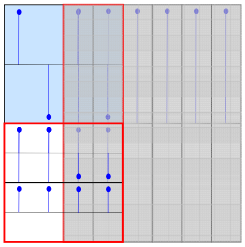
Wavelets

Incremental estimation



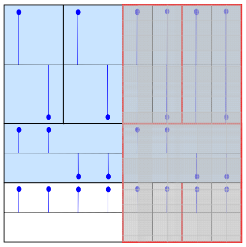
Wavelets

Incremental estimation



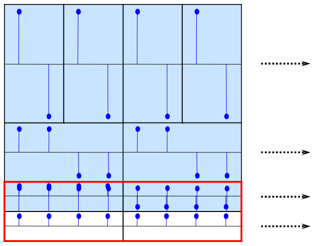
Wavelets

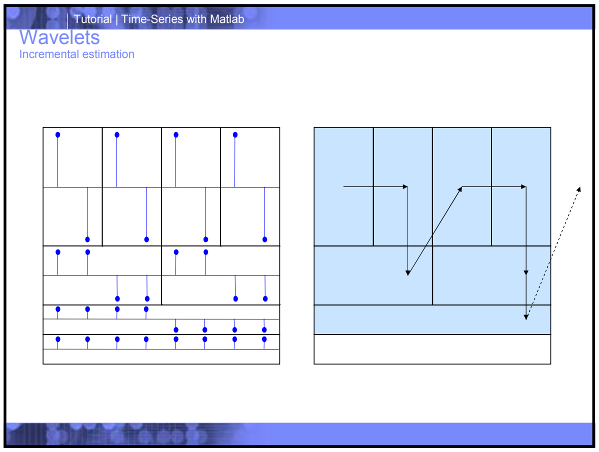
Incremental estimation



Wavelets

Incremental estimation

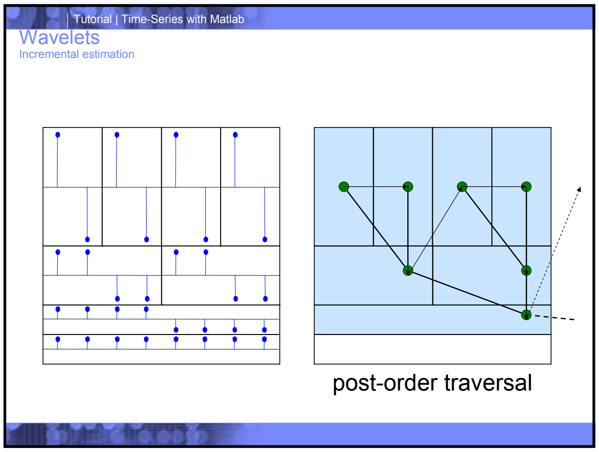




Tutorial | Time-Series with Matlab

Wavelets

Incremental estimation

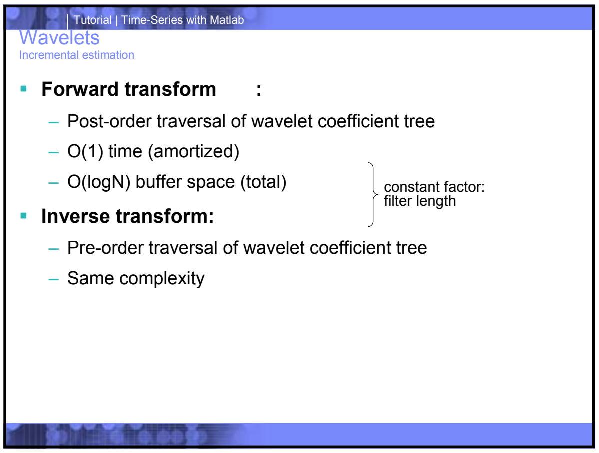


Tutorial | Time-Series with Matlab

Wavelets

Incremental estimation

post-order traversal



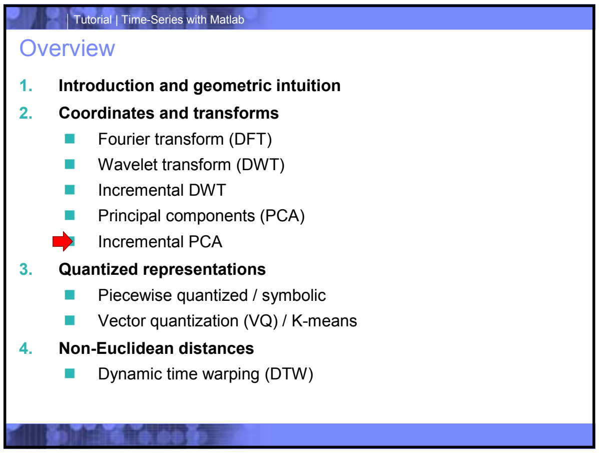
Tutorial | Time-Series with Matlab

Wavelets

Incremental estimation

- **Forward transform :**
  - Post-order traversal of wavelet coefficient tree
  - O(1) time (amortized)
  - O(logN) buffer space (total)
- **Inverse transform:**
  - Pre-order traversal of wavelet coefficient tree
  - Same complexity

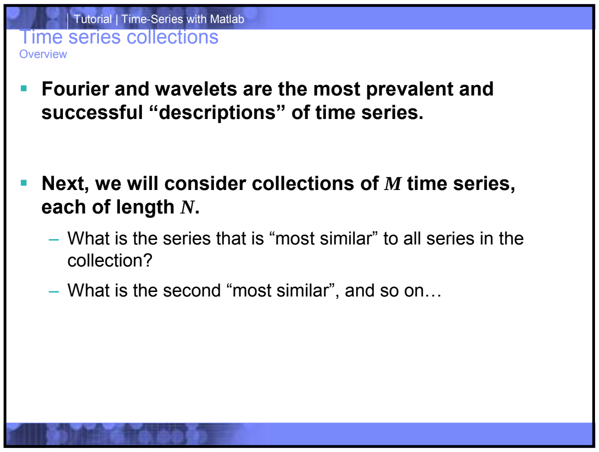
constant factor: filter length



Tutorial | Time-Series with Matlab

Overview

1. **Introduction and geometric intuition**
2. **Coordinates and transforms**
   - Fourier transform (DFT)
   - Wavelet transform (DWT)
   - Incremental DWT
   - Principal components (PCA)
   - ➡ Incremental PCA
3. **Quantized representations**
   - Piecewise quantized / symbolic
   - Vector quantization (VQ) / K-means
4. **Non-Euclidean distances**
   - Dynamic time warping (DTW)

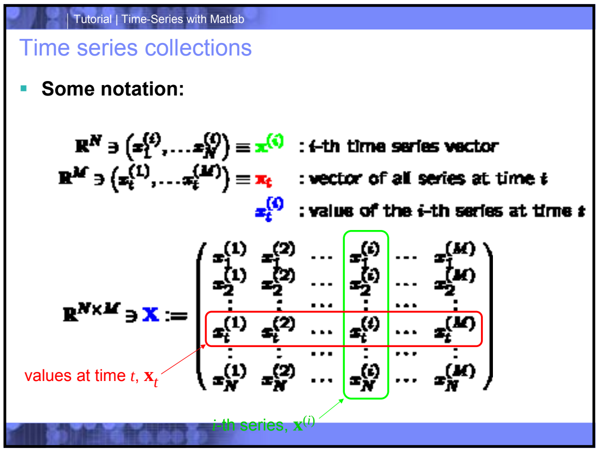


Tutorial | Time-Series with Matlab

Time series collections

Overview

- **Fourier and wavelets are the most prevalent and successful "descriptions" of time series.**
- **Next, we will consider collections of M time series, each of length N.**
  - What is the series that is "most similar" to all series in the collection?
  - What is the second "most similar", and so on…



Tutorial | Time-Series with Matlab

Time series collections

- **Some notation:**

$$\mathbb{R}^N \ni \left(x_1^{(i)}, \ldots x_N^{(i)}\right) \equiv \mathbf{x}^{(i)} \quad : i\text{-th time series vector}$$

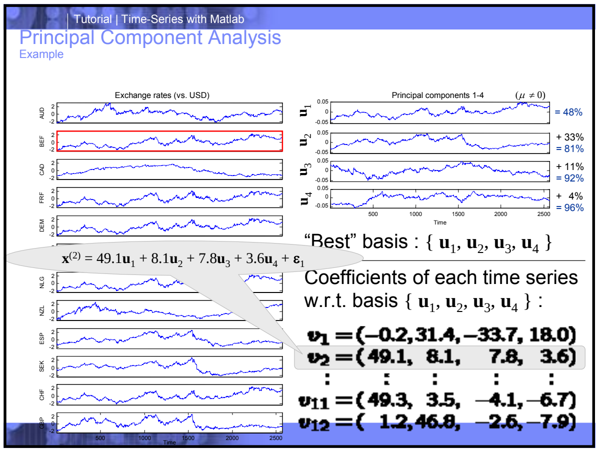
$$\mathbb{R}^M \ni \left(x_t^{(1)}, \ldots x_t^{(M)}\right) \equiv \mathbf{x}_t \quad : \text{vector of all series at time } t$$

$$x_t^{(i)} \quad : \text{value of the } i\text{-th series at time } t$$

$$\mathbb{R}^{N\times M} \ni \mathbf{X} := \begin{pmatrix} x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(i)} & \cdots & x_1^{(M)} \\ x_2^{(1)} & x_2^{(2)} & \cdots & x_2^{(i)} & \cdots & x_2^{(M)} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ x_t^{(1)} & x_t^{(2)} & \cdots & x_t^{(i)} & \cdots & x_t^{(M)} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ x_N^{(1)} & x_N^{(2)} & \cdots & x_N^{(i)} & \cdots & x_N^{(M)} \end{pmatrix}$$

values at time t, $\mathbf{x}_t$

i-th series, $\mathbf{x}^{(i)}$



Tutorial | Time-Series with Matlab

Principal Component Analysis

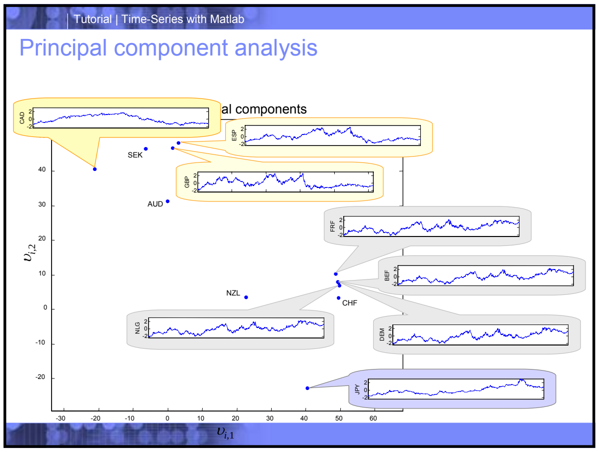
Example

$\mathbf{x}^{(2)} = 49.1\mathbf{u}_1 + 8.1\mathbf{u}_2 + 7.8\mathbf{u}_3 + 3.6\mathbf{u}_4 + \boldsymbol{\varepsilon}_1$

"Best" basis : $\{ \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{u}_4 \}$

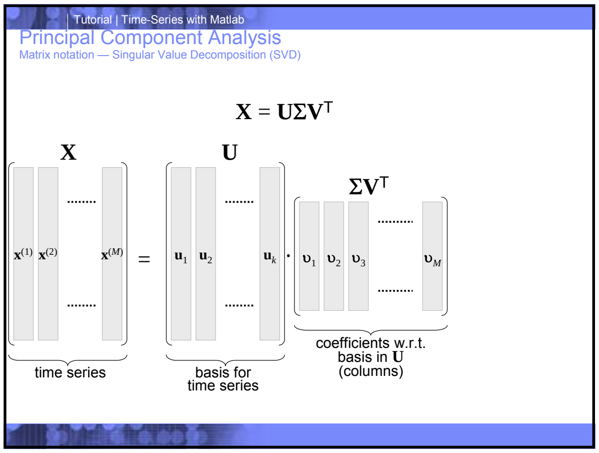
Coefficients of each time series w.r.t. basis $\{ \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{u}_4 \}$:

$$\begin{aligned}
\upsilon_1 &= (-0.2, 31.4, -33.7, 18.0) \\
\upsilon_2 &= (49.1, 8.1, 7.8, 3.6) \\
&\vdots \\
\upsilon_{11} &= (49.3, 3.5, -4.1, -6.7) \\
\upsilon_{12} &= (1.2, 46.8, -2.6, -7.9)
\end{aligned}$$



Tutorial | Time-Series with Matlab

Principal component analysis



Tutorial | Time-Series with Matlab

Principal Component Analysis

Matrix notation — Singular Value Decomposition (SVD)

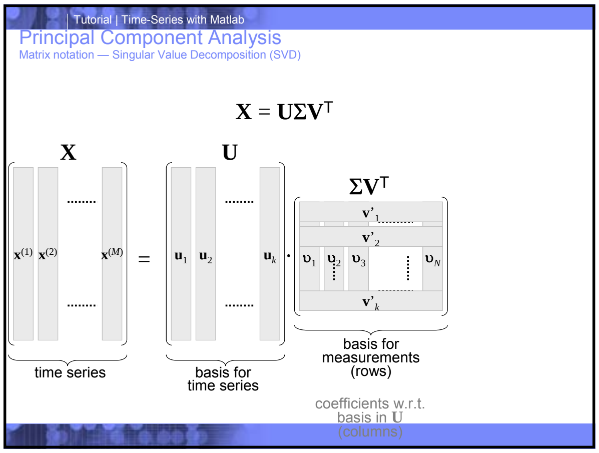
$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^\mathsf{T}$$

$\mathbf{X}$
$\mathbf{x}^{(1)}$ $\mathbf{x}^{(2)}$ .... $\mathbf{x}^{(M)}$
=
$\mathbf{U}$
$\mathbf{u}_1$ $\mathbf{u}_2$ .... $\mathbf{u}_k$
·
$\Sigma\mathbf{V}^\mathsf{T}$
υ_1 υ_2 υ_3 .... υ_M

time series

basis for time series

coefficients w.r.t. basis in $\mathbf{U}$ (columns)



Tutorial | Time-Series with Matlab

Principal Component Analysis

Matrix notation — Singular Value Decomposition (SVD)

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^\mathsf{T}$$

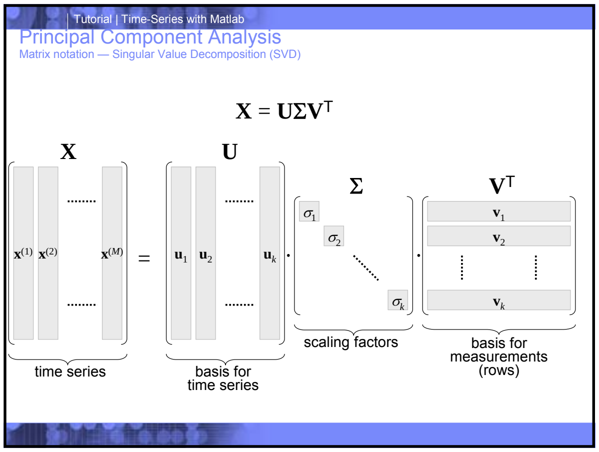
$\mathbf{X}$
$\mathbf{x}^{(1)}$ $\mathbf{x}^{(2)}$ .... $\mathbf{x}^{(M)}$
=
$\mathbf{U}$
$\mathbf{u}_1$ $\mathbf{u}_2$ .... $\mathbf{u}_k$
·
$\Sigma\mathbf{V}^\mathsf{T}$
$\mathbf{v}'_1$
$\mathbf{v}'_2$
$\mathbf{v}'_k$
υ_1 υ_2 υ_3 υ_N

time series

basis for time series

basis for measurements (rows)

coefficients w.r.t. basis in $\mathbf{U}$ (columns)



Tutorial | Time-Series with Matlab

Principal Component Analysis

Matrix notation — Singular Value Decomposition (SVD)

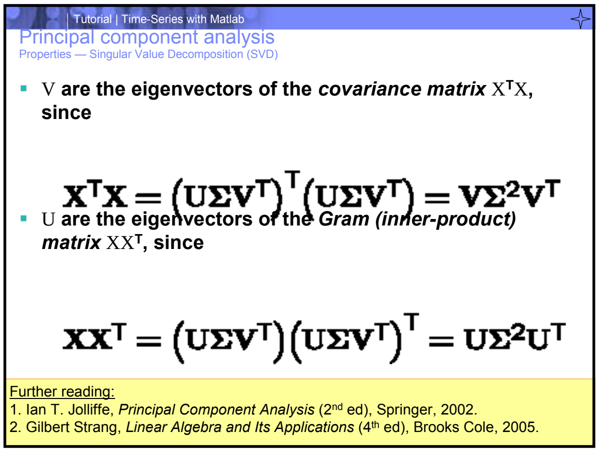
$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^\mathsf{T}$$

time series

basis for time series

scaling factors

basis for measurements (rows)



Tutorial | Time-Series with Matlab

Principal component analysis

Properties — Singular Value Decomposition (SVD)

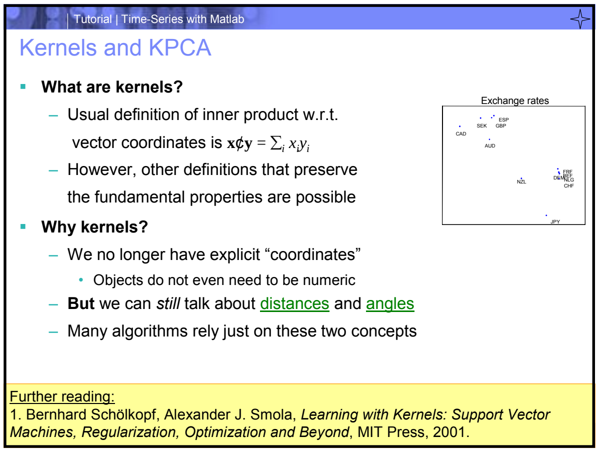
- **V are the eigenvectors of the *covariance matrix* $\mathrm{X}^\mathsf{T}\mathrm{X}$, since**

$$\mathbf{X}^\mathsf{T}\mathbf{X} = (\mathbf{U}\Sigma\mathbf{V}^\mathsf{T})^\mathsf{T}(\mathbf{U}\Sigma\mathbf{V}^\mathsf{T}) = \mathbf{V}\Sigma^2\mathbf{V}^\mathsf{T}$$

- **U are the eigenvectors of the *Gram (inner-product) matrix* $\mathrm{X}\mathrm{X}^\mathsf{T}$, since**

$$\mathbf{X}\mathbf{X}^\mathsf{T} = (\mathbf{U}\Sigma\mathbf{V}^\mathsf{T})(\mathbf{U}\Sigma\mathbf{V}^\mathsf{T})^\mathsf{T} = \mathbf{U}\Sigma^2\mathbf{U}^\mathsf{T}$$

Further reading:
1. Ian T. Jolliffe, *Principal Component Analysis* (2nd ed), Springer, 2002.
2. Gilbert Strang, *Linear Algebra and Its Applications* (4th ed), Brooks Cole, 2005.

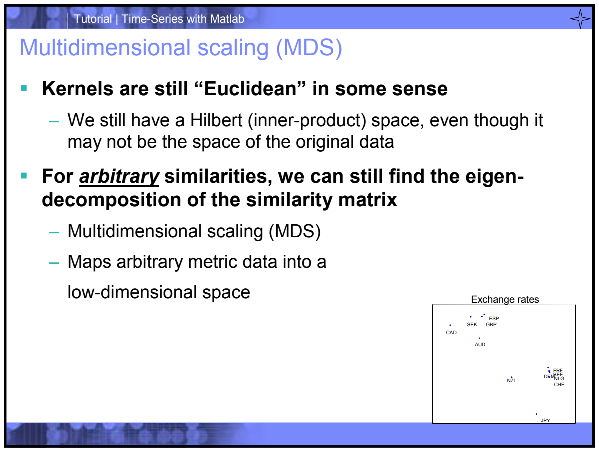


Kernels and KPCA

- **What are kernels?**
  - Usual definition of inner product w.r.t. vector coordinates is $\mathbf{x}¢\mathbf{y} = \sum_i x_i y_i$
  - However, other definitions that preserve the fundamental properties are possible
- **Why kernels?**
  - We no longer have explicit “coordinates”
    - Objects do not even need to be numeric
  - **But** we can *still* talk about distances and angles
  - Many algorithms rely just on these two concepts

Exchange rates

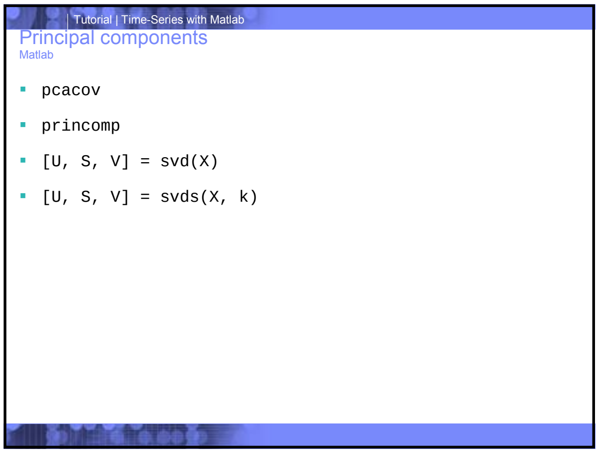
Further reading:
1. Bernhard Schölkopf, Alexander J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*, MIT Press, 2001.



Multidimensional scaling (MDS)

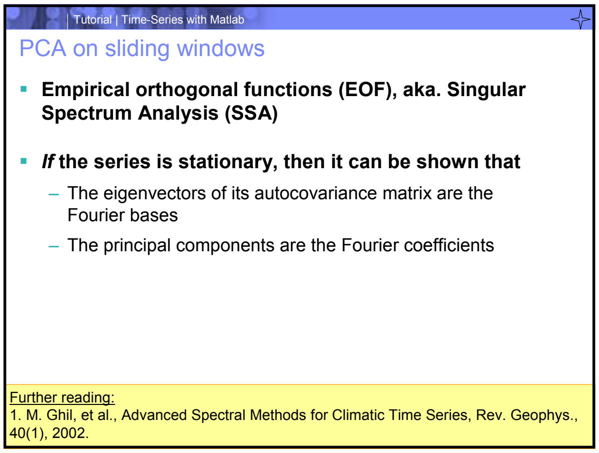
- **Kernels are still “Euclidean” in some sense**
  - We still have a Hilbert (inner-product) space, even though it may not be the space of the original data
- **For *arbitrary* similarities, we can still find the eigen-decomposition of the similarity matrix**
  - Multidimensional scaling (MDS)
  - Maps arbitrary metric data into a low-dimensional space

Exchange rates



Principal components
Matlab

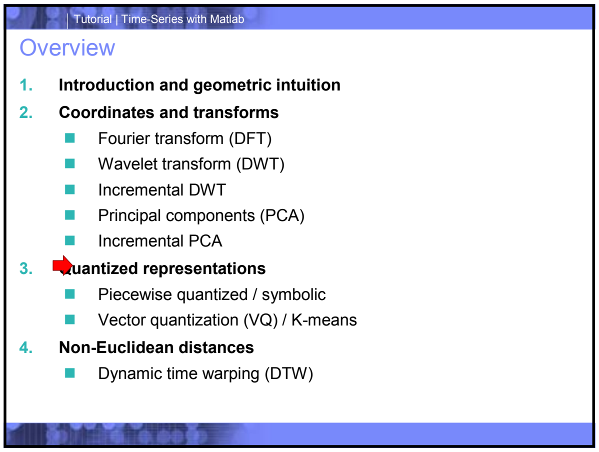
- `pcacov`
- `princomp`
- `[U, S, V] = svd(X)`
- `[U, S, V] = svds(X, k)`



PCA on sliding windows

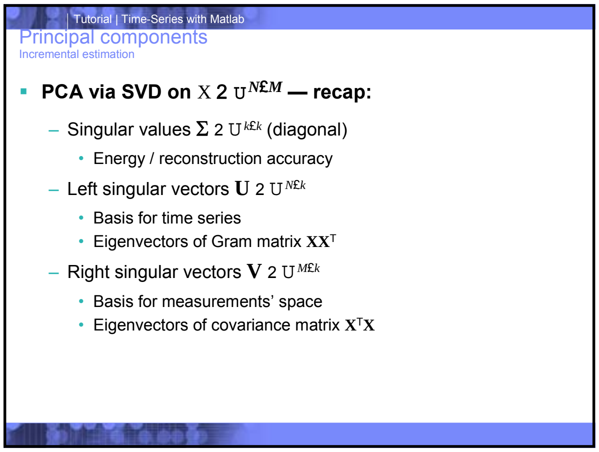
- **Empirical orthogonal functions (EOF), aka. Singular Spectrum Analysis (SSA)**
- ***If* the series is stationary, then it can be shown that**
  - The eigenvectors of its autocovariance matrix are the Fourier bases
  - The principal components are the Fourier coefficients

Further reading:
1. M. Ghil, et al., Advanced Spectral Methods for Climatic Time Series, Rev. Geophys., 40(1), 2002.



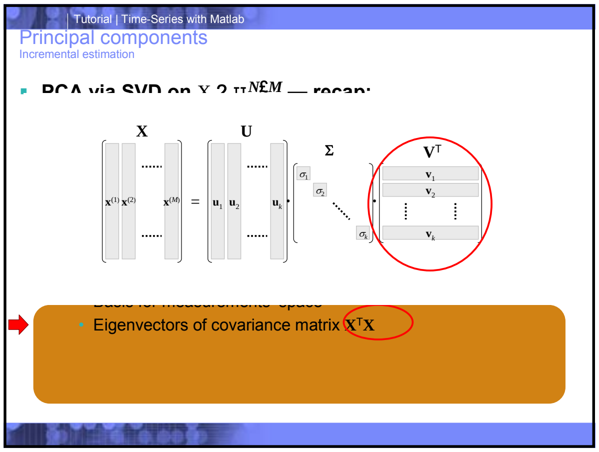
Overview

1. **Introduction and geometric intuition**
2. **Coordinates and transforms**
   - Fourier transform (DFT)
   - Wavelet transform (DWT)
   - Incremental DWT
   - Principal components (PCA)
   - Incremental PCA
3. **Quantized representations**
   - Piecewise quantized / symbolic
   - Vector quantization (VQ) / K-means
4. **Non-Euclidean distances**
   - Dynamic time warping (DTW)



Principal components
Incremental estimation

- **PCA via SVD on X 2 $\mathrm{U}^{N£M}$ — recap:**
  - Singular values $\mathbf{\Sigma}$ 2 $\mathrm{U}^{k£k}$ (diagonal)
    - Energy / reconstruction accuracy
  - Left singular vectors $\mathbf{U}$ 2 $\mathrm{U}^{N£k}$
    - Basis for time series
    - Eigenvectors of Gram matrix $\mathbf{X}\mathbf{X}^{\mathrm{T}}$
  - Right singular vectors $\mathbf{V}$ 2 $\mathrm{U}^{M£k}$
    - Basis for measurements’ space
    - Eigenvectors of covariance matrix $\mathbf{X}^{\mathrm{T}}\mathbf{X}$



Principal components

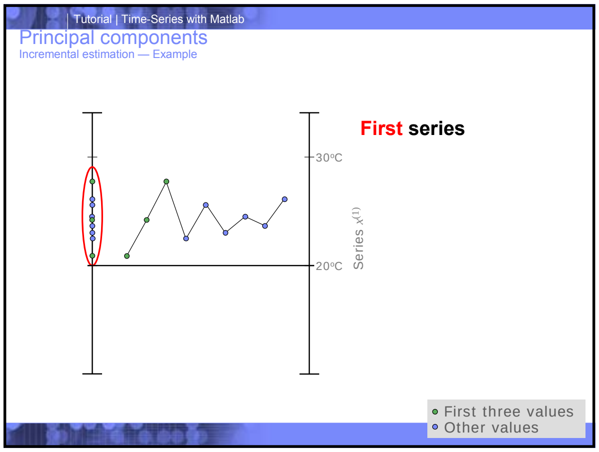
Incremental estimation

- **PCA via SVD on X 2 ɪɪ$^{N£M}$ — recap:**

X
U
Σ
$\mathbf{V}^T$
$\mathbf{x}^{(1)}$ $\mathbf{x}^{(2)}$ $\mathbf{x}^{(M)}$
=
$\mathbf{u}_1$ $\mathbf{u}_2$ $\mathbf{u}_k$
σ_1 σ_2 σ_k
$\mathbf{v}_1$ $\mathbf{v}_2$ $\mathbf{v}_k$

Basis for measurements' space

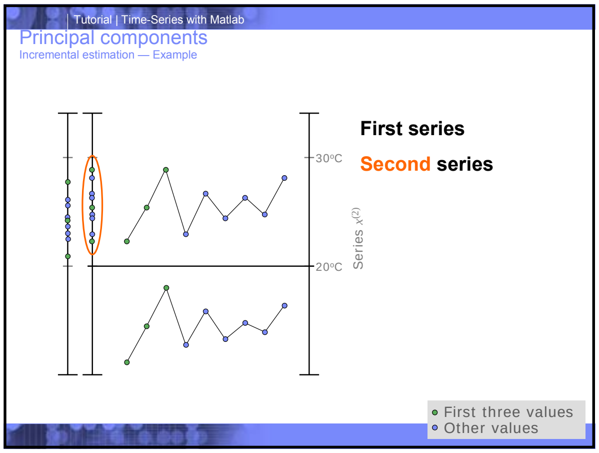
- Eigenvectors of covariance matrix $\mathbf{X}^T\mathbf{X}$



Principal components

Incremental estimation — Example

First series

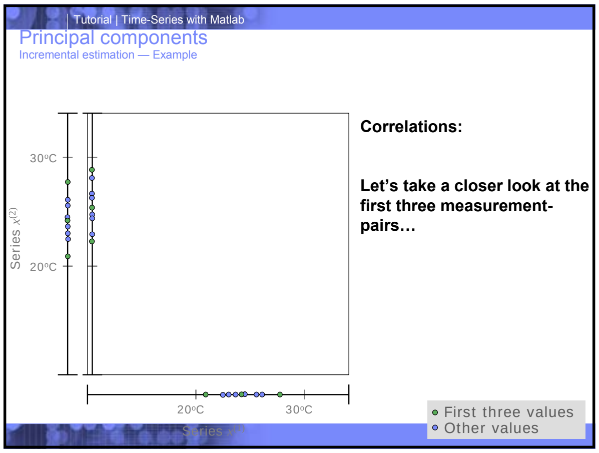


Principal components

Incremental estimation — Example

First series

Second series

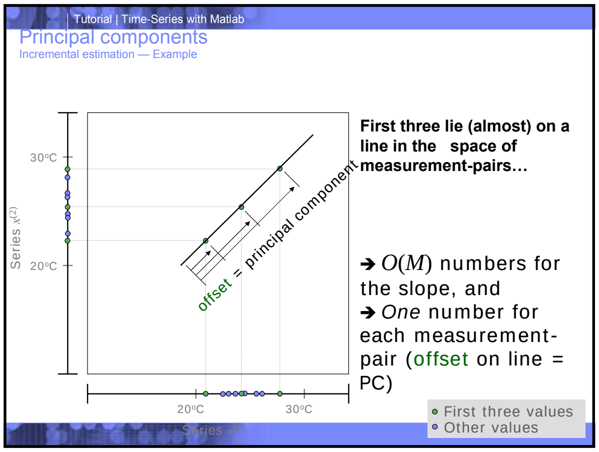


Principal components

Incremental estimation — Example

Correlations:

Let's take a closer look at the first three measurement-pairs…



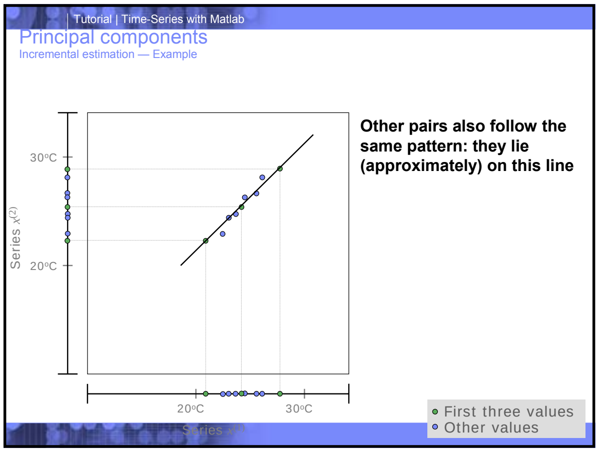
Principal components

Incremental estimation — Example

First three lie (almost) on a line in the space of measurement-pairs…

➔ $O(M)$ numbers for the slope, and

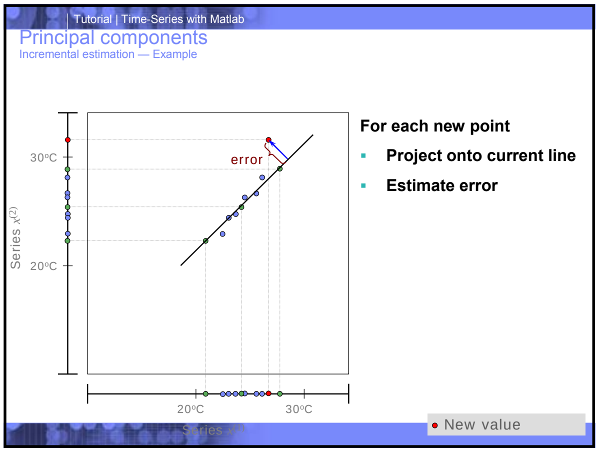
➔ *One* number for each measurement-pair (offset on line = PC)



Principal components

Incremental estimation — Example

Other pairs also follow the same pattern: they lie (approximately) on this line

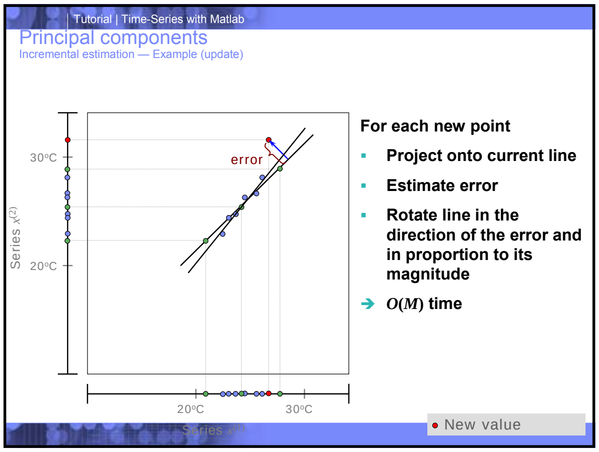


Principal components

Incremental estimation — Example

For each new point

- Project onto current line
- Estimate error



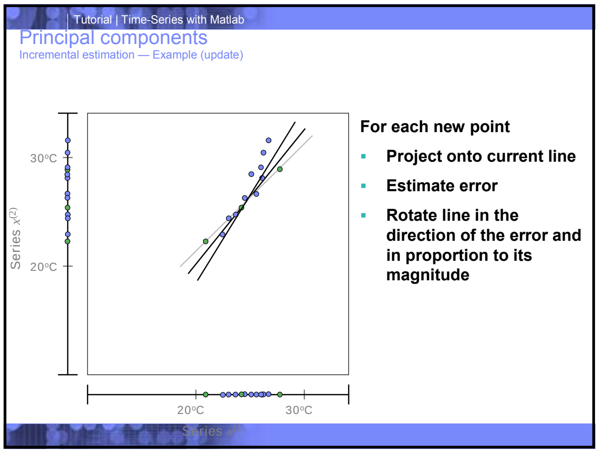
Principal components

Incremental estimation — Example (update)

For each new point

- Project onto current line
- Estimate error
- Rotate line in the direction of the error and in proportion to its magnitude

➔ $O(M)$ time

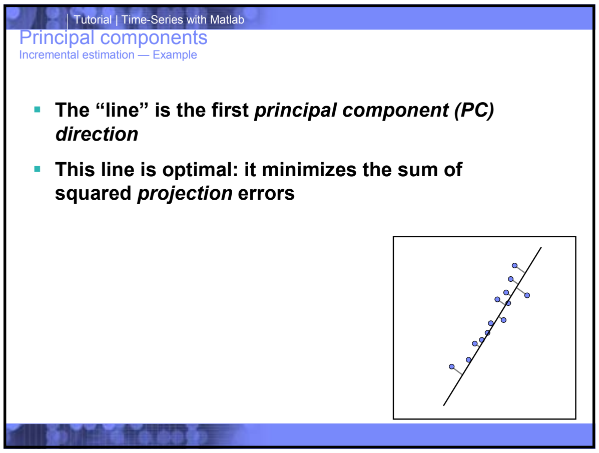


Principal components

Incremental estimation — Example (update)

For each new point

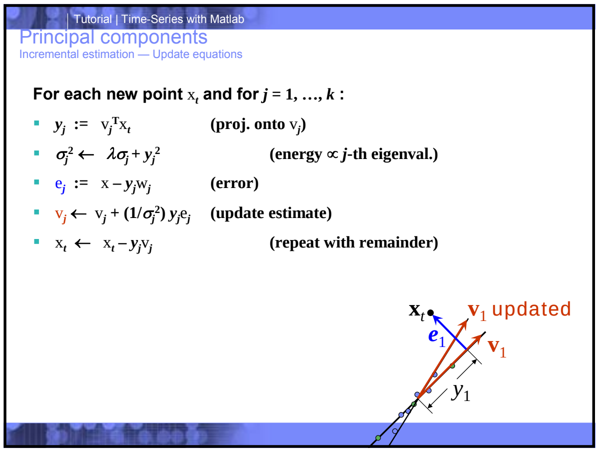
- Project onto current line
- Estimate error
- Rotate line in the direction of the error and in proportion to its magnitude



Principal components

Incremental estimation — Example

- **The “line” is the first *principal component (PC) direction***
- **This line is optimal: it minimizes the sum of squared *projection* errors**



Principal components

Incremental estimation — Update equations

For each new point x_t and for $j = 1, \ldots, k$:

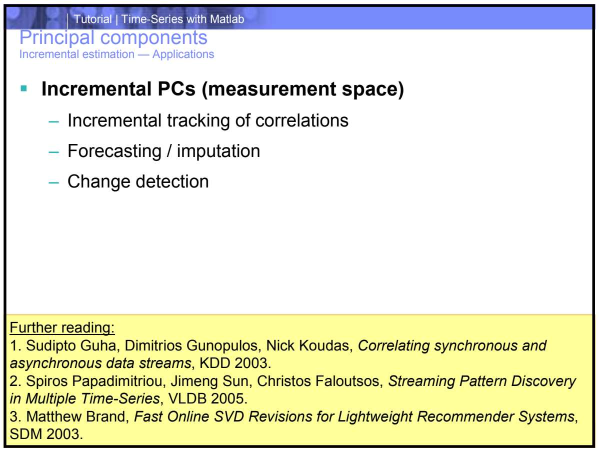
- $y_j := v_j^T x_t$ (proj. onto v_j)
- $\sigma_j^2 \leftarrow \lambda\sigma_j + y_j^2$ (energy $\propto$ j-th eigenval.)
- $e_j := x - y_j w_j$ (error)
- $v_j \leftarrow v_j + (1/\sigma_j^2)\, y_j e_j$ (update estimate)
- $x_t \leftarrow x_t - y_j v_j$ (repeat with remainder)

Principal components

Incremental estimation — Complexity

$O(Mk)$ space (total) and time (per tuple), i.e.,

- **Independent of # points**
- **Linear w.r.t. # streams (M)**
- **Linear w.r.t. # principal components (k)**

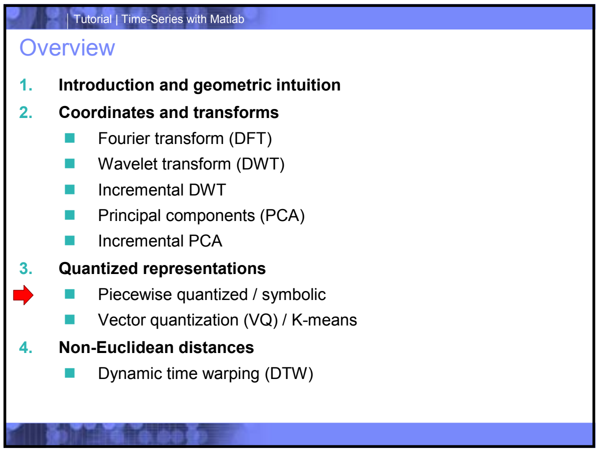


Principal components

Incremental estimation — Applications

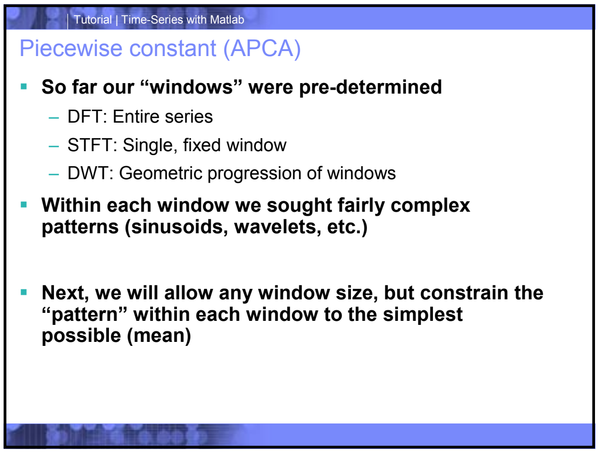
- **Incremental PCs (measurement space)**
  - Incremental tracking of correlations
  - Forecasting / imputation
  - Change detection

Further reading:
1. Sudipto Guha, Dimitrios Gunopulos, Nick Koudas, *Correlating synchronous and asynchronous data streams*, KDD 2003.
2. Spiros Papadimitriou, Jimeng Sun, Christos Faloutsos, *Streaming Pattern Discovery in Multiple Time-Series*, VLDB 2005.
3. Matthew Brand, *Fast Online SVD Revisions for Lightweight Recommender Systems*, SDM 2003.



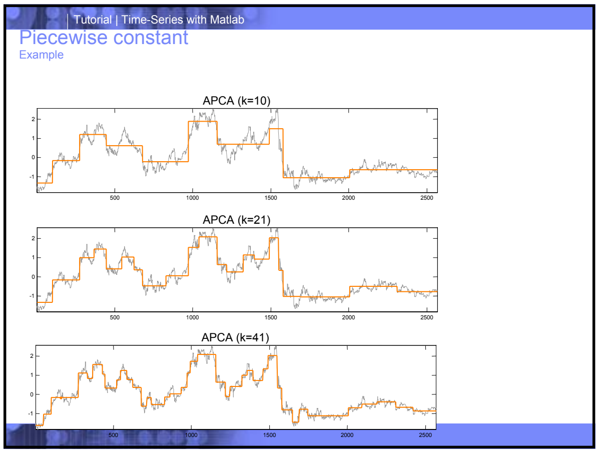
Overview

1. **Introduction and geometric intuition**
2. **Coordinates and transforms**
   - Fourier transform (DFT)
   - Wavelet transform (DWT)
   - Incremental DWT
   - Principal components (PCA)
   - Incremental PCA
3. **Quantized representations**
   - ➡ Piecewise quantized / symbolic
   - Vector quantization (VQ) / K-means
4. **Non-Euclidean distances**
   - Dynamic time warping (DTW)



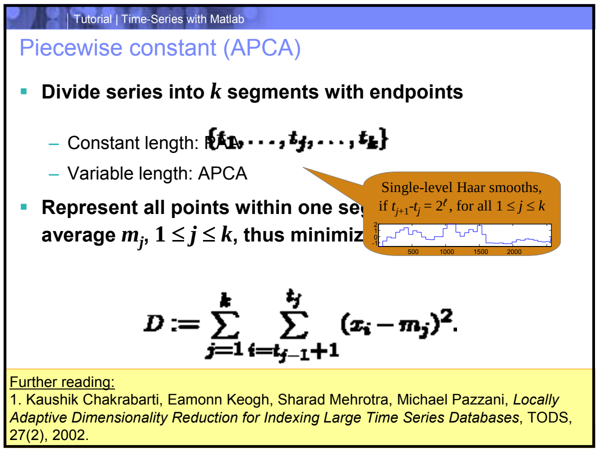
Piecewise constant (APCA)

- **So far our “windows” were pre-determined**
  - DFT: Entire series
  - STFT: Single, fixed window
  - DWT: Geometric progression of windows
- **Within each window we sought fairly complex patterns (sinusoids, wavelets, etc.)**
- **Next, we will allow any window size, but constrain the “pattern” within each window to the simplest possible (mean)**



Piecewise constant

Example



Piecewise constant (APCA)

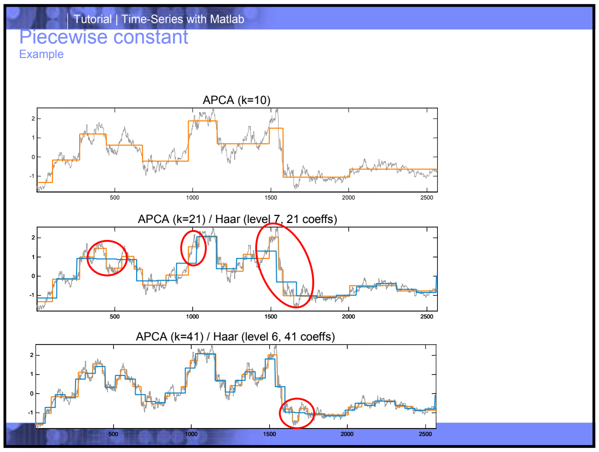
- **Divide series into k segments with endpoints**
  - Constant length: PAA $\{t_1, \ldots, t_j, \ldots, t_k\}$
  - Variable length: APCA
- **Represent all points within one seg… average m_j, $1 \le j \le k$, thus minimiz…**

Single-level Haar smooths, if t_{j+1}-$t_j = 2^\ell$, for all $1 \le j \le k$

2
1
0
-1
500
1000
1500
2000

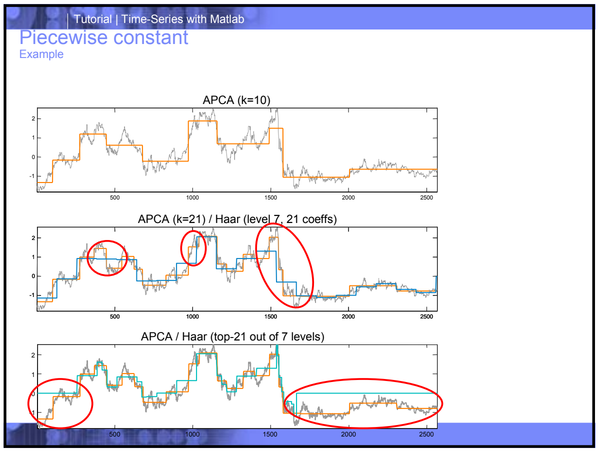
$$D := \sum_{j=1}^{k} \sum_{i=t_{j-1}+1}^{t_j} (x_i - m_j)^2.$$

Further reading:
1. Kaushik Chakrabarti, Eamonn Keogh, Sharad Mehrotra, Michael Pazzani, *Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases*, TODS, 27(2), 2002.



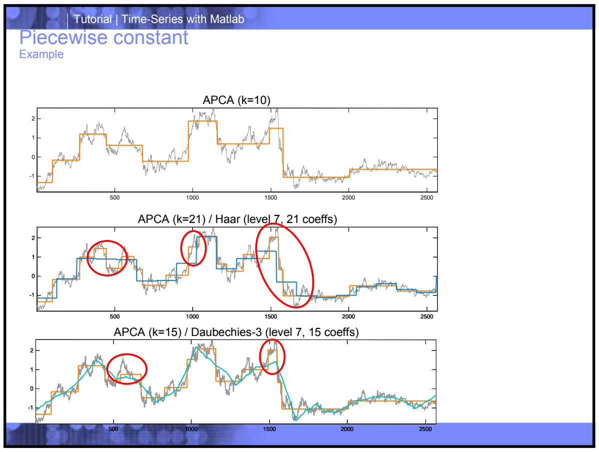
Piecewise constant

Example



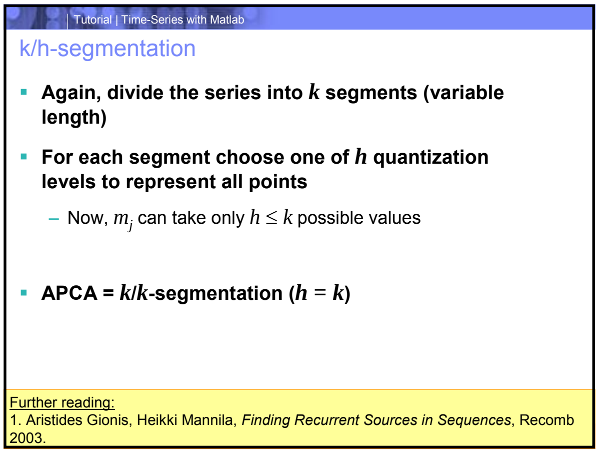
Piecewise constant

Example



Piecewise constant

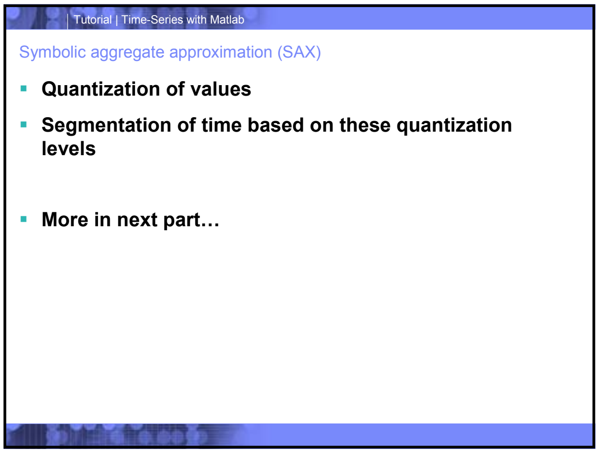
Example



k/h-segmentation

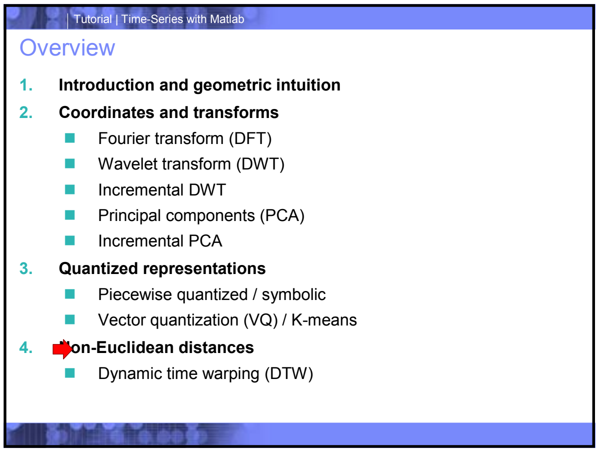
- **Again, divide the series into k segments (variable length)**
- **For each segment choose one of h quantization levels to represent all points**
  - Now, m_j can take only $h \leq k$ possible values
- **APCA = k/k-segmentation ($h = k$)**

Further reading:
1. Aristides Gionis, Heikki Mannila, *Finding Recurrent Sources in Sequences*, Recomb 2003.



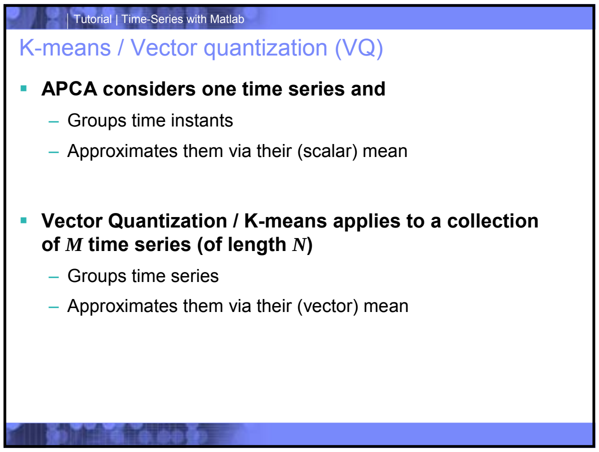
Symbolic aggregate approximation (SAX)

- **Quantization of values**
- **Segmentation of time based on these quantization levels**
- **More in next part…**



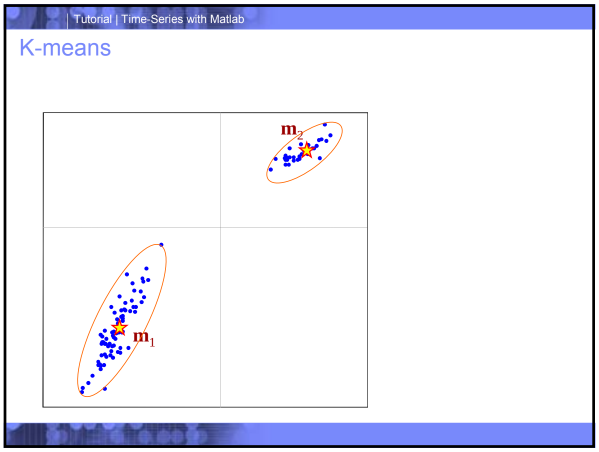
Overview

1. **Introduction and geometric intuition**
2. **Coordinates and transforms**
   - Fourier transform (DFT)
   - Wavelet transform (DWT)
   - Incremental DWT
   - Principal components (PCA)
   - Incremental PCA
3. **Quantized representations**
   - Piecewise quantized / symbolic
   - Vector quantization (VQ) / K-means
4. ➡ **Non-Euclidean distances**
   - Dynamic time warping (DTW)

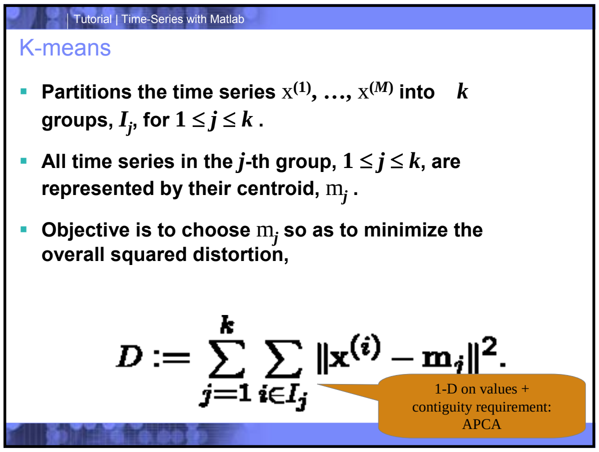


K-means / Vector quantization (VQ)

- **APCA considers one time series and**
  - Groups time instants
  - Approximates them via their (scalar) mean
- **Vector Quantization / K-means applies to a collection of M time series (of length N)**
  - Groups time series
  - Approximates them via their (vector) mean



K-means

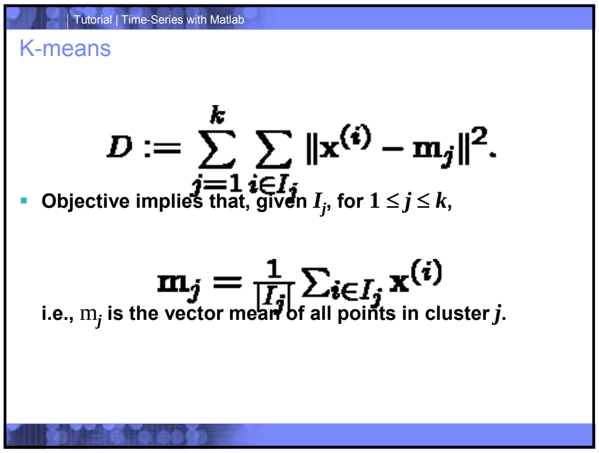


K-means

- Partitions the time series $x^{(1)}, \ldots, x^{(M)}$ into k groups, I_j, for $1 \leq j \leq k$.
- All time series in the j-th group, $1 \leq j \leq k$, are represented by their centroid, m_j .
- Objective is to choose m_j so as to minimize the overall squared distortion,

$$D := \sum_{j=1}^{k} \sum_{i \in I_j} \|\mathbf{x}^{(i)} - \mathbf{m}_j\|^2.$$

1-D on values + contiguity requirement: APCA



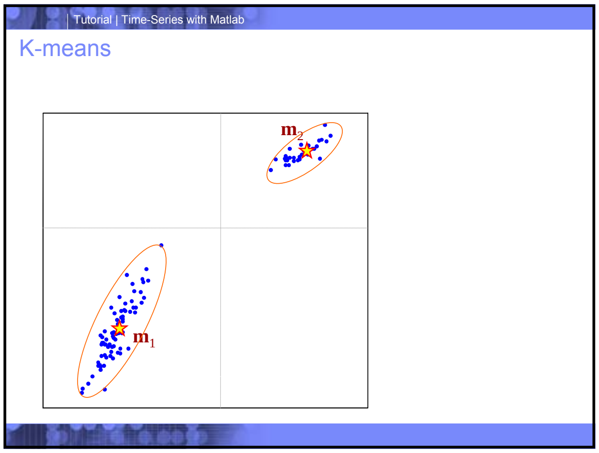
K-means

$$D := \sum_{j=1}^{k} \sum_{i \in I_j} \|\mathbf{x}^{(i)} - \mathbf{m}_j\|^2.$$

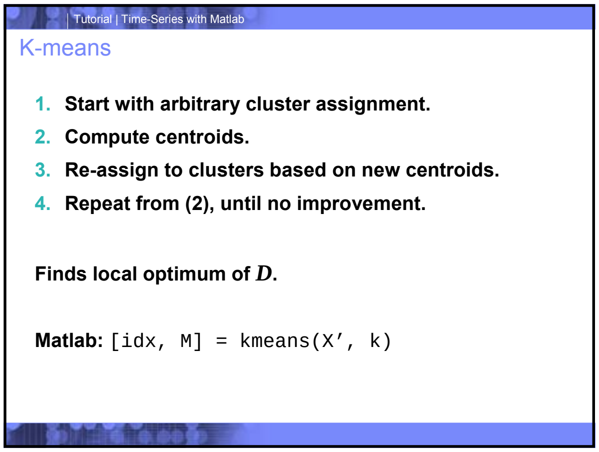
- Objective implies that, given I_j, for $1 \leq j \leq k$,

$$\mathbf{m}_j = \frac{1}{|I_j|} \sum_{i \in I_j} \mathbf{x}^{(i)}$$

i.e., m_j is the vector mean of all points in cluster j.



K-means

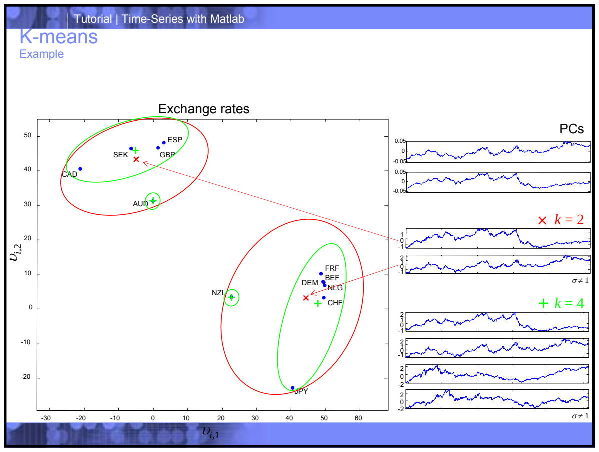


K-means

1. Start with arbitrary cluster assignment.
2. Compute centroids.
3. Re-assign to clusters based on new centroids.
4. Repeat from (2), until no improvement.

Finds local optimum of D.

Matlab: `[idx, M] = kmeans(X’, k)`



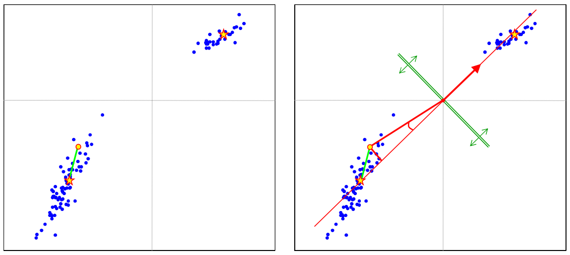
K-means

Example

K-means in other coordinates

- An orthonormal transform (e.g., DFT, DWT, PCA) preserves distances.
- K-means can be applied in any of these "coordinate systems."
- Can transform data to speed up distance computations (if N large)

K-means and PCA



Further reading:
1. Hongyuan Zha, Xiaofeng He, Chris H.Q. Ding, Ming Gu, Horst D. Simon, *Spectral Relaxation for K-means Clustering*, NIPS 2001.

Overview

1. **Introduction and geometric intuition**
2. **Coordinates and transforms**
   - Fourier transform (DFT)
   - Wavelet transform (DWT)
   - Incremental DWT
   - Principal components (PCA)
   - Incremental PCA
3. **Quantized representations**
   - Piecewise quantized / symbolic
   - Vector quantization (VQ) / K-means
4. **Non-Euclidean distances**
   - ➡ Dynamic time warping (DTW)

Dynamic time warping (DTW)

- So far we have been discussing shapes via various kinds of "features" or "descriptions" (bases)
- However, the series were always fixed
- Dynamic time warping:
  - Allows local deformations (stretch/shrink)
  - Can thus also handle series of different lengths

Dynamic time warping (DTW)

- Euclidean (L2) distance is $D_2(\mathbf{x},\mathbf{y}) := \sum_{i=1}^{N}(x_i - y_i)^2$

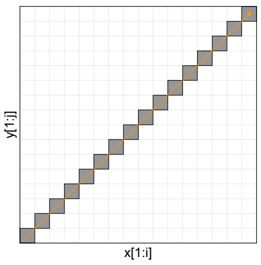
  or, recursively, $D_2(\mathbf{x}_{1:i}, \mathbf{y}_{1:i}) := (x_i - y_i)^2 + D_2(\mathbf{x}_{1:(i-1)}, \mathbf{y}_{1:(i-1)})$.

- Dynamic time warping distance is

$$D_w(\mathbf{x}_{1:i}, \mathbf{y}_{1:j}) := (x_i - x_j)^2 + \min\begin{cases} D_w(\mathbf{x}_{1:(i-1)}, \mathbf{y}_{1:(j-1)}) \\ D_w(\mathbf{x}_{1:(i-1)}, \mathbf{y}_{1:j}) & \leftarrow \text{shrink } \mathbf{x} \text{ / stretch } \mathbf{y} \\ D_w(\mathbf{x}_{1:i}, \mathbf{y}_{1:(j-1)}) & \leftarrow \text{stretch } \mathbf{x} \text{ / shrink } \mathbf{y} \end{cases}$$

  where $x_{1:i}$ is the subsequence $(x_1, \ldots, x_i)$

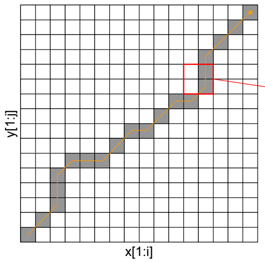
Dynamic time warping (DTW)

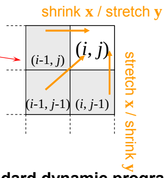


- Each cell $c = (i,j)$ is a pair of indices whose corresponding values will be compared, $(x_i - y_j)^2$, and included in the sum for the distance
- Euclidean path:
  - $i = j$ always
  - Ignores off-diagonal cells

Dynamic time warping (DTW)

- DTW allows any path
- Examine all paths:





- Standard dynamic programming to fill in table—top-right cell contains final result

Dynamic time-warping

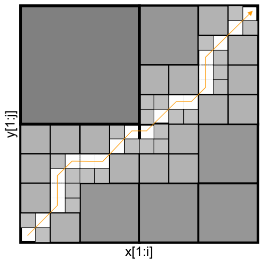
Fast estimation

- **Standard dynamic programming: $O(N^2)$**
- **Envelope-based technique**
  - Introduced by [Keogh 2000 & 2002]
  - Multi-scale, wavelet-like technique and formalism by [Salvador et al. 2004] and, independently, by [Sakurai et al. 2005]

Further reading:
1. Eamonn J. Keogh, *Exact Indexing of Dynamic Time Warping*, VLDB 2002.
2. Stan Salvador, Philip Chan, *FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space*, TDM 2004.
3. Yasushi Sakurai, Masatoshi Yoshikawa, Christos Faloutsos, *FTW: Fast Similarity Under the Time Warping Distance*, PODS 2005.

Dynamic time warping

Fast estimation — Summary



- **Create lower-bounding distance on coarser granularity, either at**
  - Single scale
  - Multiple scales

Use to prune search space

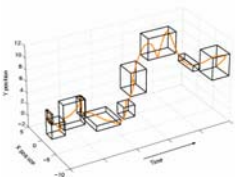
Non-Euclidean metrics

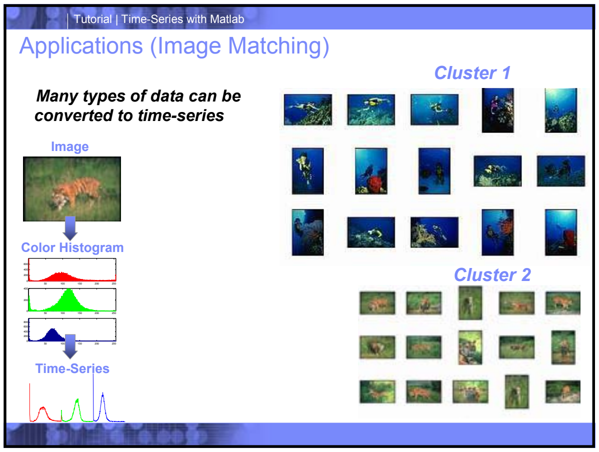
- **More in part 3**

PART C: Similarity Search and Applications

Timeline of part C

- Introduction
- Time-Series Representations
- Distance Measures
- Lower Bounding
- Clustering/Classification/Visualization
- Applications

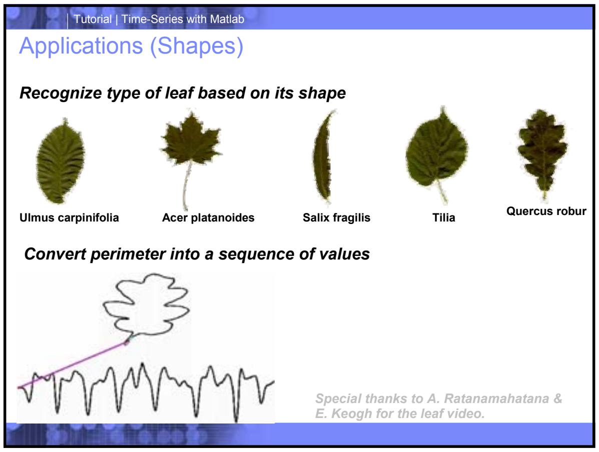




Tutorial | Time-Series with Matlab

Applications (Image Matching)

Many types of data can be converted to time-series



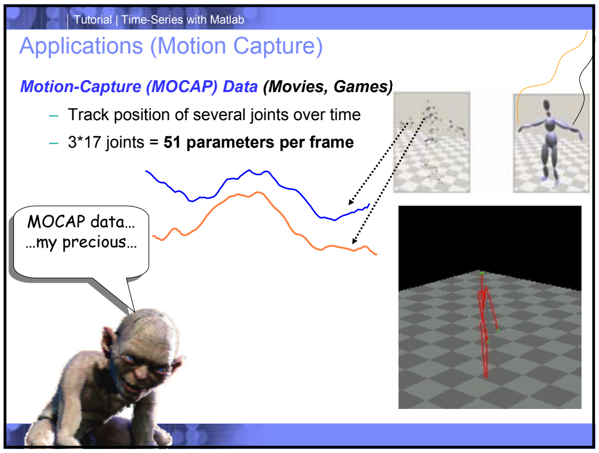
Tutorial | Time-Series with Matlab

Applications (Shapes)

Recognize type of leaf based on its shape

Convert perimeter into a sequence of values

Special thanks to A. Ratanamahatana & E. Keogh for the leaf video.

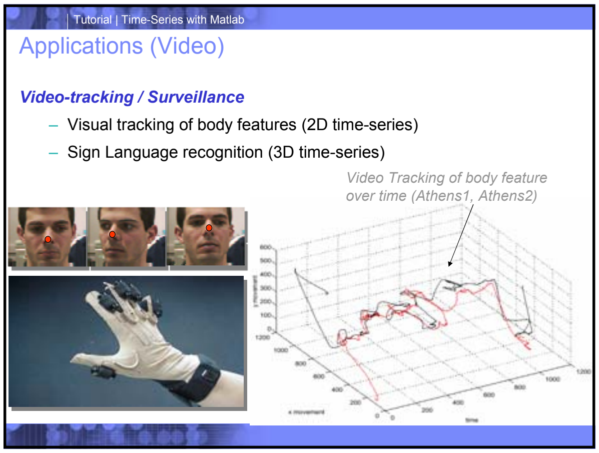


Tutorial | Time-Series with Matlab

Applications (Motion Capture)

Motion-Capture (MOCAP) Data **(Movies, Games)**

- Track position of several joints over time
- 3*17 joints = **51 parameters per frame**



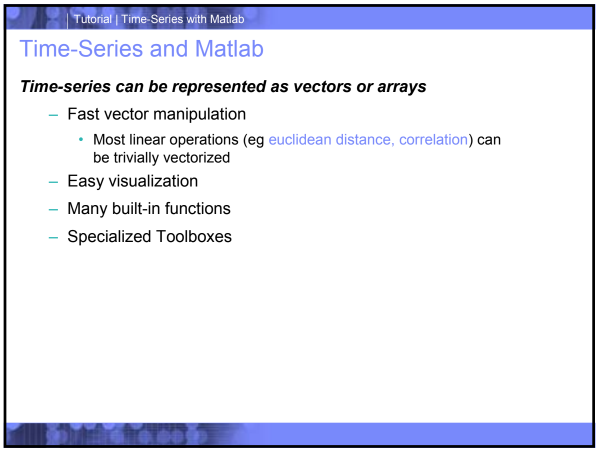
Tutorial | Time-Series with Matlab

Applications (Video)

Video-tracking / Surveillance

- Visual tracking of body features (2D time-series)
- Sign Language recognition (3D time-series)

Video Tracking of body feature over time (Athens1, Athens2)

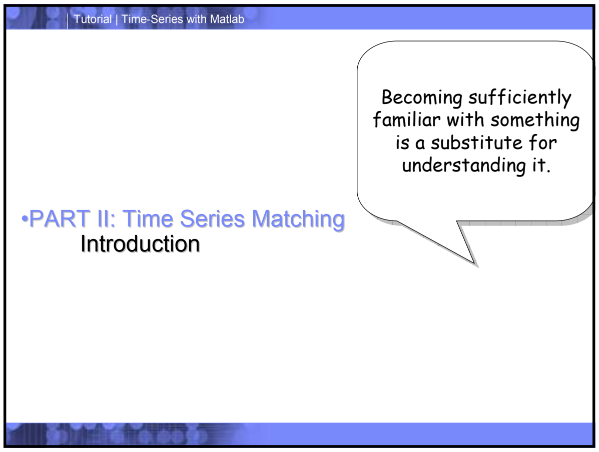


Tutorial | Time-Series with Matlab

Time-Series and Matlab

Time-series can be represented as vectors or arrays

- Fast vector manipulation
  - Most linear operations (eg euclidean distance, correlation) can be trivially vectorized
- Easy visualization
- Many built-in functions
- Specialized Toolboxes

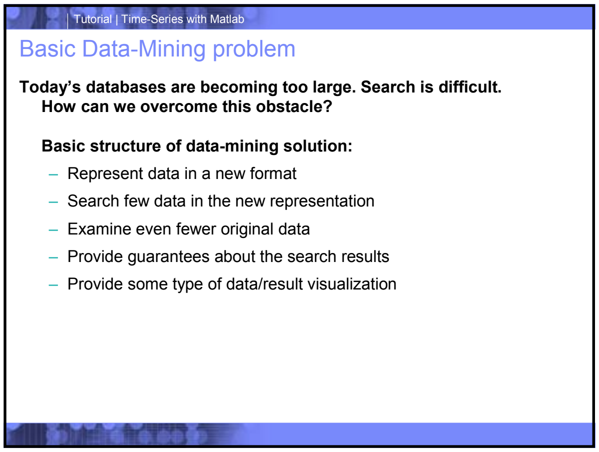


Tutorial | Time-Series with Matlab

Becoming sufficiently familiar with something is a substitute for understanding it.

•PART II: Time Series Matching

Introduction



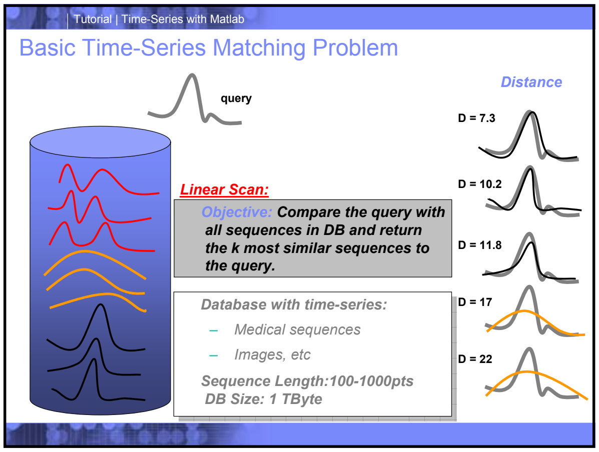
Tutorial | Time-Series with Matlab

Basic Data-Mining problem

Today's databases are becoming too large. Search is difficult. How can we overcome this obstacle?

Basic structure of data-mining solution:

- Represent data in a new format
- Search few data in the new representation
- Examine even fewer original data
- Provide guarantees about the search results
- Provide some type of data/result visualization



Tutorial | Time-Series with Matlab

Basic Time-Series Matching Problem

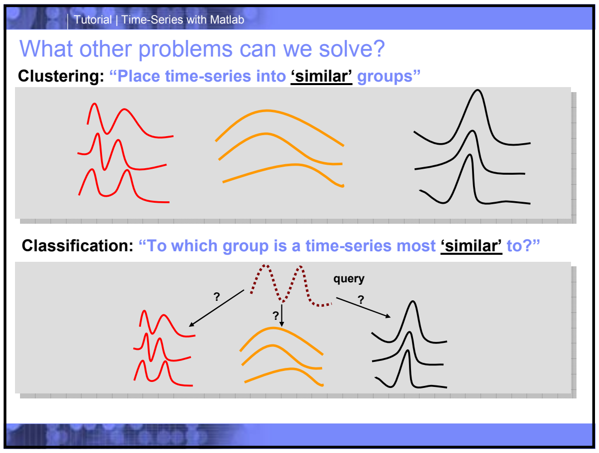
Linear Scan:

Objective: ***Compare the query with all sequences in DB and return the k most similar sequences to the query.***

Database with time-series:

- *Medical sequences*
- *Images, etc*

Sequence Length:100-1000pts
DB Size: 1 TByte

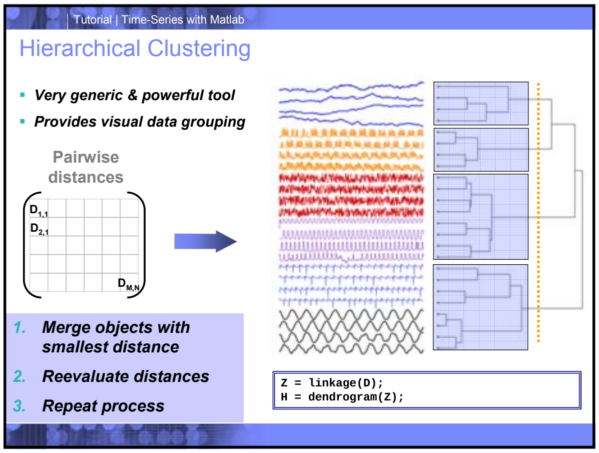


Tutorial | Time-Series with Matlab

What other problems can we solve?

Clustering: "Place time-series into 'similar' groups"

Classification: "To which group is a time-series most 'similar' to?"



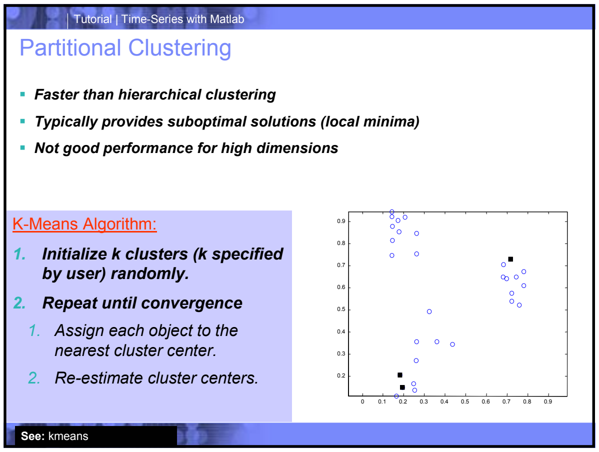
Tutorial | Time-Series with Matlab

Hierarchical Clustering

- ***Very generic & powerful tool***
- ***Provides visual data grouping***

1. ***Merge objects with smallest distance***
2. ***Reevaluate distances***
3. ***Repeat process***

```
Z = linkage(D);
H = dendrogram(Z);
```



Tutorial | Time-Series with Matlab

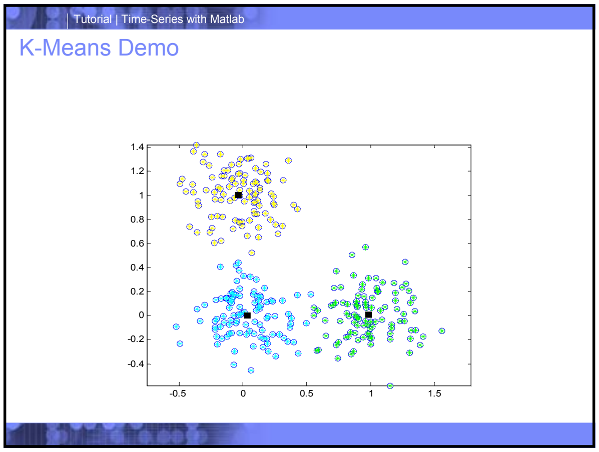
Partitional Clustering

- ***Faster than hierarchical clustering***
- ***Typically provides suboptimal solutions (local minima)***
- ***Not good performance for high dimensions***

K-Means Algorithm:

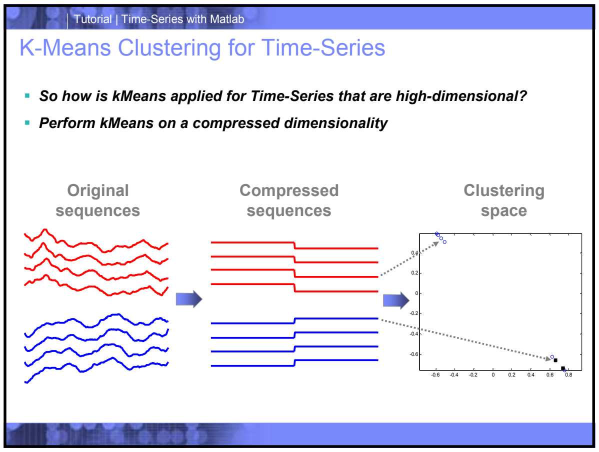
1. ***Initialize k clusters (k specified by user) randomly.***
2. ***Repeat until convergence***
   1. *Assign each object to the nearest cluster center.*
   2. *Re-estimate cluster centers.*

See: kmeans



Tutorial | Time-Series with Matlab

K-Means Demo



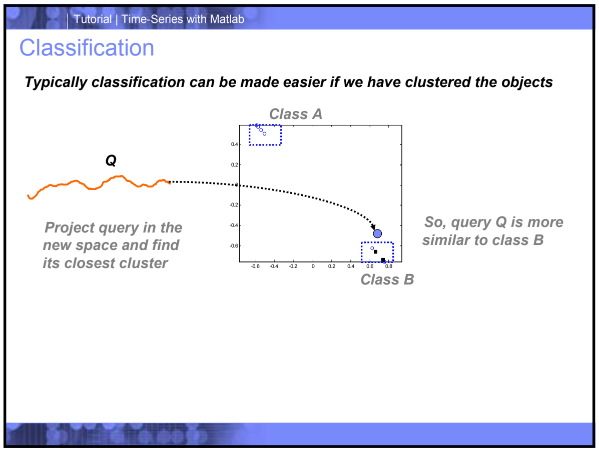
K-Means Clustering for Time-Series

- ***So how is kMeans applied for Time-Series that are high-dimensional?***
- ***Perform kMeans on a compressed dimensionality***

Original sequences

Compressed sequences

Clustering space



Classification

Typically classification can be made easier if we have clustered the objects

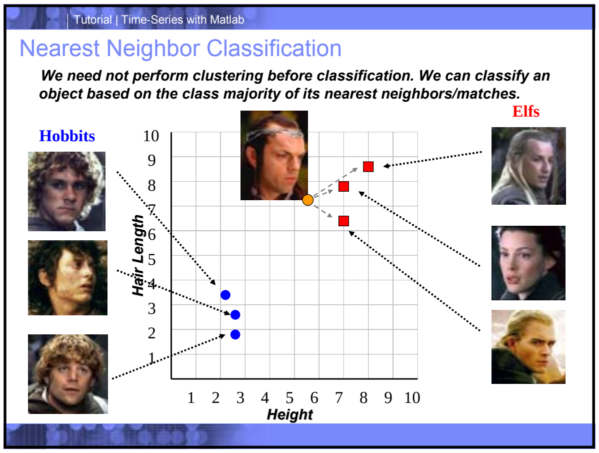
Class A

Q

Project query in the new space and find its closest cluster

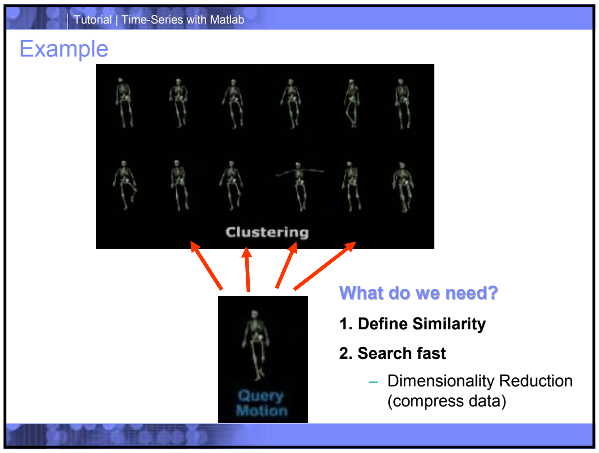
So, query Q is more similar to class B

Class B



Nearest Neighbor Classification

We need not perform clustering before classification. We can classify an object based on the class majority of its nearest neighbors/matches.



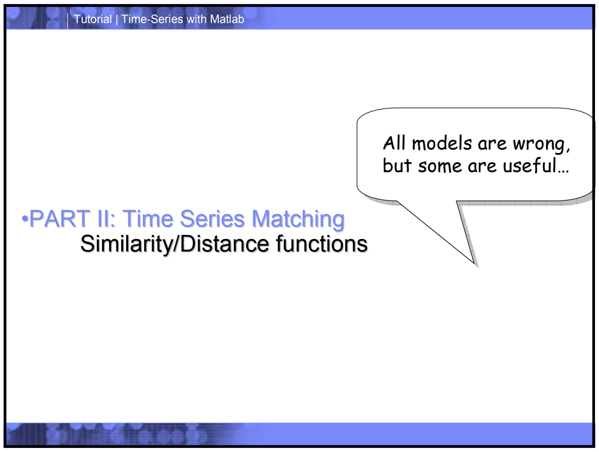
Example

What do we need?

1. Define Similarity

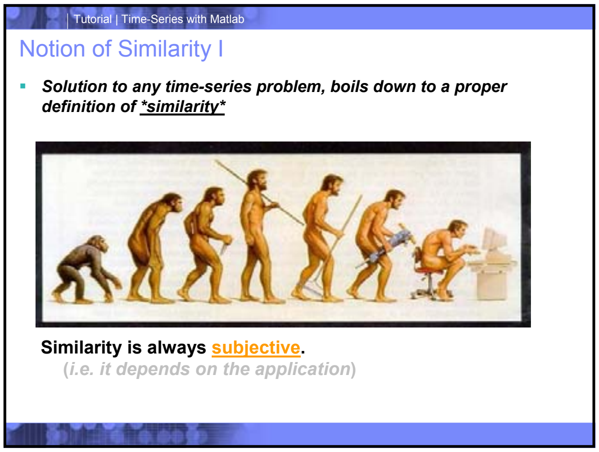
2. Search fast

- Dimensionality Reduction (compress data)



All models are wrong, but some are useful…

•PART II: Time Series Matching
Similarity/Distance functions

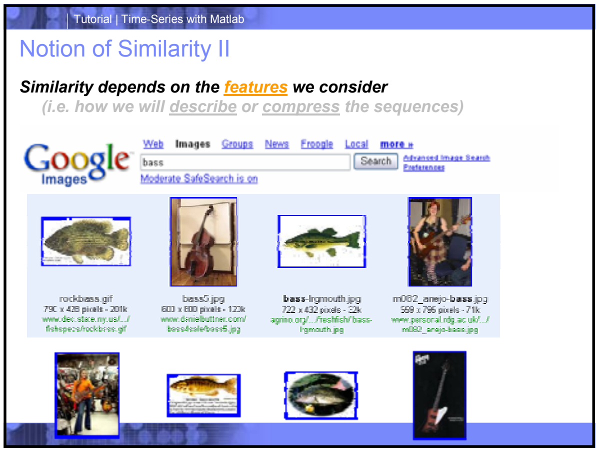


Notion of Similarity I

- ***Solution to any time-series problem, boils down to a proper definition of *similarity****

Similarity is always subjective.

(i.e. it depends on the application)

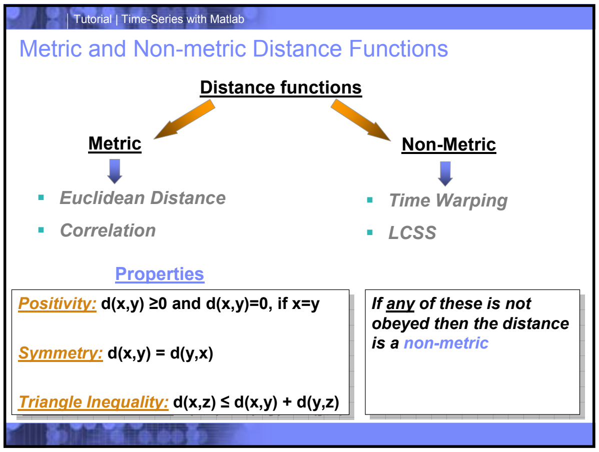


Notion of Similarity II

Similarity depends on the features we consider

(i.e. how we will describe or compress the sequences)

Google Images
Web Images Groups News Froogle Local more »
bass
Search
Advanced Image Search
Preferences
Moderate SafeSearch is on
rockbass.gif
790 x 438 pixels - 201k
www.dec.state.ny.us/.../fishspecs/rockbass.gif
bass5.jpg
600 x 800 pixels - 120k
www.danielbuttner.com/bass4sale/bass5.jpg
bass-lrgmouth.jpg
722 x 432 pixels - 32k
agrino.org/.../freshfish/bass-lrgmouth.jpg
m082_anejo-bass.jpg
559 x 796 pixels - 71k
www.personal.rdg.ac.uk/.../m082_anejo-bass.jpg



Metric and Non-metric Distance Functions

Distance functions

Metric

- *Euclidean Distance*
- *Correlation*

Non-Metric

- *Time Warping*
- *LCSS*

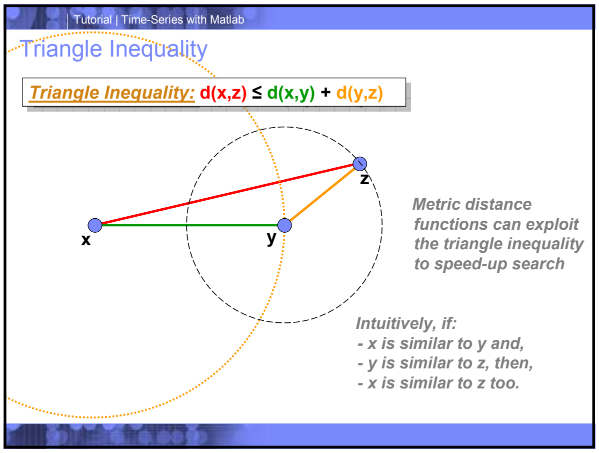
Properties

Positivity: d(x,y) ≥0 and d(x,y)=0, if x=y

Symmetry: d(x,y) = d(y,x)

Triangle Inequality: d(x,z) ≤ d(x,y) + d(y,z)

If any of these is not obeyed then the distance is a non-metric

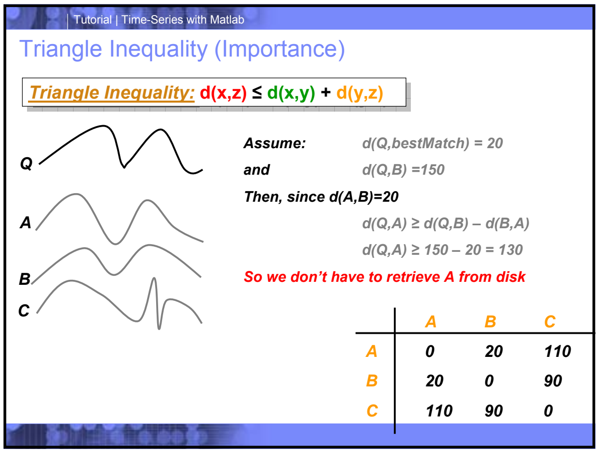


Triangle Inequality

Triangle Inequality: d(x,z) ≤ d(x,y) + d(y,z)

Metric distance functions can exploit the triangle inequality to speed-up search

Intuitively, if:
- x is similar to y and,
- y is similar to z, then,
- x is similar to z too.



Triangle Inequality (Importance)

Triangle Inequality: d(x,z) ≤ d(x,y) + d(y,z)

Assume: *d(Q,bestMatch) = 20*

and *d(Q,B) =150*

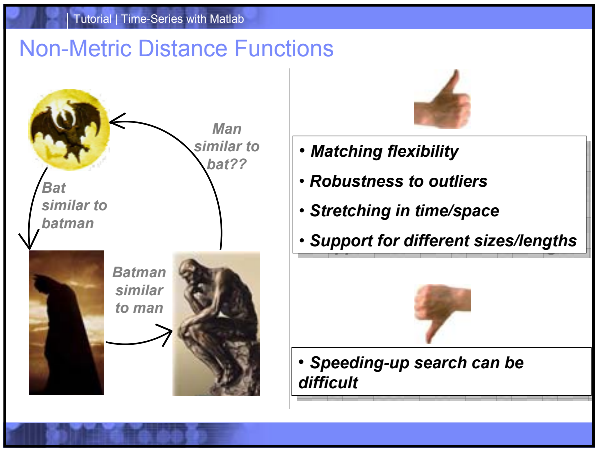
Then, since d(A,B)=20

d(Q,A) ≥ d(Q,B) – d(B,A)

d(Q,A) ≥ 150 – 20 = 130

So we don't have to retrieve A from disk

	A	B	C
A	0	20	110
B	20	0	90
C	110	90	0



Non-Metric Distance Functions

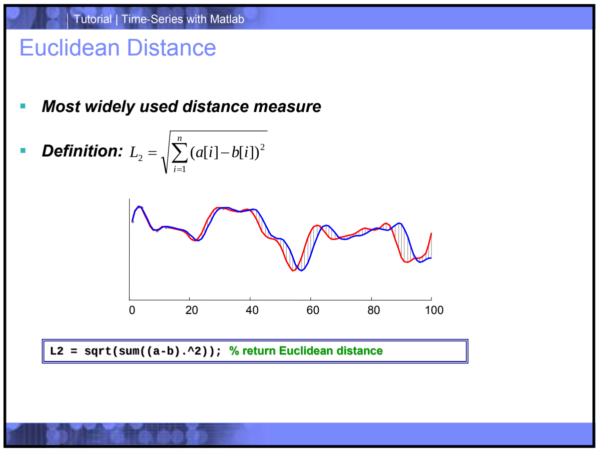
Man similar to bat??

Bat similar to batman

Batman similar to man

- ***Matching flexibility***
- ***Robustness to outliers***
- ***Stretching in time/space***
- ***Support for different sizes/lengths***

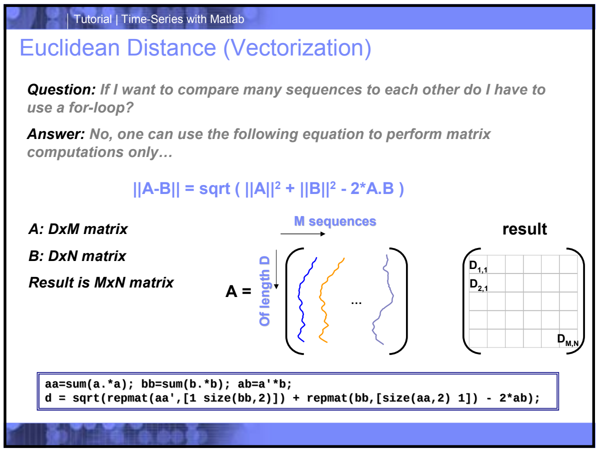
- ***Speeding-up search can be difficult***



Euclidean Distance

- ***Most widely used distance measure***
- ***Definition:*** $L_2 = \sqrt{\sum_{i=1}^{n}(a[i]-b[i])^2}$

```
L2 = sqrt(sum((a-b).^2)); % return Euclidean distance
```



Euclidean Distance (Vectorization)

Question: *If I want to compare many sequences to each other do I have to use a for-loop?*

Answer: *No, one can use the following equation to perform matrix computations only…*

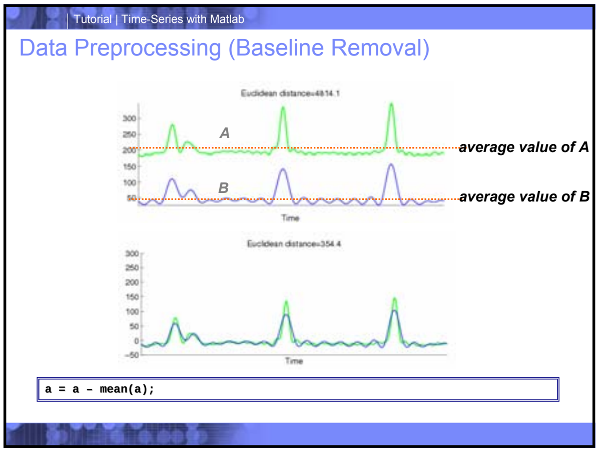
$$||A-B|| = sqrt (||A||^2 + ||B||^2 - 2*A.B)$$

A: DxM matrix

B: DxN matrix

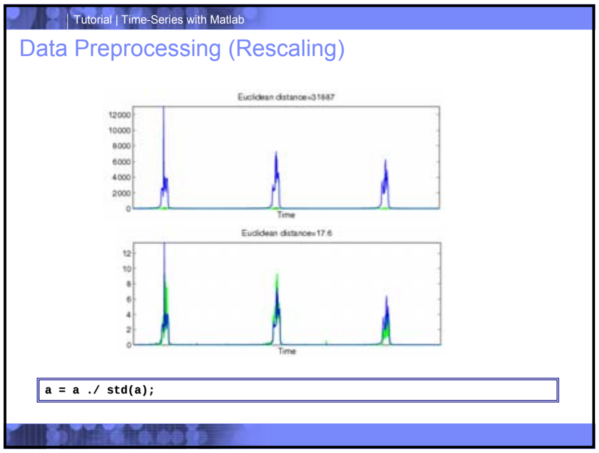
Result is MxN matrix

```
aa=sum(a.*a); bb=sum(b.*b); ab=a'*b;
d = sqrt(repmat(aa',[1 size(bb,2)]) + repmat(bb,[size(aa,2) 1]) - 2*ab);
```



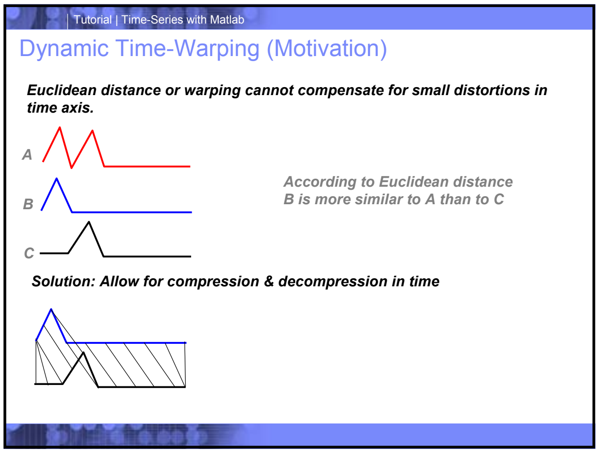
Data Preprocessing (Baseline Removal)

```
a = a – mean(a);
```



Data Preprocessing (Rescaling)

```
a = a ./ std(a);
```

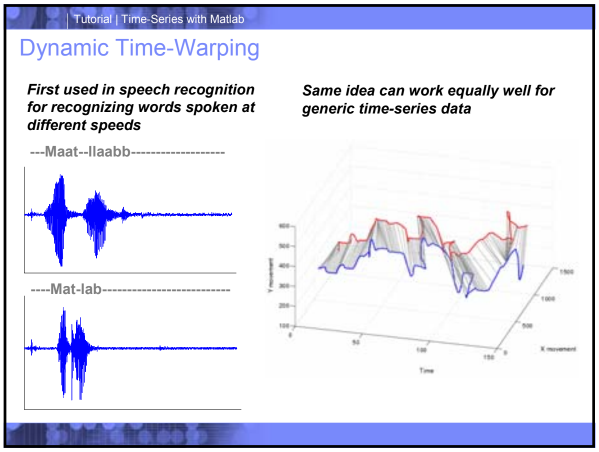


Dynamic Time-Warping (Motivation)

Euclidean distance or warping cannot compensate for small distortions in time axis.

According to Euclidean distance B is more similar to A than to C

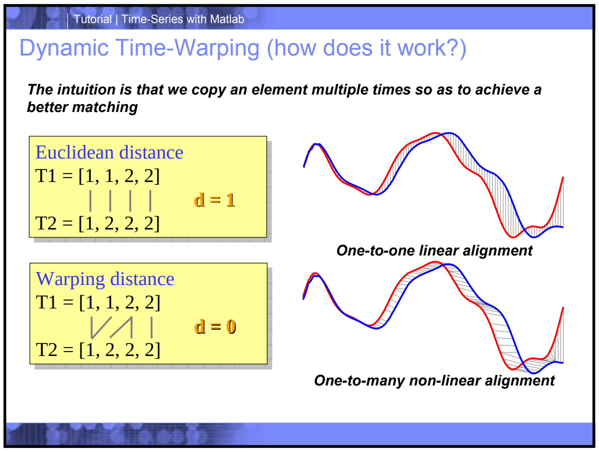
Solution: Allow for compression & decompression in time



Dynamic Time-Warping

First used in speech recognition for recognizing words spoken at different speeds

Same idea can work equally well for generic time-series data



Dynamic Time-Warping (how does it work?)

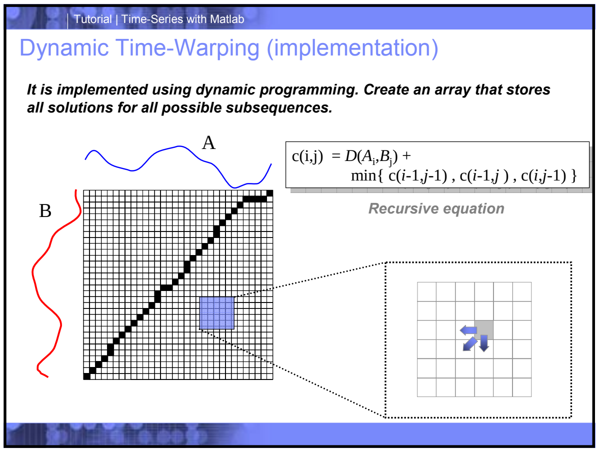
The intuition is that we copy an element multiple times so as to achieve a better matching

Euclidean distance
T1 = [1, 1, 2, 2]
| | | | d = 1
T2 = [1, 2, 2, 2]

Warping distance
T1 = [1, 1, 2, 2]
d = 0
T2 = [1, 2, 2, 2]

One-to-one linear alignment

One-to-many non-linear alignment

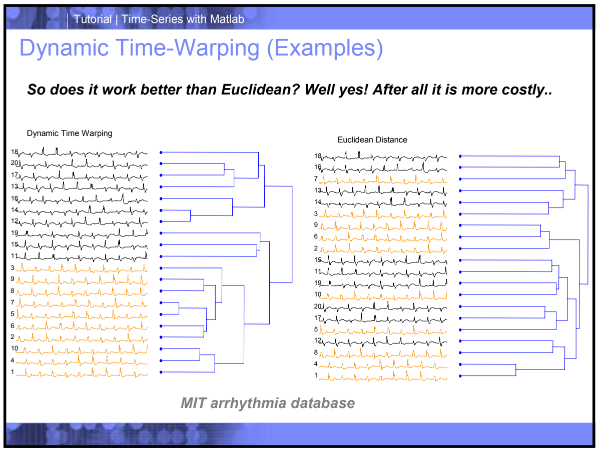


Dynamic Time-Warping (implementation)

It is implemented using dynamic programming. Create an array that stores all solutions for all possible subsequences.

$$c(i,j) = D(A_i,B_j) + \min\{\, c(i\text{-}1,j\text{-}1) , c(i\text{-}1,j) , c(i,j\text{-}1) \,\}$$

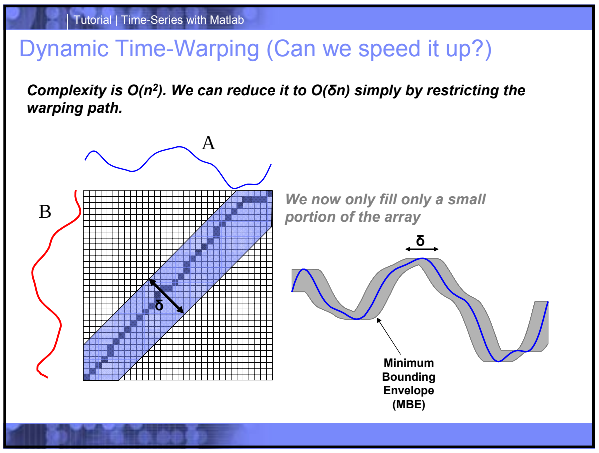
Recursive equation



Dynamic Time-Warping (Examples)

So does it work better than Euclidean? Well yes! After all it is more costly..

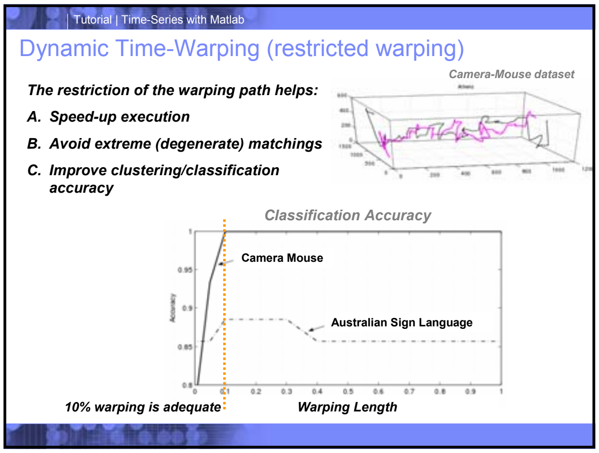
MIT arrhythmia database



Dynamic Time-Warping (Can we speed it up?)

Complexity is O(n²). We can reduce it to O(δn) simply by restricting the warping path.

We now only fill only a small portion of the array



Dynamic Time-Warping (restricted warping)

The restriction of the warping path helps:

A. Speed-up execution

B. Avoid extreme (degenerate) matchings

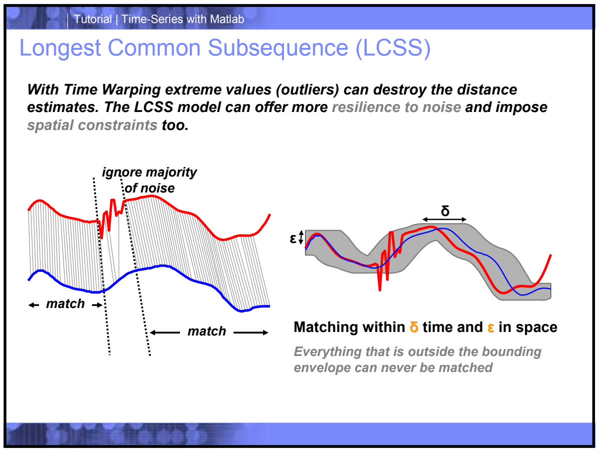
C. Improve clustering/classification accuracy

Camera-Mouse dataset

Classification Accuracy

10% warping is adequate

Warping Length

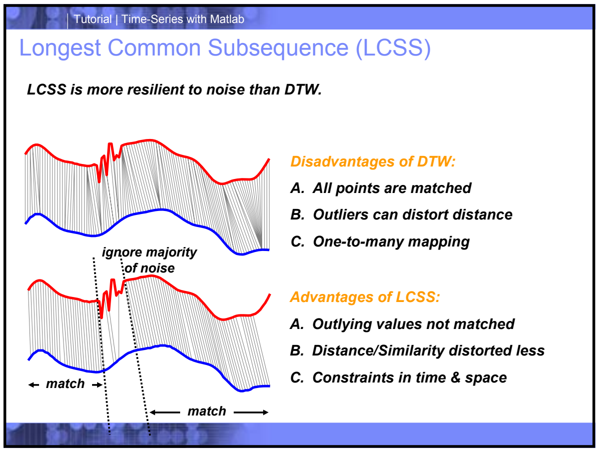


Longest Common Subsequence (LCSS)

With Time Warping extreme values (outliers) can destroy the distance estimates. The LCSS model can offer more resilience to noise and impose spatial constraints too.

Matching within δ time and ε in space

Everything that is outside the bounding envelope can never be matched



Longest Common Subsequence (LCSS)

LCSS is more resilient to noise than DTW.

Disadvantages of DTW:

A. All points are matched

B. Outliers can distort distance

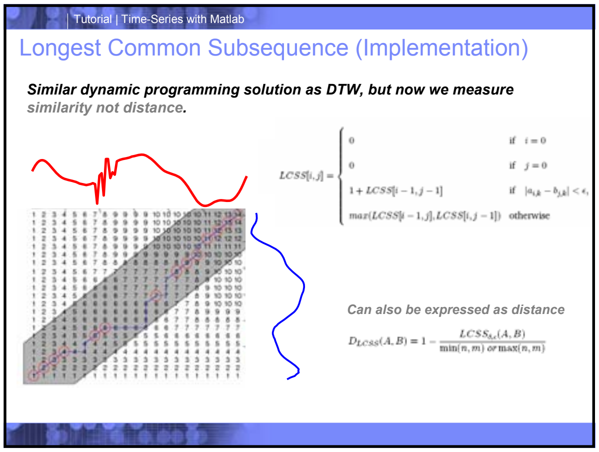
C. One-to-many mapping

Advantages of LCSS:

A. Outlying values not matched

B. Distance/Similarity distorted less

C. Constraints in time & space



Tutorial | Time-Series with Matlab

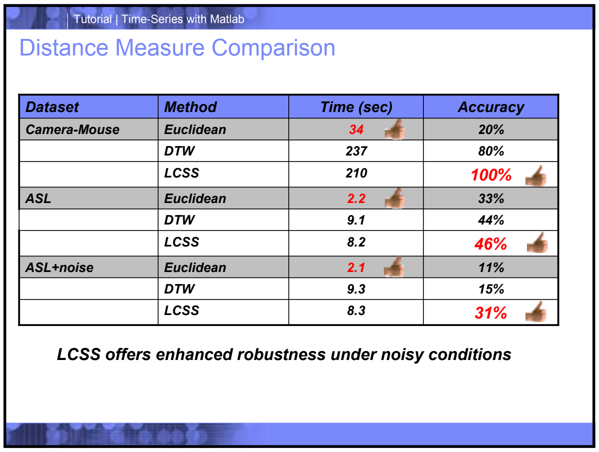
Longest Common Subsequence (Implementation)

Similar dynamic programming solution as DTW, but now we measure similarity not distance.

$$LCSS[i,j] = \begin{cases} 0 & \text{if } \; i = 0 \\ 0 & \text{if } \; j = 0 \\ 1 + LCSS[i-1, j-1] & \text{if } \; |a_{i,k} - b_{j,k}| < \epsilon, \\ max(LCSS[i-1,j], LCSS[i, j-1]) & \text{otherwise} \end{cases}$$

Can also be expressed as distance

$$D_{LCSS}(A,B) = 1 - \frac{LCSS_{\delta,\epsilon}(A,B)}{\min(n,m) \; or \; \max(n,m)}$$

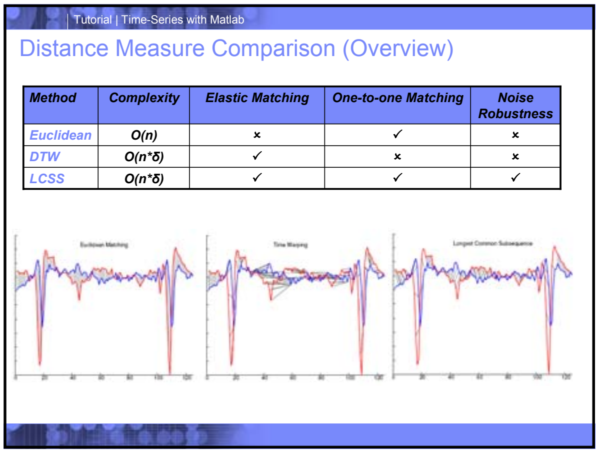


Tutorial | Time-Series with Matlab

Distance Measure Comparison

Dataset	Method	Time (sec)	Accuracy
Camera-Mouse	Euclidean	34	20%
	DTW	237	80%
	LCSS	210	100%
ASL	Euclidean	2.2	33%
	DTW	9.1	44%
	LCSS	8.2	46%
ASL+noise	Euclidean	2.1	11%
	DTW	9.3	15%
	LCSS	8.3	31%

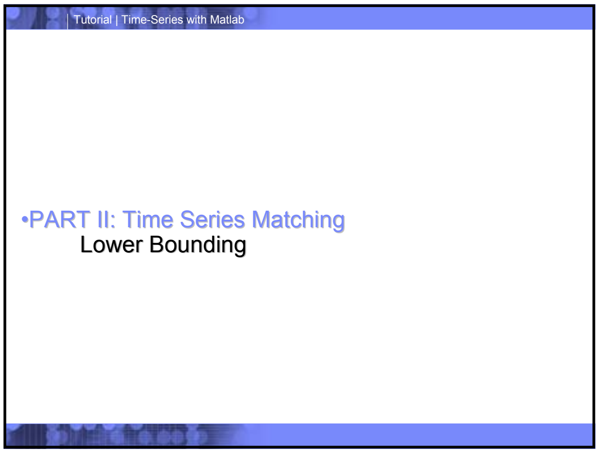
LCSS offers enhanced robustness under noisy conditions



Tutorial | Time-Series with Matlab

Distance Measure Comparison (Overview)

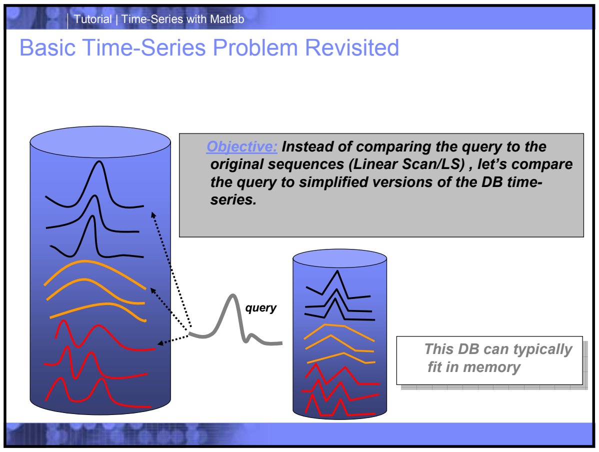
Method	Complexity	Elastic Matching	One-to-one Matching	Noise Robustness
Euclidean	O(n)	×	✓	×
DTW	O(n*δ)	✓	×	×
LCSS	O(n*δ)	✓	✓	✓



Tutorial | Time-Series with Matlab

•PART II: Time Series Matching

Lower Bounding

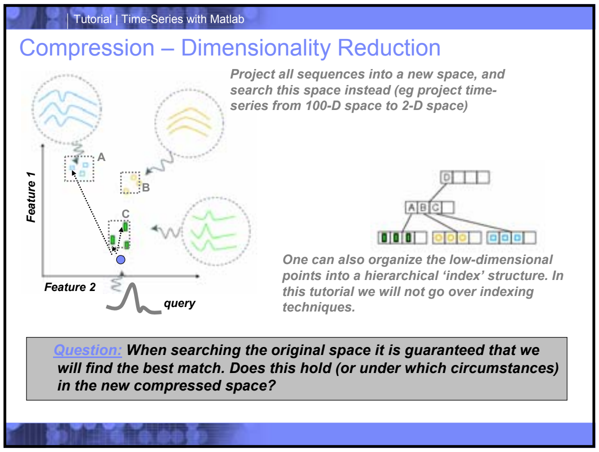


Tutorial | Time-Series with Matlab

Basic Time-Series Problem Revisited

Objective: ***Instead of comparing the query to the original sequences (Linear Scan/LS) , let's compare the query to simplified versions of the DB time-series.***

This DB can typically fit in memory



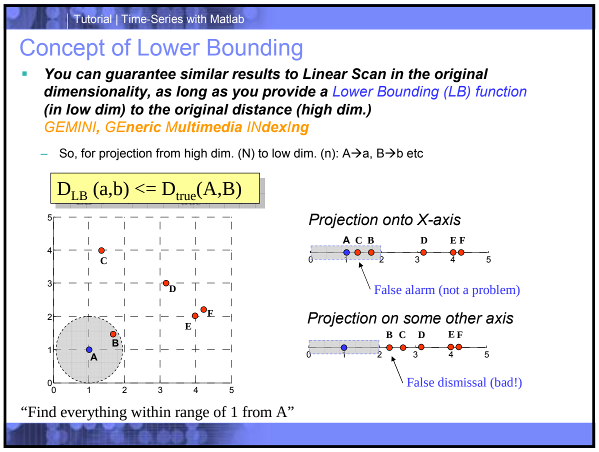
Tutorial | Time-Series with Matlab

Compression – Dimensionality Reduction

Project all sequences into a new space, and search this space instead (eg project time-series from 100-D space to 2-D space)

One can also organize the low-dimensional points into a hierarchical 'index' structure. In this tutorial we will not go over indexing techniques.

Question: ***When searching the original space it is guaranteed that we will find the best match. Does this hold (or under which circumstances) in the new compressed space?***



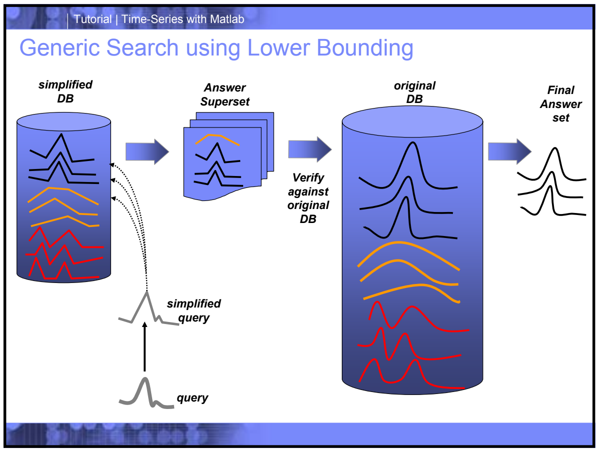
Tutorial | Time-Series with Matlab

Concept of Lower Bounding

- ***You can guarantee similar results to Linear Scan in the original dimensionality, as long as you provide a** Lower Bounding (LB) function* ***(in low dim) to the original distance (high dim.)***
  GEMINI, GEneric Multimedia INdexIng
  - So, for projection from high dim. (N) to low dim. (n): A→a, B→b etc

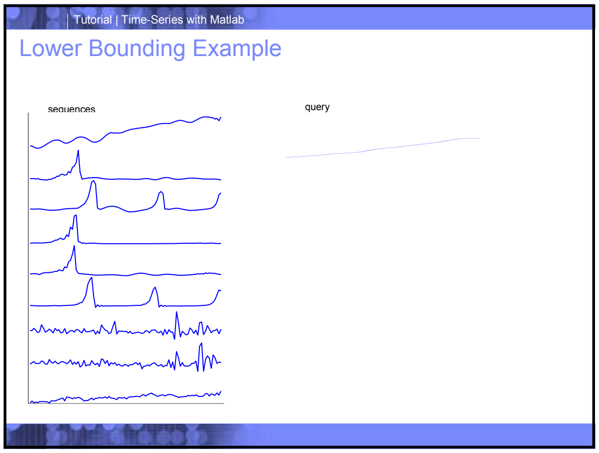
$D_{LB}(a,b) <= D_{true}(A,B)$

"Find everything within range of 1 from A"



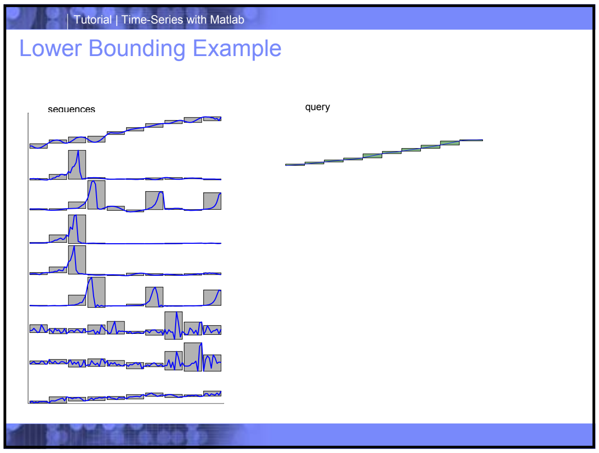
Tutorial | Time-Series with Matlab

Generic Search using Lower Bounding



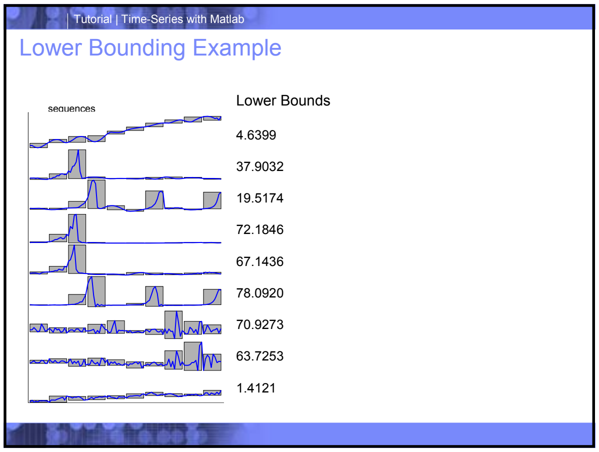
Tutorial | Time-Series with Matlab

Lower Bounding Example



Tutorial | Time-Series with Matlab

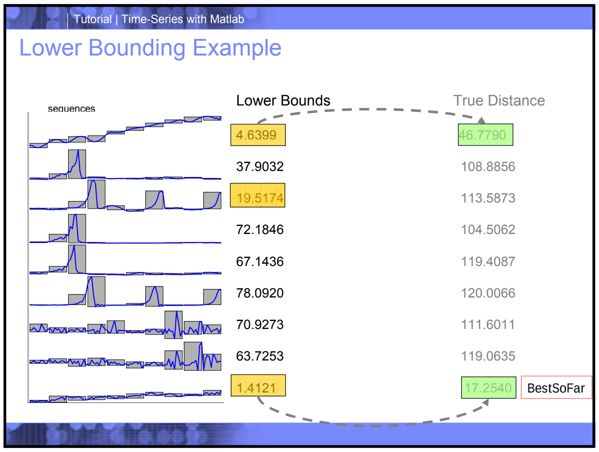
Lower Bounding Example



Tutorial | Time-Series with Matlab

Lower Bounding Example

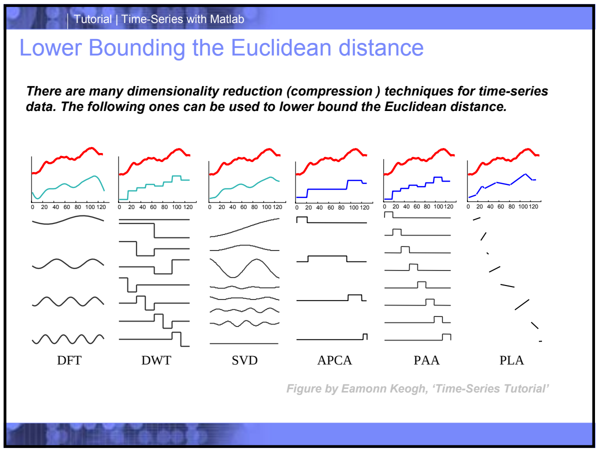
Lower Bounds
4.6399
37.9032
19.5174
72.1846
67.1436
78.0920
70.9273
63.7253
1.4121



Tutorial | Time-Series with Matlab

Lower Bounding Example

Lower Bounds	True Distance	
4.6399	46.7790	
37.9032	108.8856	
19.5174	113.5873	
72.1846	104.5062	
67.1436	119.4087	
78.0920	120.0066	
70.9273	111.6011	
63.7253	119.0635	
1.4121	17.2540	BestSoFar

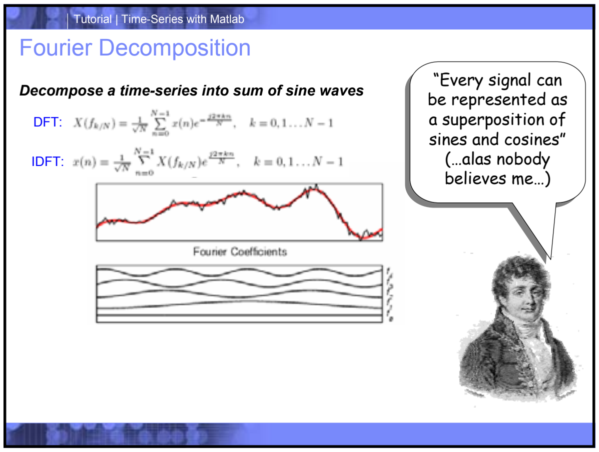


Tutorial | Time-Series with Matlab

Lower Bounding the Euclidean distance

There are many dimensionality reduction (compression) techniques for time-series data. The following ones can be used to lower bound the Euclidean distance.

Figure by Eamonn Keogh, 'Time-Series Tutorial'



Tutorial | Time-Series with Matlab

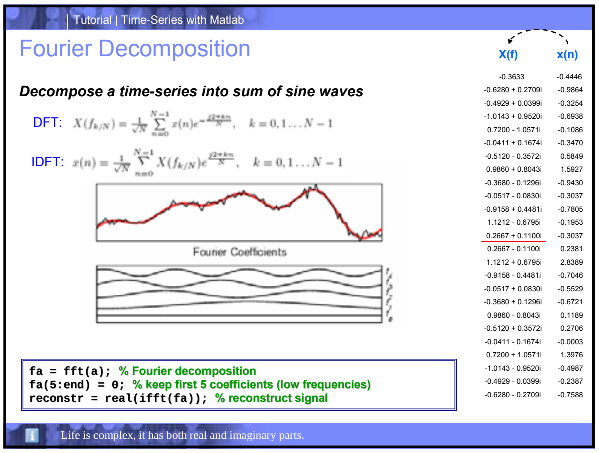
Fourier Decomposition

Decompose a time-series into sum of sine waves

DFT: $X(f_{k/N}) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) e^{-\frac{j2\pi kn}{N}}, \quad k = 0,1\ldots N-1$

IDFT: $x(n) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} X(f_{k/N}) e^{\frac{j2\pi kn}{N}}, \quad k = 0,1\ldots N-1$

Fourier Coefficients
"Every signal can be represented as a superposition of sines and cosines" (…alas nobody believes me…)



Tutorial | Time-Series with Matlab

Fourier Decomposition

Decompose a time-series into sum of sine waves

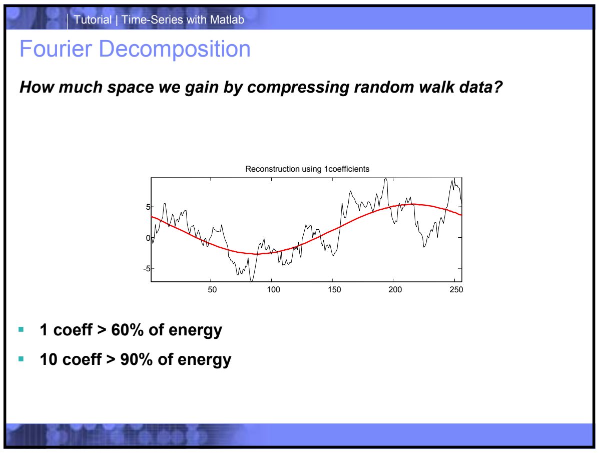
DFT: $X(f_{k/N}) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) e^{-\frac{j2\pi kn}{N}}, \quad k = 0,1\ldots N-1$

IDFT: $x(n) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} X(f_{k/N}) e^{\frac{j2\pi kn}{N}}, \quad k = 0,1\ldots N-1$

```
fa = fft(a);  % Fourier decomposition
fa(5:end) = 0;  % keep first 5 coefficients (low frequencies)
reconstr = real(ifft(fa));  % reconstruct signal
```

X(f)	x(n)
-0.3633	-0.4446
-0.6280 + 0.2709i	-0.9864
-0.4929 + 0.0399i	-0.3254
-1.0143 + 0.9520i	-0.6938
0.7200 - 1.0571i	-0.1086
-0.0411 + 0.1674i	-0.3470
-0.5120 - 0.3572i	0.5849
0.9860 + 0.8043i	1.5927
-0.3680 - 0.1296i	-0.9430
-0.0517 - 0.0830i	-0.3037
-0.9158 + 0.4481i	-0.7805
1.1212 - 0.6795i	-0.1953
0.2667 + 0.1100i	-0.3037
0.2667 - 0.1100i	0.2381
1.1212 + 0.6795i	2.8389
-0.9158 - 0.4481i	-0.7046
-0.0517 + 0.0830i	-0.5529
-0.3680 + 0.1296i	-0.6721
0.9860 - 0.8043i	0.1189
-0.5120 + 0.3572i	0.2706
-0.0411 - 0.1674i	-0.0003
0.7200 + 1.0571i	1.3976
-1.0143 - 0.9520i	-0.4987
-0.4929 - 0.0399i	-0.2387
-0.6280 - 0.2709i	-0.7588

Life is complex, it has both real and imaginary parts.

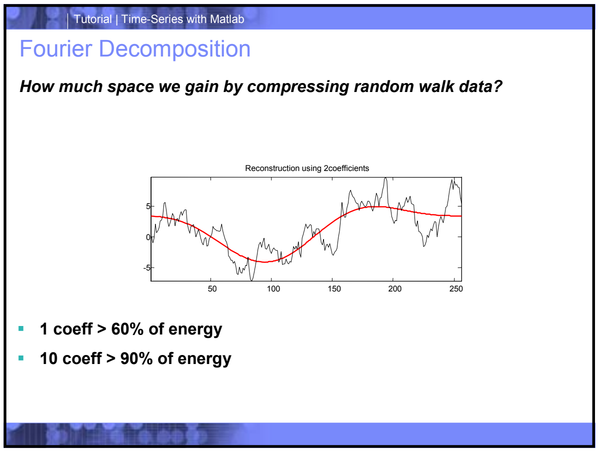


Tutorial | Time-Series with Matlab

Fourier Decomposition

How much space we gain by compressing random walk data?

- **1 coeff > 60% of energy**
- **10 coeff > 90% of energy**

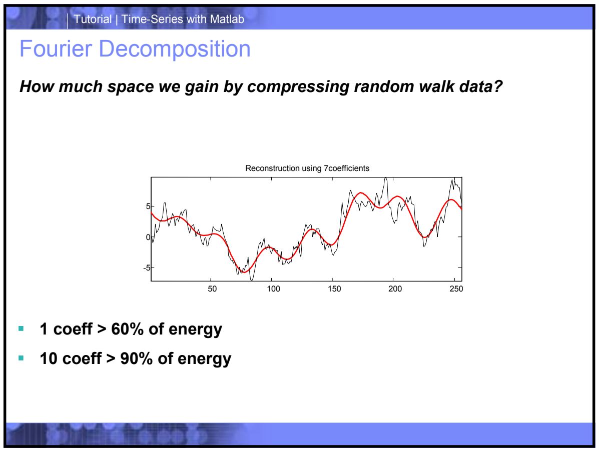


Tutorial | Time-Series with Matlab

Fourier Decomposition

How much space we gain by compressing random walk data?

- **1 coeff > 60% of energy**
- **10 coeff > 90% of energy**

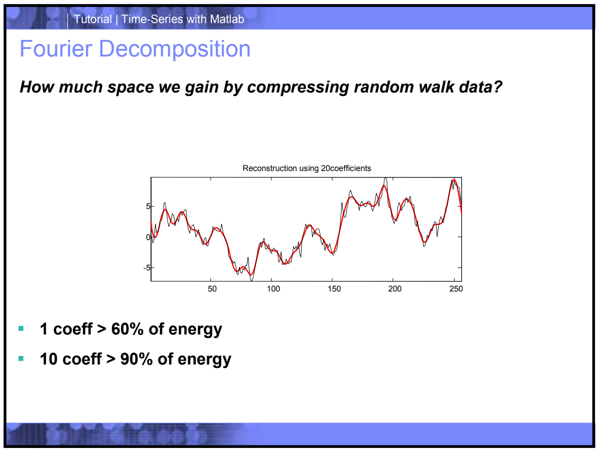


Tutorial | Time-Series with Matlab

Fourier Decomposition

How much space we gain by compressing random walk data?

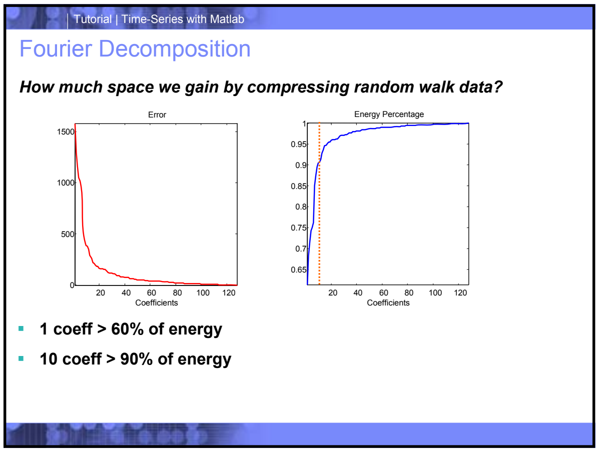
- **1 coeff > 60% of energy**
- **10 coeff > 90% of energy**



Fourier Decomposition

How much space we gain by compressing random walk data?

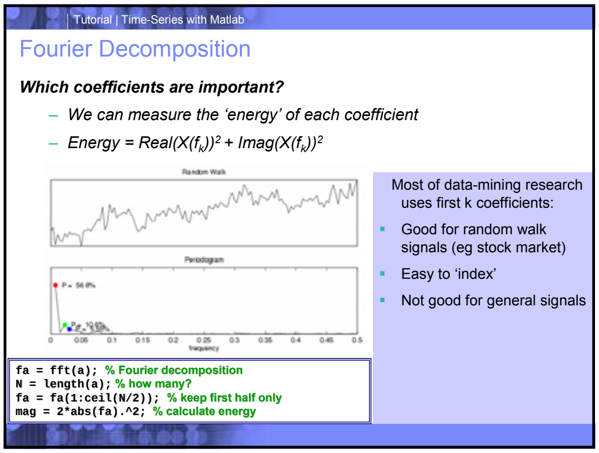
- **1 coeff > 60% of energy**
- **10 coeff > 90% of energy**



Fourier Decomposition

How much space we gain by compressing random walk data?

- **1 coeff > 60% of energy**
- **10 coeff > 90% of energy**



Fourier Decomposition

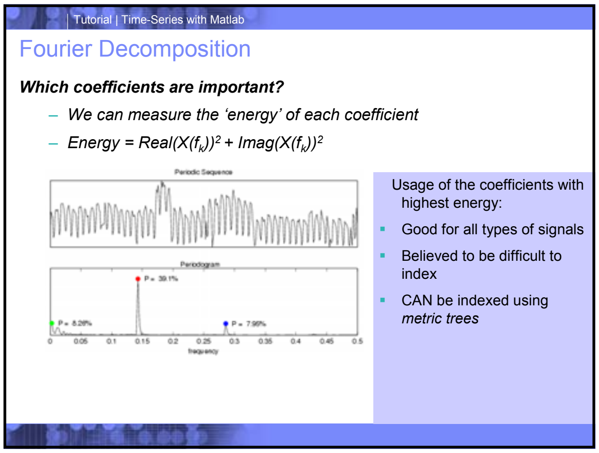
Which coefficients are important?

- *We can measure the 'energy' of each coefficient*
- *Energy = Real(X(f_k))^2 + Imag(X(f_k))^2*

```
fa = fft(a);  % Fourier decomposition
N = length(a); % how many?
fa = fa(1:ceil(N/2));  % keep first half only
mag = 2*abs(fa).^2;  % calculate energy
```

Most of data-mining research uses first k coefficients:

- Good for random walk signals (eg stock market)
- Easy to 'index'
- Not good for general signals



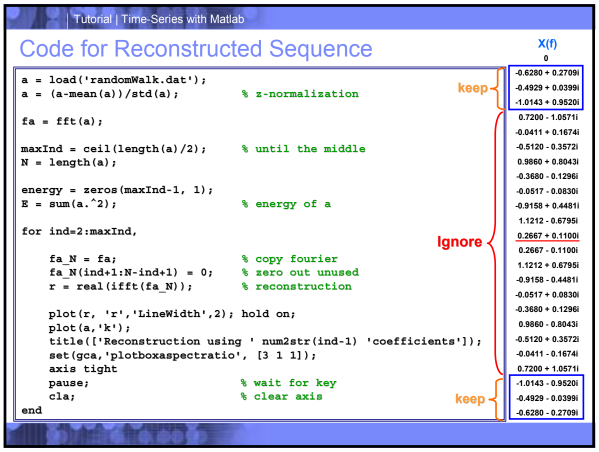
Fourier Decomposition

Which coefficients are important?

- *We can measure the 'energy' of each coefficient*
- *Energy = Real(X(f_k))^2 + Imag(X(f_k))^2*

Usage of the coefficients with highest energy:

- Good for all types of signals
- Believed to be difficult to index
- CAN be indexed using *metric trees*



Code for Reconstructed Sequence

```
a = load('randomWalk.dat');
a = (a-mean(a))/std(a);          % z-normalization

fa = fft(a);

maxInd = ceil(length(a)/2);      % until the middle
N = length(a);

energy = zeros(maxInd-1, 1);
E = sum(a.^2);                   % energy of a

for ind=2:maxInd,

    fa_N = fa;                   % copy fourier
    fa_N(ind+1:N-ind+1) = 0;     % zero out unused
    r = real(ifft(fa_N));        % reconstruction

    plot(r, 'r','LineWidth',2); hold on;
    plot(a,'k');
    title(['Reconstruction using ' num2str(ind-1) 'coefficients']);
    set(gca,'plotboxaspectratio', [3 1 1]);
    axis tight
    pause;                       % wait for key
    cla;                         % clear axis
end
```

X(f)

0

keep

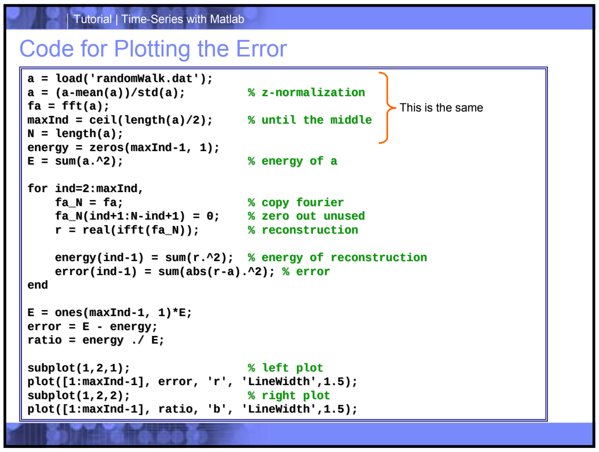
-0.6280 + 0.2709i
-0.4929 + 0.0399i
-1.0143 + 0.9520i

Ignore

0.7200 - 1.0571i
-0.0411 + 0.1674i
-0.5120 - 0.3572i
0.9860 + 0.8043i
-0.3680 - 0.1296i
-0.0517 - 0.0830i
-0.9158 + 0.4481i
1.1212 - 0.6795i
0.2667 + 0.1100i
0.2667 - 0.1100i
1.1212 + 0.6795i
-0.9158 - 0.4481i
-0.0517 + 0.0830i
-0.3680 + 0.1296i
0.9860 - 0.8043i
-0.5120 + 0.3572i
-0.0411 - 0.1674i
0.7200 + 1.0571i

keep

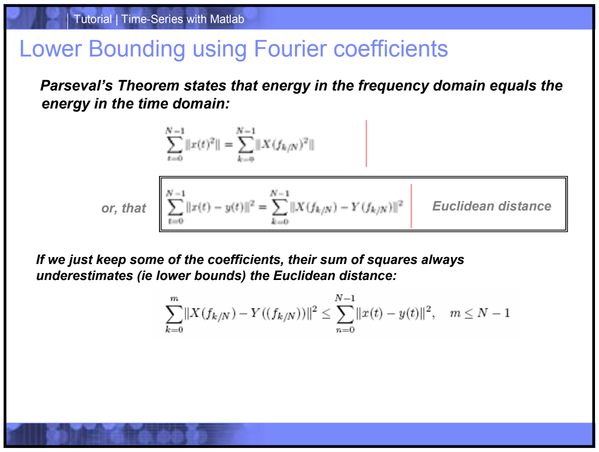
-1.0143 - 0.9520i
-0.4929 - 0.0399i
-0.6280 - 0.2709i



Code for Plotting the Error

```
a = load('randomWalk.dat');
a = (a-mean(a))/std(a);          % z-normalization
fa = fft(a);
maxInd = ceil(length(a)/2);      % until the middle
N = length(a);
energy = zeros(maxInd-1, 1);
E = sum(a.^2);                   % energy of a

for ind=2:maxInd,
    fa_N = fa;                   % copy fourier
    fa_N(ind+1:N-ind+1) = 0;     % zero out unused
    r = real(ifft(fa_N));        % reconstruction

    energy(ind-1) = sum(r.^2);  % energy of reconstruction
    error(ind-1) = sum(abs(r-a).^2); % error
end

E = ones(maxInd-1, 1)*E;
error = E - energy;
ratio = energy ./ E;

subplot(1,2,1);                  % left plot
plot([1:maxInd-1], error, 'r', 'LineWidth',1.5);
subplot(1,2,2);                  % right plot
plot([1:maxInd-1], ratio, 'b', 'LineWidth',1.5);
```

This is the same



Tutorial | Time-Series with Matlab

Lower Bounding using Fourier coefficients

Parseval's Theorem states that energy in the frequency domain equals the energy in the time domain:

$$\sum_{t=0}^{N-1}\|x(t)^2\| = \sum_{k=0}^{N-1}\|X(f_{k/N})^2\|$$

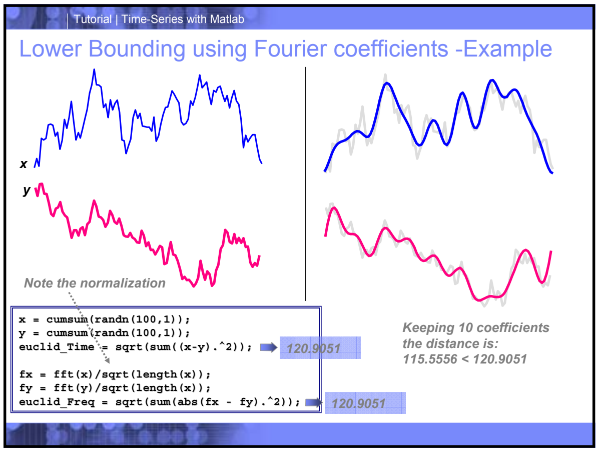
or, that

$$\sum_{t=0}^{N-1}\|x(t)-y(t)\|^2 = \sum_{k=0}^{N-1}\|X(f_{k/N})-Y(f_{k/N})\|^2$$

Euclidean distance

If we just keep some of the coefficients, their sum of squares always underestimates (ie lower bounds) the Euclidean distance:

$$\sum_{k=0}^{m}\|X(f_{k/N})-Y((f_{k/N}))\|^2 \le \sum_{n=0}^{N-1}\|x(t)-y(t)\|^2, \quad m \le N-1$$



Tutorial | Time-Series with Matlab

Lower Bounding using Fourier coefficients -Example

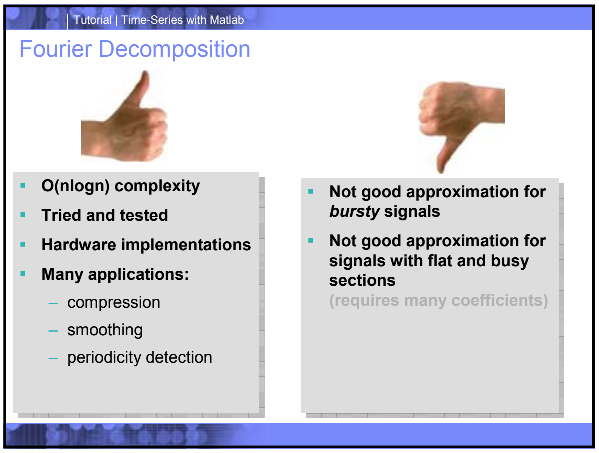
Note the normalization

```
x = cumsum(randn(100,1));
y = cumsum(randn(100,1));
euclid_Time = sqrt(sum((x-y).^2));

fx = fft(x)/sqrt(length(x));
fy = fft(y)/sqrt(length(x));
euclid_Freq = sqrt(sum(abs(fx - fy).^2));
```

120.9051

120.9051

Keeping 10 coefficients the distance is: 115.5556 < 120.9051

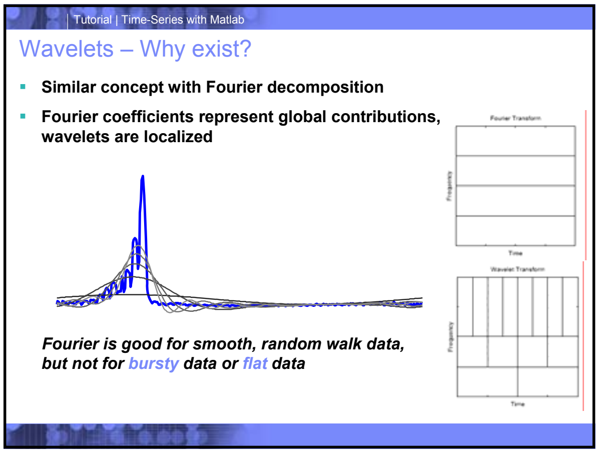


Tutorial | Time-Series with Matlab

Fourier Decomposition

- **O(nlogn) complexity**
- **Tried and tested**
- **Hardware implementations**
- **Many applications:**
  - compression
  - smoothing
  - periodicity detection

- **Not good approximation for *bursty* signals**
- **Not good approximation for signals with flat and busy sections**
  (requires many coefficients)

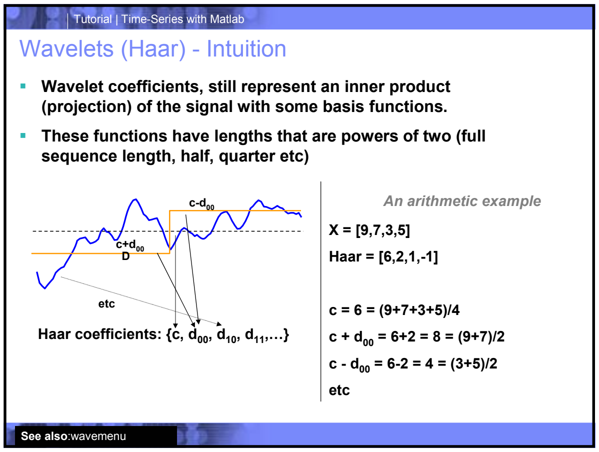


Tutorial | Time-Series with Matlab

Wavelets – Why exist?

- **Similar concept with Fourier decomposition**
- **Fourier coefficients represent global contributions, wavelets are localized**

Fourier is good for smooth, random walk data, but not for bursty data or flat data



Tutorial | Time-Series with Matlab

Wavelets (Haar) - Intuition

- **Wavelet coefficients, still represent an inner product (projection) of the signal with some basis functions.**
- **These functions have lengths that are powers of two (full sequence length, half, quarter etc)**

Haar coefficients: {c, d_{00}, d_{10}, d_{11},…}

An arithmetic example

X = [9,7,3,5]

Haar = [6,2,1,-1]

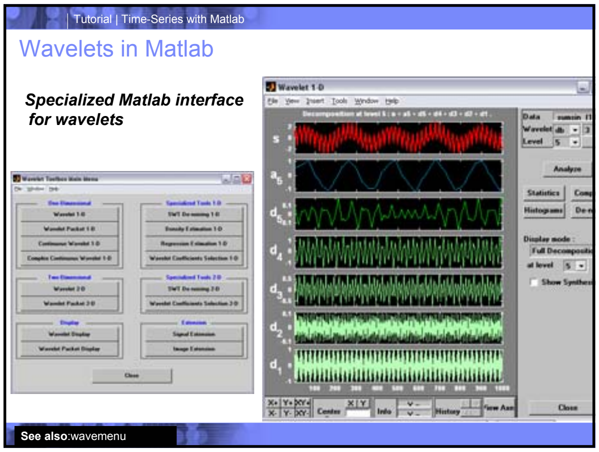
c = 6 = (9+7+3+5)/4

c + d_{00} = 6+2 = 8 = (9+7)/2

c - d_{00} = 6-2 = 4 = (3+5)/2

etc

See also:wavemenu

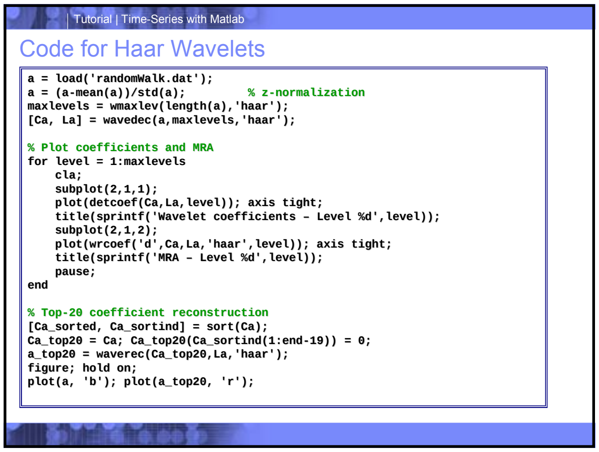


Tutorial | Time-Series with Matlab

Wavelets in Matlab

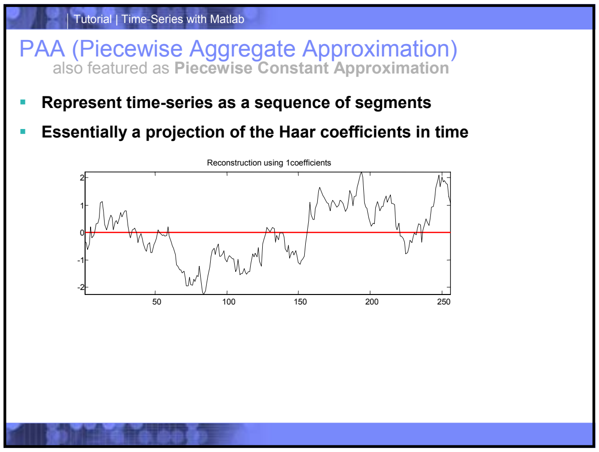
Specialized Matlab interface for wavelets

See also:wavemenu



Code for Haar Wavelets

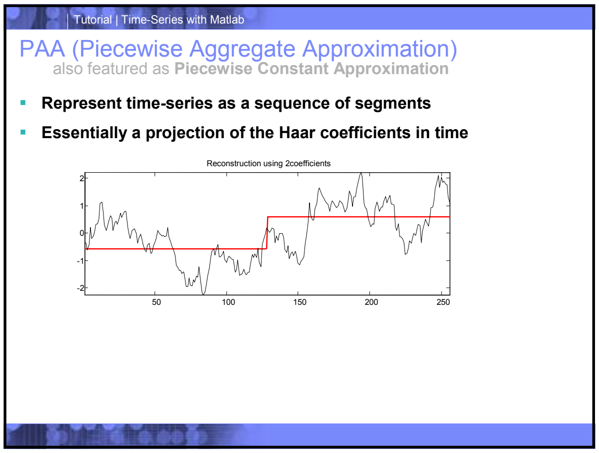
```
a = load('randomWalk.dat');
a = (a-mean(a))/std(a);         % z-normalization
maxlevels = wmaxlev(length(a),'haar');
[Ca, La] = wavedec(a,maxlevels,'haar');

% Plot coefficients and MRA
for level = 1:maxlevels
    cla;
    subplot(2,1,1);
    plot(detcoef(Ca,La,level)); axis tight;
    title(sprintf('Wavelet coefficients – Level %d',level));
    subplot(2,1,2);
    plot(wrcoef('d',Ca,La,'haar',level)); axis tight;
    title(sprintf('MRA – Level %d',level));
    pause;
end

% Top-20 coefficient reconstruction
[Ca_sorted, Ca_sortind] = sort(Ca);
Ca_top20 = Ca; Ca_top20(Ca_sortind(1:end-19)) = 0;
a_top20 = waverec(Ca_top20,La,'haar');
figure; hold on;
plot(a, 'b'); plot(a_top20, 'r');
```



PAA (Piecewise Aggregate Approximation)

also featured as **Piecewise Constant Approximation**

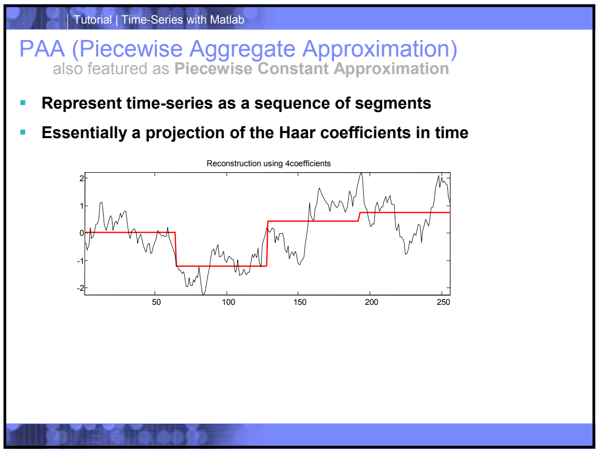
- **Represent time-series as a sequence of segments**
- **Essentially a projection of the Haar coefficients in time**



PAA (Piecewise Aggregate Approximation)

also featured as **Piecewise Constant Approximation**

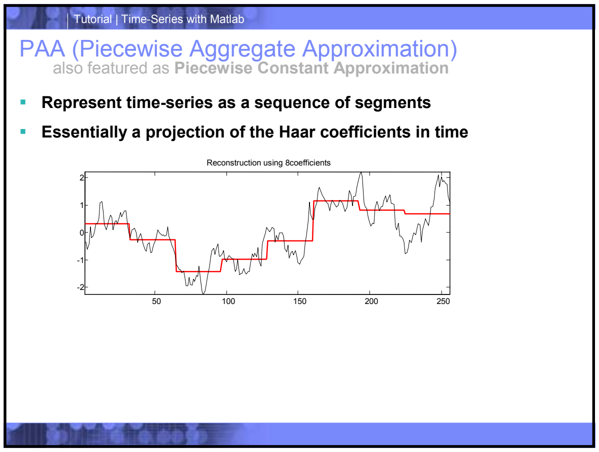
- **Represent time-series as a sequence of segments**
- **Essentially a projection of the Haar coefficients in time**



PAA (Piecewise Aggregate Approximation)

also featured as **Piecewise Constant Approximation**

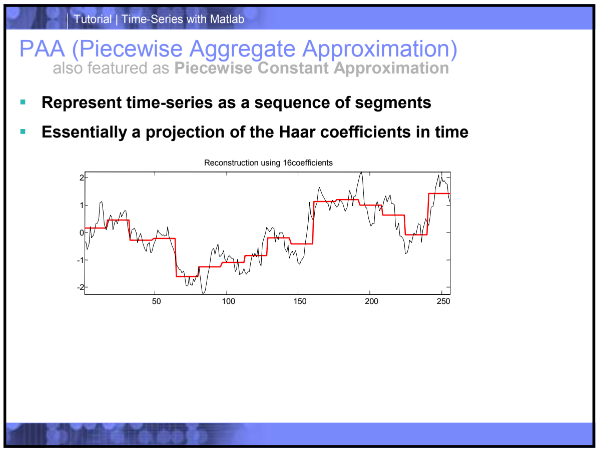
- **Represent time-series as a sequence of segments**
- **Essentially a projection of the Haar coefficients in time**



PAA (Piecewise Aggregate Approximation)

also featured as **Piecewise Constant Approximation**

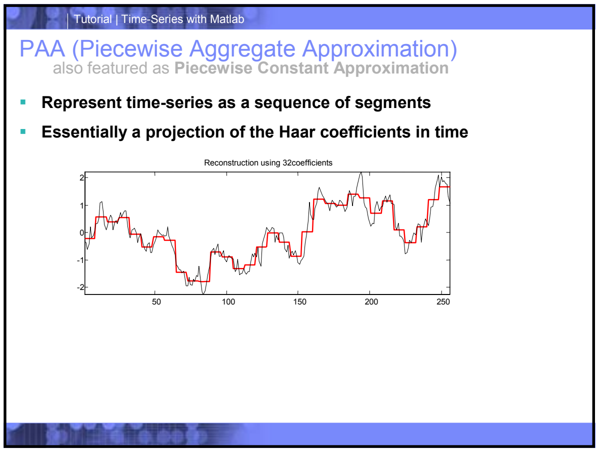
- **Represent time-series as a sequence of segments**
- **Essentially a projection of the Haar coefficients in time**



PAA (Piecewise Aggregate Approximation)

also featured as **Piecewise Constant Approximation**

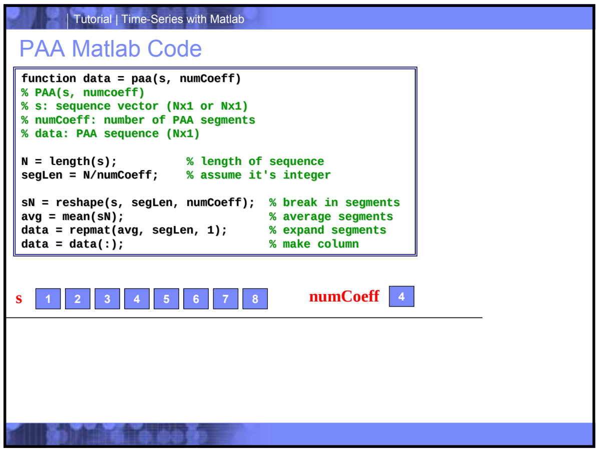
- **Represent time-series as a sequence of segments**
- **Essentially a projection of the Haar coefficients in time**


Tutorial | Time-Series with Matlab

PAA (Piecewise Aggregate Approximation)

also featured as **Piecewise Constant Approximation**

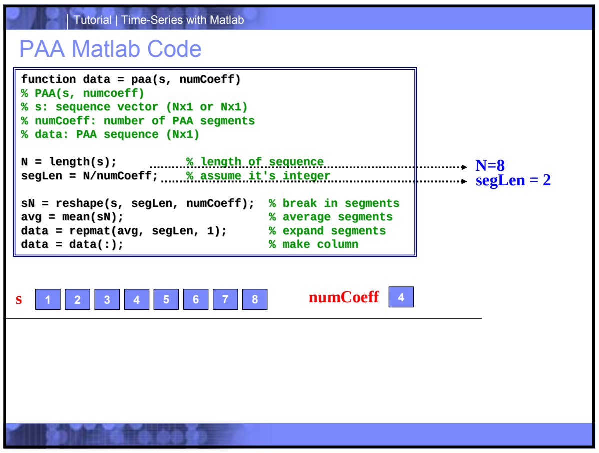
- **Represent time-series as a sequence of segments**
- **Essentially a projection of the Haar coefficients in time**


Tutorial | Time-Series with Matlab

PAA Matlab Code

```
function data = paa(s, numCoeff)
% PAA(s, numcoeff)
% s: sequence vector (Nx1 or Nx1)
% numCoeff: number of PAA segments
% data: PAA sequence (Nx1)

N = length(s);          % length of sequence
segLen = N/numCoeff;    % assume it's integer

sN = reshape(s, segLen, numCoeff);  % break in segments
avg = mean(sN);                     % average segments
data = repmat(avg, segLen, 1);      % expand segments
data = data(:);                     % make column
```

s 1 2 3 4 5 6 7 8 numCoeff 4

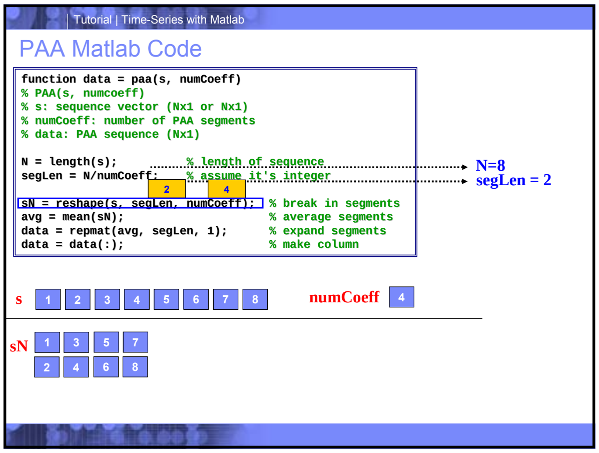

Tutorial | Time-Series with Matlab

PAA Matlab Code

```
function data = paa(s, numCoeff)
% PAA(s, numcoeff)
% s: sequence vector (Nx1 or Nx1)
% numCoeff: number of PAA segments
% data: PAA sequence (Nx1)

N = length(s);          % length of sequence
segLen = N/numCoeff;    % assume it's integer

sN = reshape(s, segLen, numCoeff);  % break in segments
avg = mean(sN);                     % average segments
data = repmat(avg, segLen, 1);      % expand segments
data = data(:);                     % make column
```

N=8
segLen = 2

s 1 2 3 4 5 6 7 8 numCoeff 4


Tutorial | Time-Series with Matlab

PAA Matlab Code

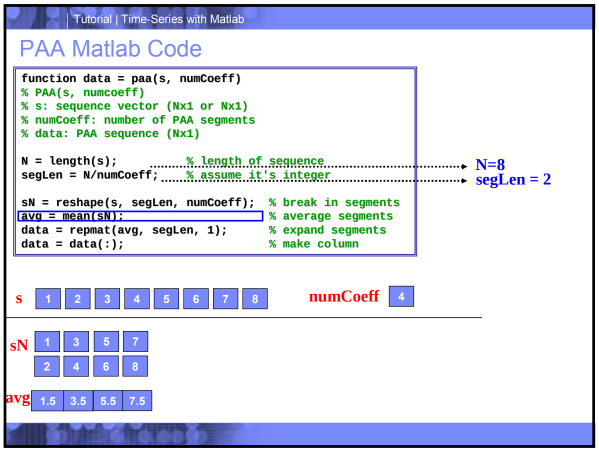
```
function data = paa(s, numCoeff)
% PAA(s, numcoeff)
% s: sequence vector (Nx1 or Nx1)
% numCoeff: number of PAA segments
% data: PAA sequence (Nx1)

N = length(s);          % length of sequence
segLen = N/numCoeff;    % assume it's integer

sN = reshape(s, segLen, numCoeff);  % break in segments
avg = mean(sN);                     % average segments
data = repmat(avg, segLen, 1);      % expand segments
data = data(:);                     % make column
```

N=8
segLen = 2

2 4

s 1 2 3 4 5 6 7 8 numCoeff 4

sN 1 3 5 7
2 4 6 8


Tutorial | Time-Series with Matlab

PAA Matlab Code

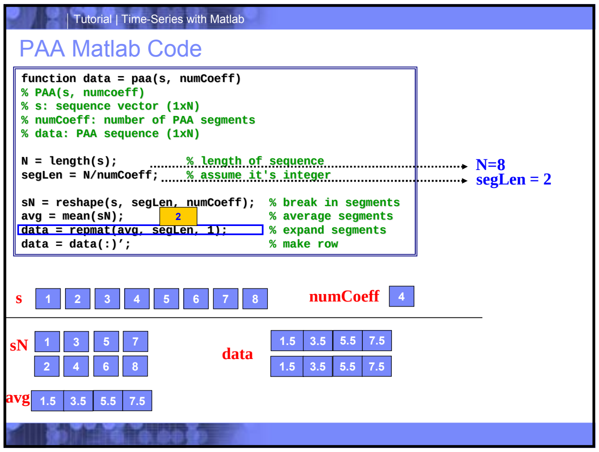
```
function data = paa(s, numCoeff)
% PAA(s, numcoeff)
% s: sequence vector (Nx1 or Nx1)
% numCoeff: number of PAA segments
% data: PAA sequence (Nx1)

N = length(s);          % length of sequence
segLen = N/numCoeff;    % assume it's integer

sN = reshape(s, segLen, numCoeff);  % break in segments
avg = mean(sN);                     % average segments
data = repmat(avg, segLen, 1);      % expand segments
data = data(:);                     % make column
```

N=8
segLen = 2

s 1 2 3 4 5 6 7 8 numCoeff 4

sN 1 3 5 7
2 4 6 8

avg 1.5 3.5 5.5 7.5


Tutorial | Time-Series with Matlab

PAA Matlab Code

```
function data = paa(s, numCoeff)
% PAA(s, numcoeff)
% s: sequence vector (1xN)
% numCoeff: number of PAA segments
% data: PAA sequence (1xN)

N = length(s);          % length of sequence
segLen = N/numCoeff;    % assume it's integer

sN = reshape(s, segLen, numCoeff);  % break in segments
avg = mean(sN);                     % average segments
data = repmat(avg, segLen, 1);      % expand segments
data = data(:)';                    % make row
```

N=8
segLen = 2

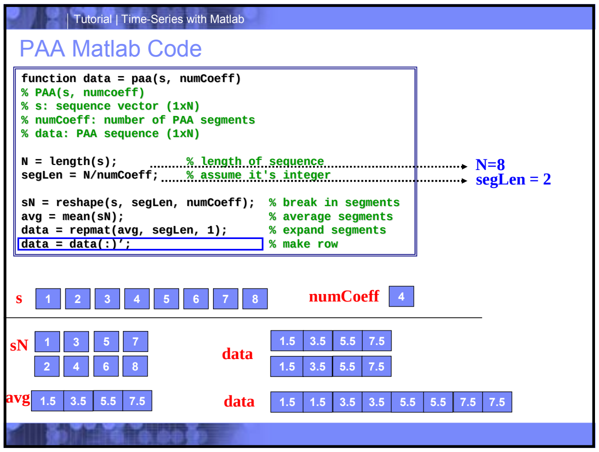
2

s 1 2 3 4 5 6 7 8 numCoeff 4

sN 1 3 5 7
2 4 6 8

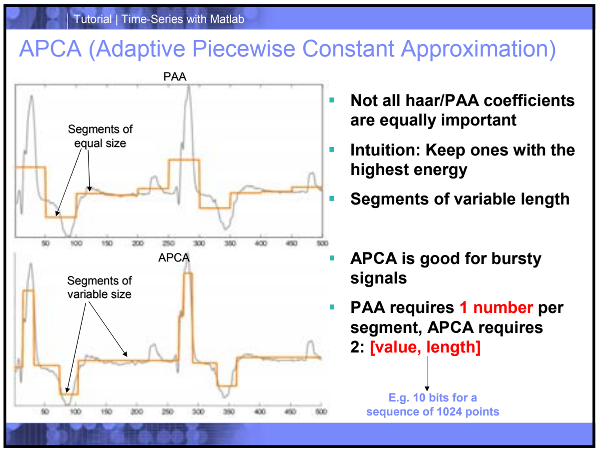
data 1.5 3.5 5.5 7.5
1.5 3.5 5.5 7.5

avg 1.5 3.5 5.5 7.5



PAA Matlab Code

```
function data = paa(s, numCoeff)
% PAA(s, numcoeff)
% s: sequence vector (1xN)
% numCoeff: number of PAA segments
% data: PAA sequence (1xN)

N = length(s);          % length of sequence
segLen = N/numCoeff;    % assume it's integer

sN = reshape(s, segLen, numCoeff);  % break in segments
avg = mean(sN);                     % average segments
data = repmat(avg, segLen, 1);      % expand segments
data = data(:)';                    % make row
```

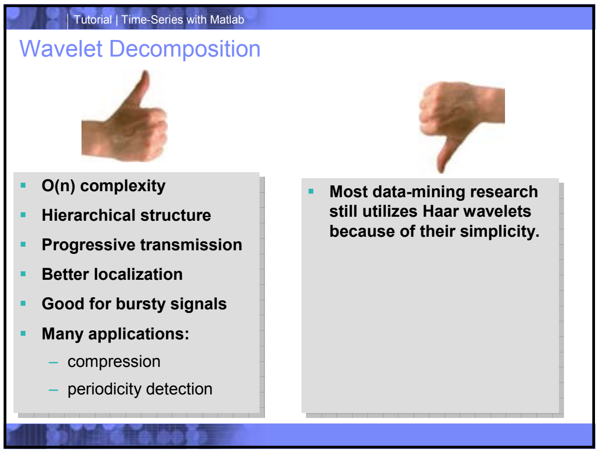


APCA (Adaptive Piecewise Constant Approximation)

- **Not all haar/PAA coefficients are equally important**
- **Intuition: Keep ones with the highest energy**
- **Segments of variable length**

- **APCA is good for bursty signals**
- **PAA requires 1 number per segment, APCA requires 2: [value, length]**

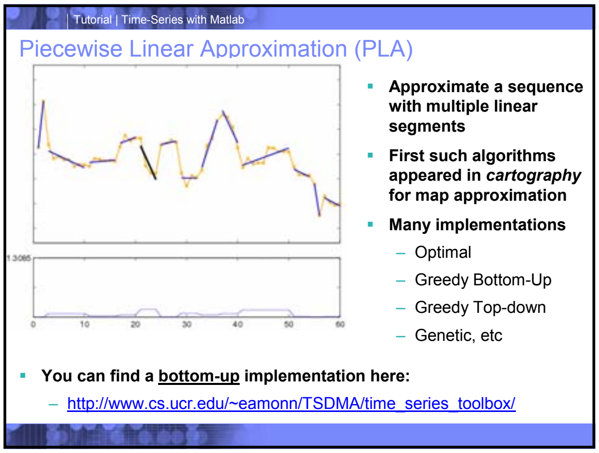
E.g. 10 bits for a sequence of 1024 points



Wavelet Decomposition

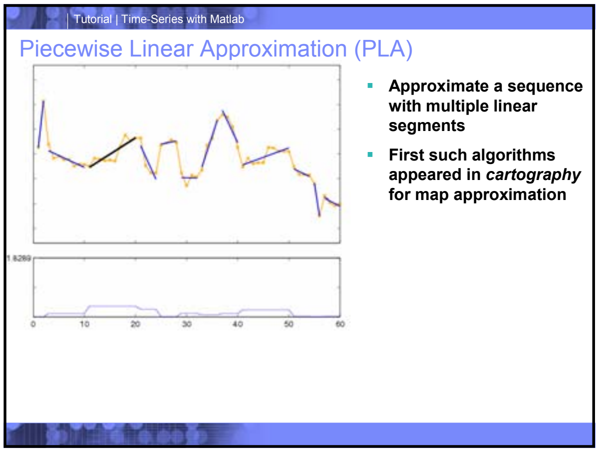
- **O(n) complexity**
- **Hierarchical structure**
- **Progressive transmission**
- **Better localization**
- **Good for bursty signals**
- **Many applications:**
  - compression
  - periodicity detection

- **Most data-mining research still utilizes Haar wavelets because of their simplicity.**



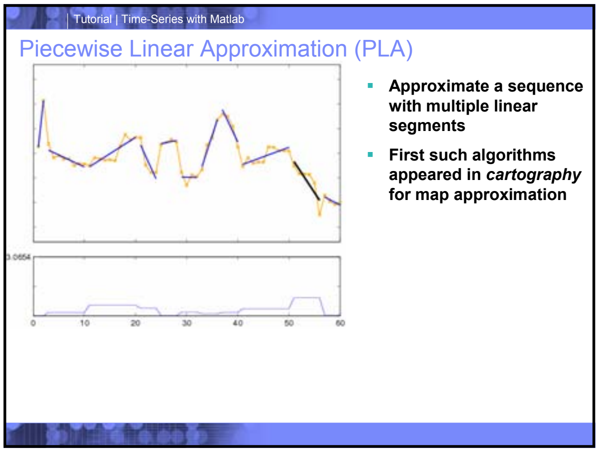
Piecewise Linear Approximation (PLA)

- **Approximate a sequence with multiple linear segments**
- **First such algorithms appeared in *cartography* for map approximation**
- **Many implementations**
  - Optimal
  - Greedy Bottom-Up
  - Greedy Top-down
  - Genetic, etc
- **You can find a bottom-up implementation here:**
  - http://www.cs.ucr.edu/~eamonn/TSDMA/time_series_toolbox/



Piecewise Linear Approximation (PLA)

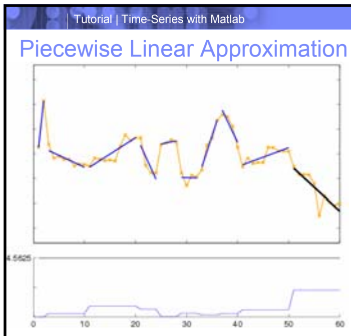
- **Approximate a sequence with multiple linear segments**
- **First such algorithms appeared in *cartography* for map approximation**



Piecewise Linear Approximation (PLA)

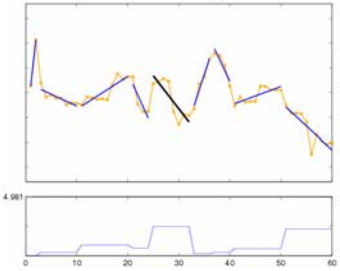
- **Approximate a sequence with multiple linear segments**
- **First such algorithms appeared in *cartography* for map approximation**

Piecewise Linear Approximation (PLA)



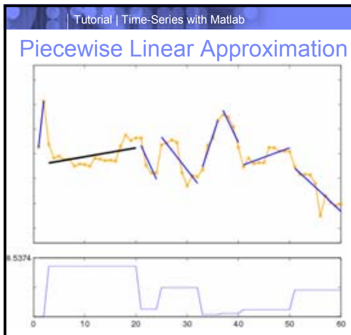
- Approximate a sequence with multiple linear segments
- First such algorithms appeared in *cartography* for map approximation

Piecewise Linear Approximation (PLA)



- Approximate a sequence with multiple linear segments
- First such algorithms appeared in *cartography* for map approximation

Piecewise Linear Approximation (PLA)



- Approximate a sequence with multiple linear segments
- First such algorithms appeared in *cartography* for map approximation

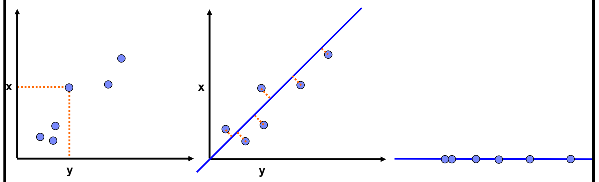
Piecewise Linear Approximation (PLA)

- O(nlogn) complexity for “bottom up” algorithm
- Incremental computation possible
- Provable error bounds
- Applications for:
  - Image / signal simplification
  - Trend detection

- Visually not very smooth or pleasing.

Singular Value Decomposition (SVD)

- SVD attempts to find the ‘optimal’ basis for describing a set of multidimensional points
- Objective: Find the axis (‘directions’) that describe better the data variance



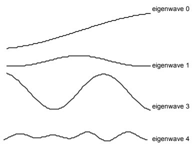
We need 2 numbers (x,y) for every point

Now we can describe each point with 1 number, their projection on the line

New axis and position of points (after projection and rotation)

Singular Value Decomposition (SVD)

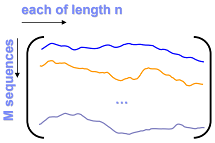
- Each time-series is essentially a multidimensional point
- Objective: Find the ‘eigenwaves’ (basis) whose linear combination describes best the sequences. Eigenwaves are data-dependent.



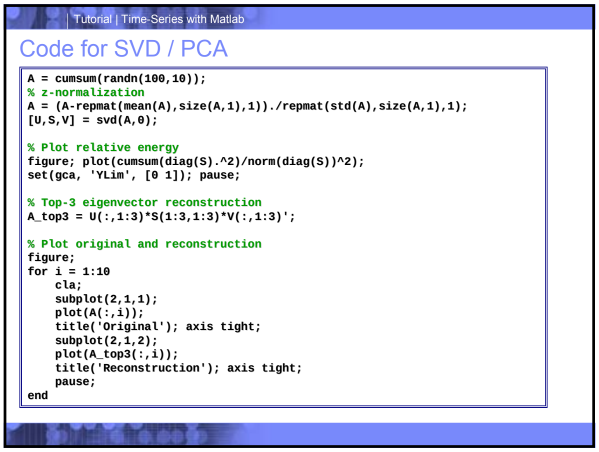
A linear combination of the eigenwaves can produce any sequence in the database

$$A_{Mxn} = U_{Mxr} * \Sigma_{rxr} * V^T_{nxr}$$

Factoring of data array into 3 matrices

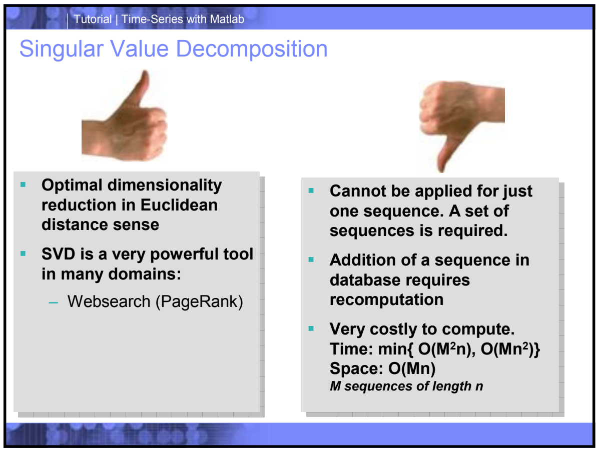


```
[U,S,V] = svd(A)
```



Code for SVD / PCA

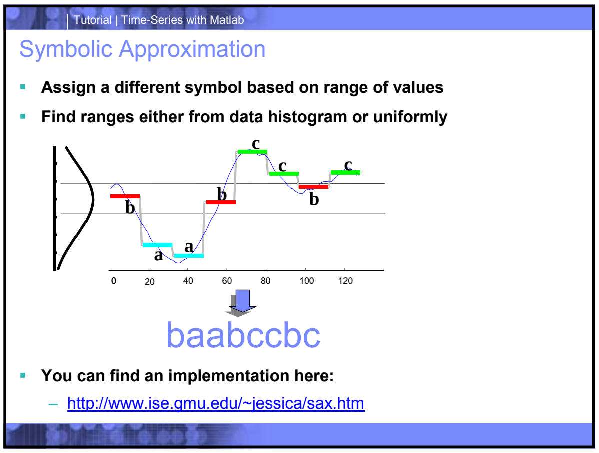
```
A = cumsum(randn(100,10));
% z-normalization
A = (A-repmat(mean(A),size(A,1),1))./repmat(std(A),size(A,1),1);
[U,S,V] = svd(A,0);

% Plot relative energy
figure; plot(cumsum(diag(S).^2)/norm(diag(S))^2);
set(gca, 'YLim', [0 1]); pause;

% Top-3 eigenvector reconstruction
A_top3 = U(:,1:3)*S(1:3,1:3)*V(:,1:3)';

% Plot original and reconstruction
figure;
for i = 1:10
    cla;
    subplot(2,1,1);
    plot(A(:,i));
    title('Original'); axis tight;
    subplot(2,1,2);
    plot(A_top3(:,i));
    title('Reconstruction'); axis tight;
    pause;
end
```



Singular Value Decomposition

- **Optimal dimensionality reduction in Euclidean distance sense**
- **SVD is a very powerful tool in many domains:**
  - Websearch (PageRank)

- **Cannot be applied for just one sequence. A set of sequences is required.**
- **Addition of a sequence in database requires recomputation**
- **Very costly to compute. Time: min{ $O(M^2n)$, $O(Mn^2)$} Space: $O(Mn)$**
  M sequences of length n

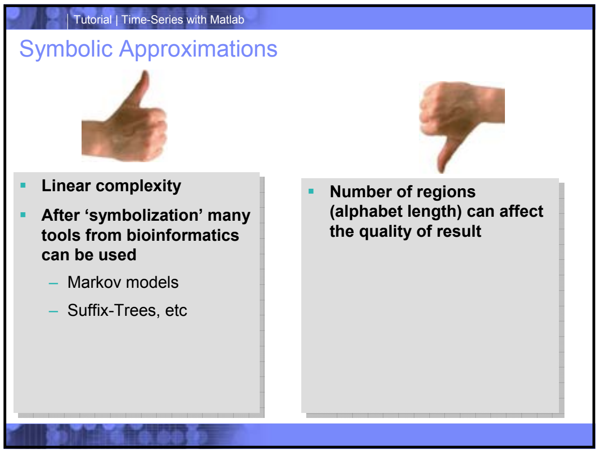


Symbolic Approximation

- **Assign a different symbol based on range of values**
- **Find ranges either from data histogram or uniformly**

baabccbc

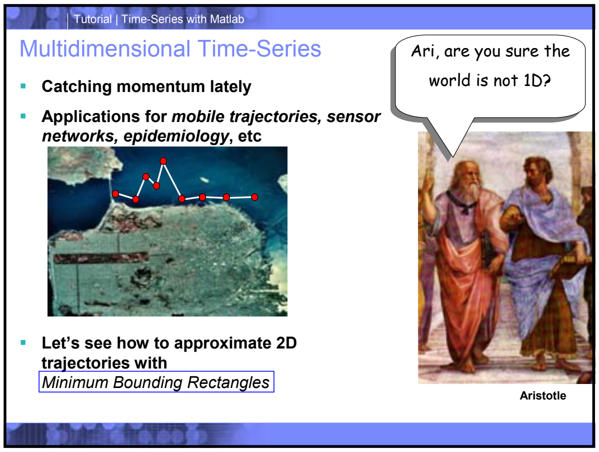
- **You can find an implementation here:**
  - http://www.ise.gmu.edu/~jessica/sax.htm



Symbolic Approximations

- **Linear complexity**
- **After 'symbolization' many tools from bioinformatics can be used**
  - Markov models
  - Suffix-Trees, etc

- **Number of regions (alphabet length) can affect the quality of result**



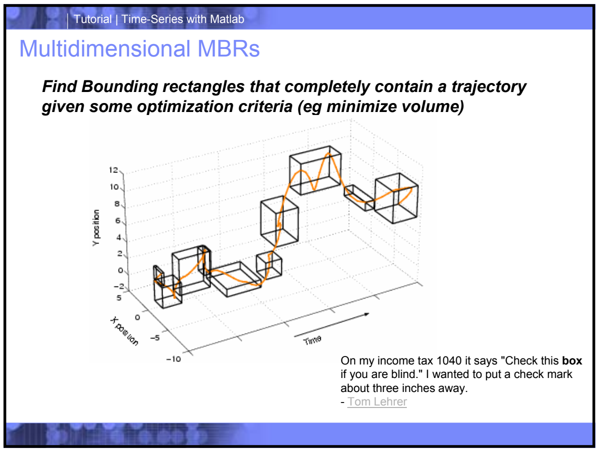
Multidimensional Time-Series

- **Catching momentum lately**
- **Applications for *mobile trajectories, sensor networks, epidemiology*, etc**

Ari, are you sure the world is not 1D?

- **Let's see how to approximate 2D trajectories with**
  Minimum Bounding Rectangles

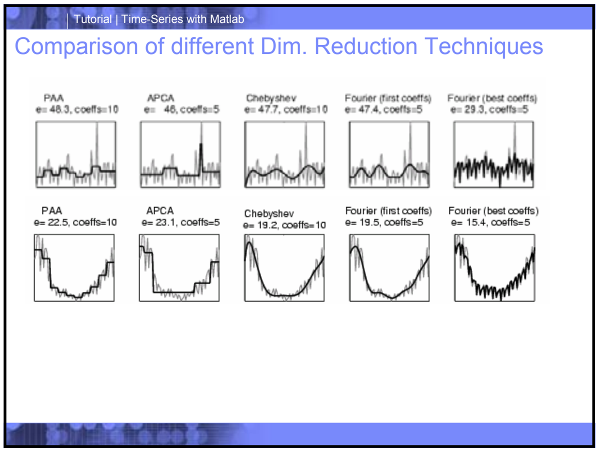
Aristotle



Multidimensional MBRs

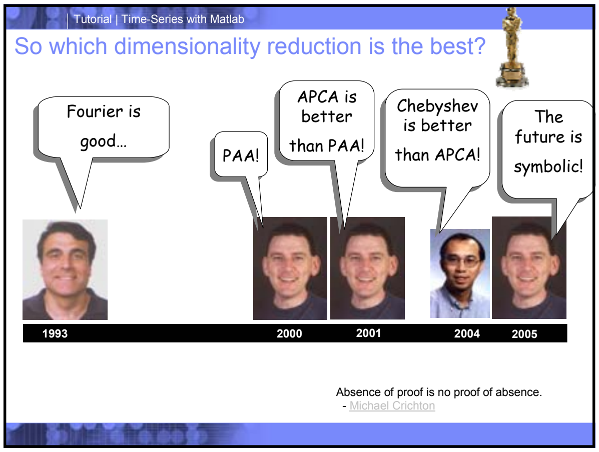
Find Bounding rectangles that completely contain a trajectory given some optimization criteria (eg minimize volume)

On my income tax 1040 it says "Check this **box** if you are blind." I wanted to put a check mark about three inches away.
- Tom Lehrer



Tutorial | Time-Series with Matlab

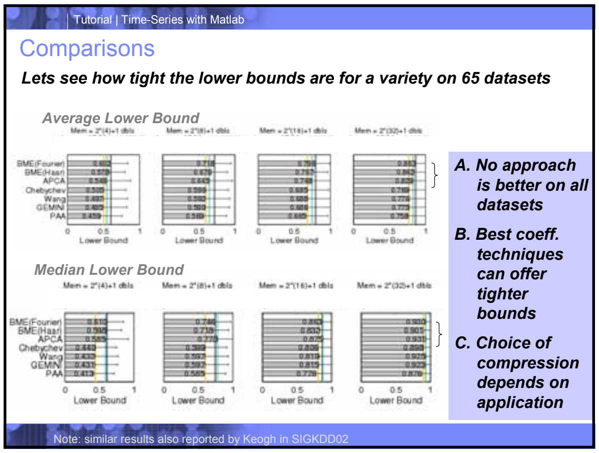
Comparison of different Dim. Reduction Techniques



Tutorial | Time-Series with Matlab

So which dimensionality reduction is the best?

Absence of proof is no proof of absence.
- Michael Crichton



Tutorial | Time-Series with Matlab

Comparisons

Lets see how tight the lower bounds are for a variety on 65 datasets

A. No approach is better on all datasets

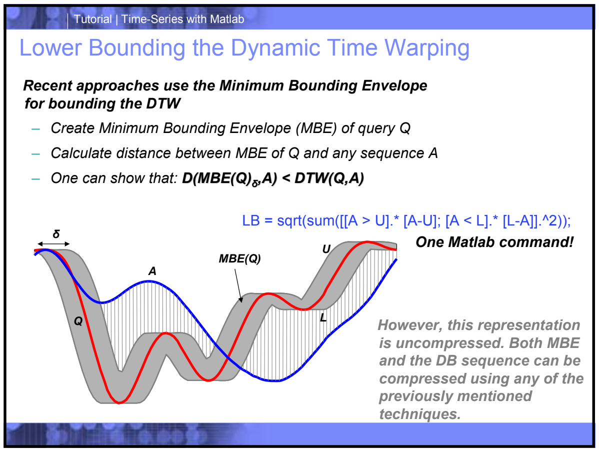
B. Best coeff. techniques can offer tighter bounds

C. Choice of compression depends on application

Note: similar results also reported by Keogh in SIGKDD02

Tutorial | Time-Series with Matlab

•PART II: Time Series Matching
Lower Bounding the DTW and LCSS



Tutorial | Time-Series with Matlab

Lower Bounding the Dynamic Time Warping

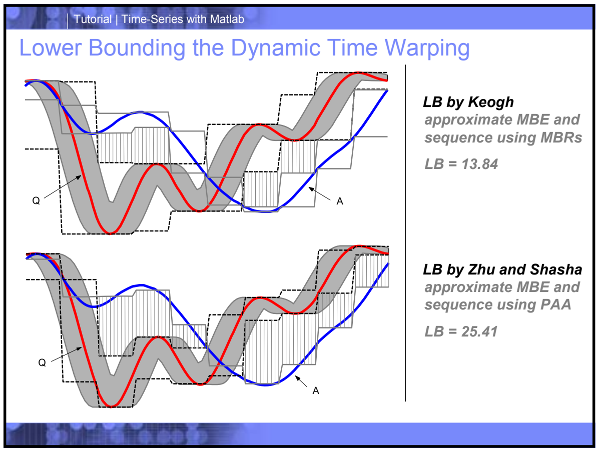
Recent approaches use the Minimum Bounding Envelope for bounding the DTW

- *Create Minimum Bounding Envelope (MBE) of query Q*
- *Calculate distance between MBE of Q and any sequence A*
- *One can show that:* $D(MBE(Q)_\delta,A) < DTW(Q,A)$

LB = sqrt(sum([[A > U].* [A-U]; [A < L].* [L-A]].^2));

One Matlab command!

However, this representation is uncompressed. Both MBE and the DB sequence can be compressed using any of the previously mentioned techniques.



Tutorial | Time-Series with Matlab

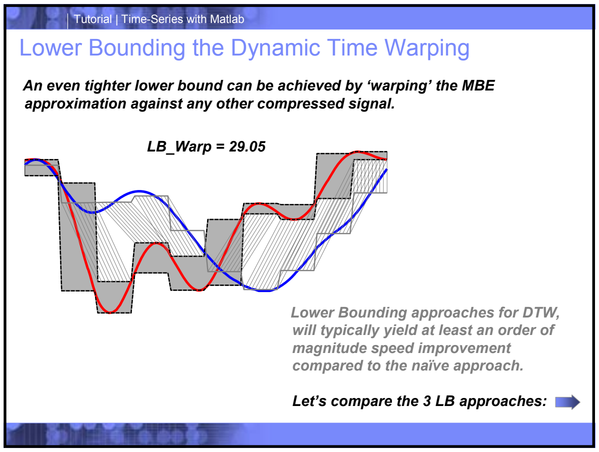
Lower Bounding the Dynamic Time Warping

LB by Keogh
approximate MBE and sequence using MBRs

LB = 13.84

LB by Zhu and Shasha
approximate MBE and sequence using PAA

LB = 25.41

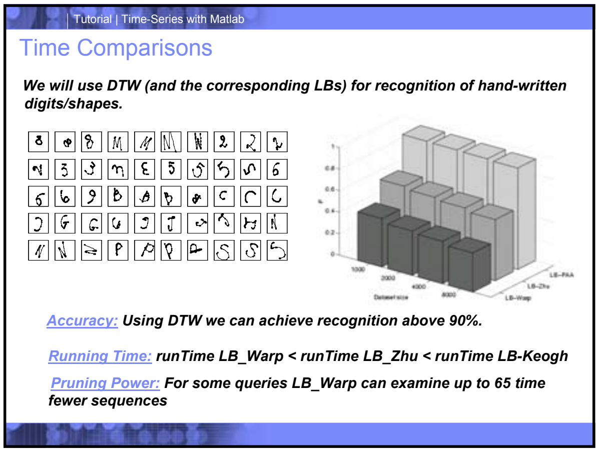


Lower Bounding the Dynamic Time Warping

An even tighter lower bound can be achieved by 'warping' the MBE approximation against any other compressed signal.

Lower Bounding approaches for DTW, will typically yield at least an order of magnitude speed improvement compared to the naïve approach.

Let's compare the 3 LB approaches:



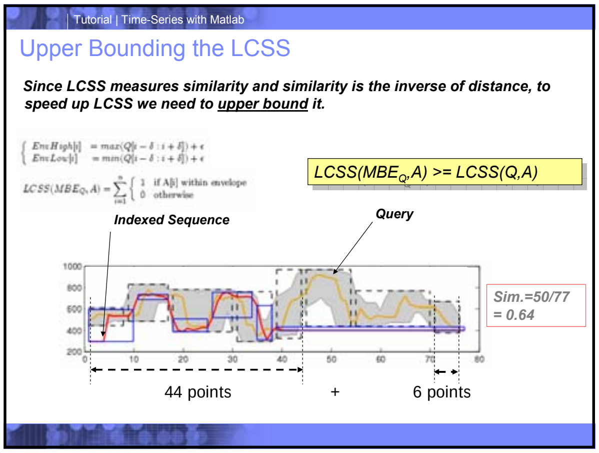
Time Comparisons

We will use DTW (and the corresponding LBs) for recognition of hand-written digits/shapes.

***Accuracy:** Using DTW we can achieve recognition above 90%.*

***Running Time:** runTime LB_Warp < runTime LB_Zhu < runTime LB-Keogh*

***Pruning Power:** For some queries LB_Warp can examine up to 65 time fewer sequences*



Upper Bounding the LCSS

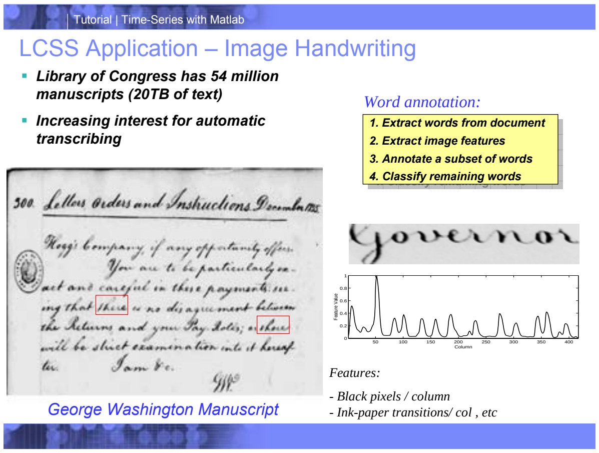
Since LCSS measures similarity and similarity is the inverse of distance, to speed up LCSS we need to <u>upper bound</u> it.

$$\begin{cases} EnvHigh[i] & = max(Q[i-\delta : i+\delta]) + \epsilon \\ EnvLow[i] & = min(Q[i-\delta : i+\delta]) + \epsilon \end{cases}$$

$$LCSS(MBE_Q, A) = \sum_{i=1}^{n} \begin{cases} 1 & \text{if A[i] within envelope} \\ 0 & \text{otherwise} \end{cases}$$

LCSS(MBE_Q,A) >= LCSS(Q,A)

Sim.=50/77 = 0.64



LCSS Application – Image Handwriting

- ***Library of Congress has 54 million manuscripts (20TB of text)***
- ***Increasing interest for automatic transcribing***

Word annotation:

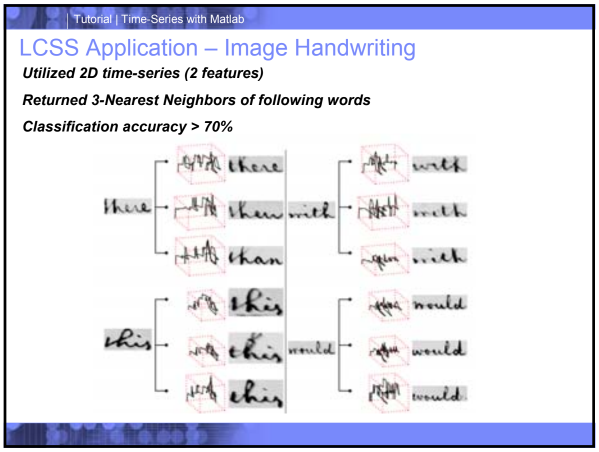
1. *Extract words from document*
2. *Extract image features*
3. *Annotate a subset of words*
4. *Classify remaining words*

George Washington Manuscript

Features:

- Black pixels / column

- Ink-paper transitions/ col , etc

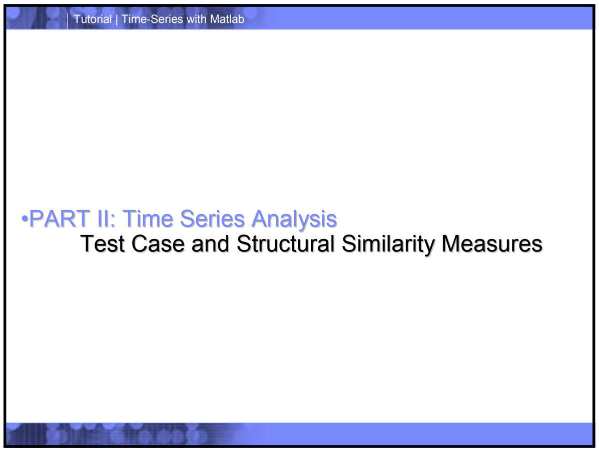


LCSS Application – Image Handwriting

Utilized 2D time-series (2 features)

Returned 3-Nearest Neighbors of following words

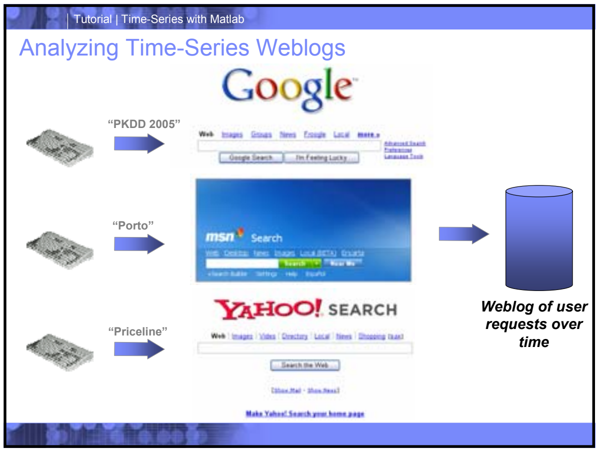
Classification accuracy > 70%



•PART II: Time Series Analysis

Test Case and Structural Similarity Measures

Analyzing Time-Series Weblogs

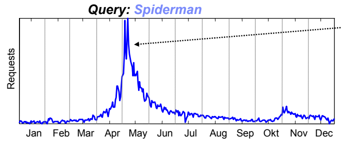


Weblog of user requests over time

Weblog Data Representation

Record aggregate information, eg, number of requests per day for each keyword

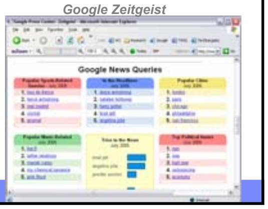
Query: Spiderman



May 2002. Spiderman 1 was released in theaters

- Capture trends and periodicities
- Privacy preserving

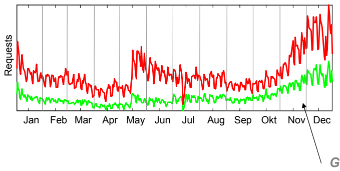
Google Zeitgeist



Finding similar patterns in query logs

We can find useful patterns and correlation in the user demand patterns which can be useful for:

- Search engine optimization
- Recommendations
- Advertisement pricing (e.g. keyword more expensive at the popular months)



Query: xbox

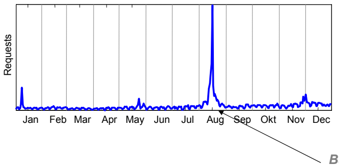
Query: ps2

Game consoles are more popular closer to Christmas

Finding similar patterns in query logs

We can find useful patterns and correlation in the user demand patterns which can be useful for:

- Search engine optimization
- Recommendations
- Advertisement pricing (e.g. keyword more expensive at the popular months)



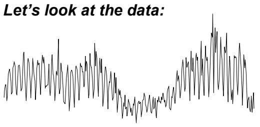
Query: elvis

Burst on Aug. 16th Death Anniversary of Elvis

Matching of Weblog data

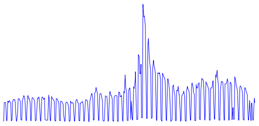
Use Euclidean distance to match time-series. But which dimensionality reduction technique to use?

Let's look at the data:



Query "Bach"

1 year span



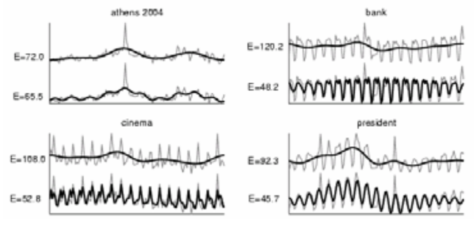
Query "stock market"

The data is smooth and highly periodic, so we can use Fourier decomposition.

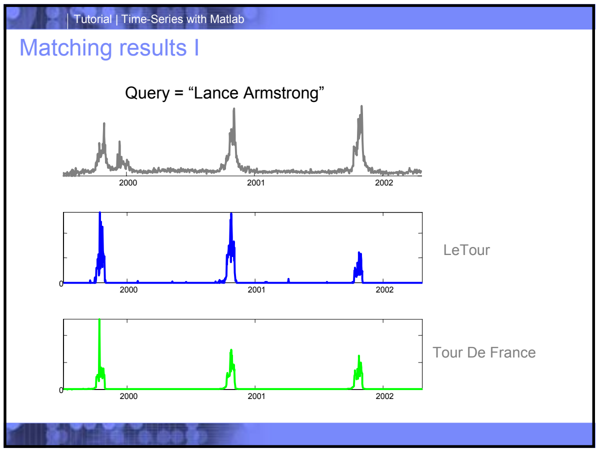
Instead of using the first Fourier coefficients we can use the best ones instead.

Let's see how the approximation will look:

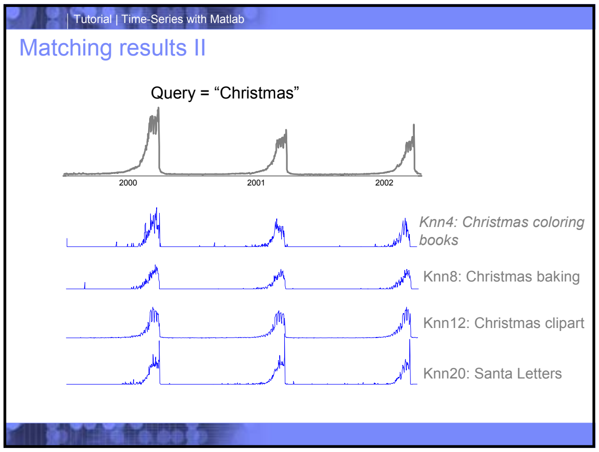
First Fourier Coefficients vs Best Fourier Coefficients



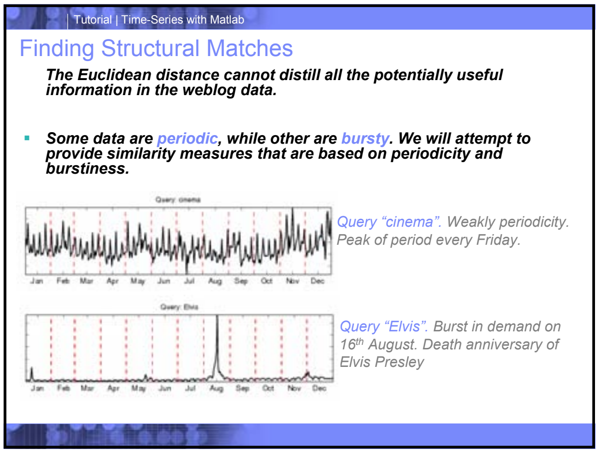
Using the best coefficients, provides a very high quality approximation of the original time-series


Tutorial | Time-Series with Matlab

Matching results I


Tutorial | Time-Series with Matlab

Matching results II


Tutorial | Time-Series with Matlab

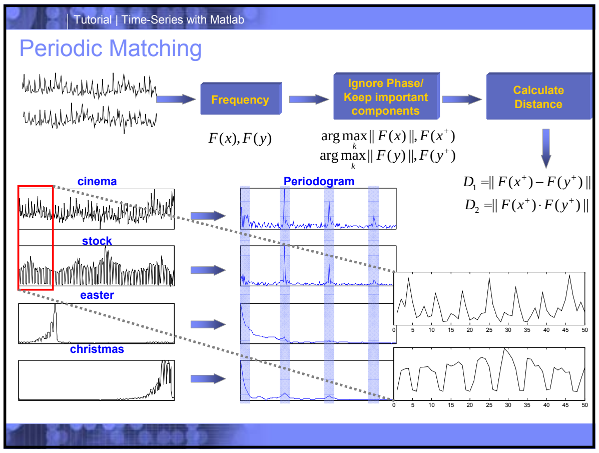
Finding Structural Matches

The Euclidean distance cannot distill all the potentially useful information in the weblog data.

- ***Some data are periodic, while other are bursty. We will attempt to provide similarity measures that are based on periodicity and burstiness.***

Query “cinema”. Weakly periodicity. Peak of period every Friday.

Query “Elvis”. Burst in demand on 16th August. Death anniversary of Elvis Presley


Tutorial | Time-Series with Matlab

Periodic Matching

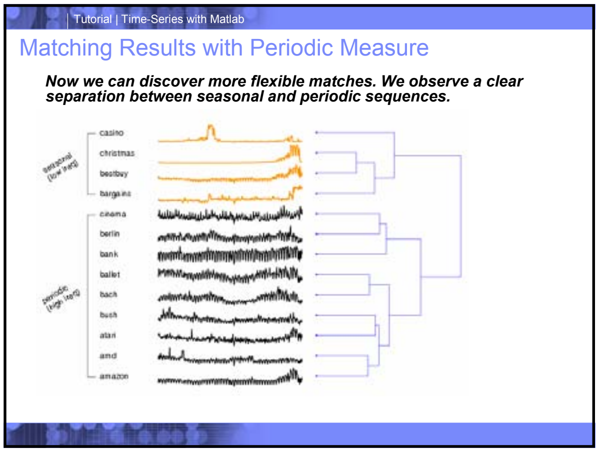
$F(x), F(y)$

$\arg\max_k \| F(x) \|, F(x^+)$

$\arg\max_k \| F(y) \|, F(y^+)$

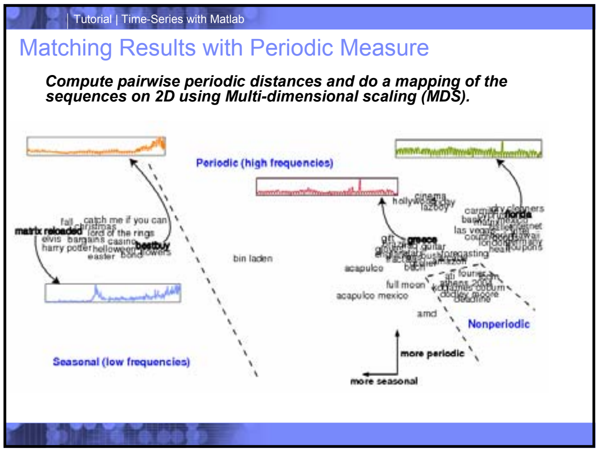
$D_1 = \| F(x^+) - F(y^+) \|$

$D_2 = \| F(x^+) \cdot F(y^+) \|$


Tutorial | Time-Series with Matlab

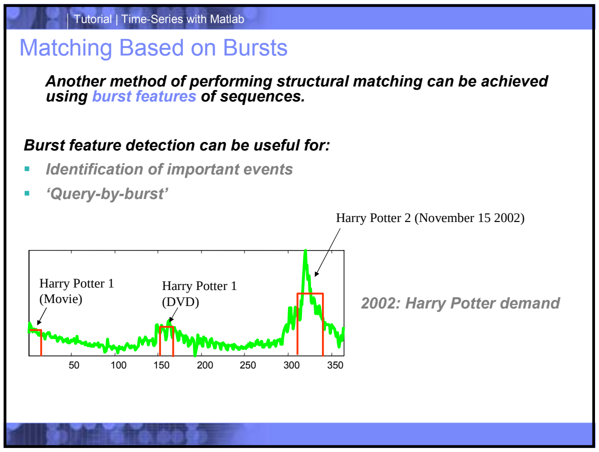
Matching Results with Periodic Measure

Now we can discover more flexible matches. We observe a clear separation between seasonal and periodic sequences.


Tutorial | Time-Series with Matlab

Matching Results with Periodic Measure

Compute pairwise periodic distances and do a mapping of the sequences on 2D using Multi-dimensional scaling (MDS).



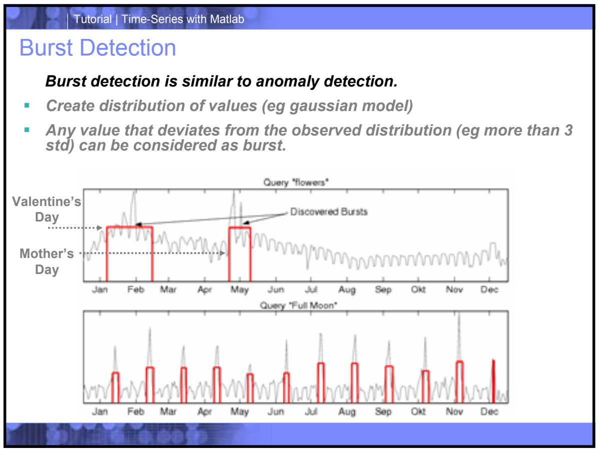
Matching Based on Bursts

Another method of performing structural matching can be achieved using *burst features* ***of sequences.***

Burst feature detection can be useful for:

- *Identification of important events*
- *'Query-by-burst'*

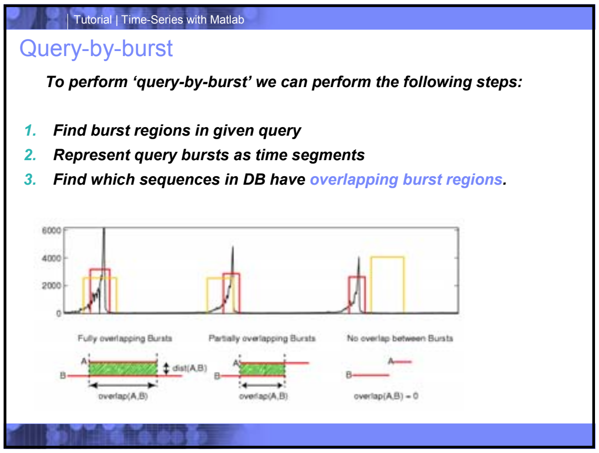
2002: Harry Potter demand



Burst Detection

Burst detection is similar to anomaly detection.

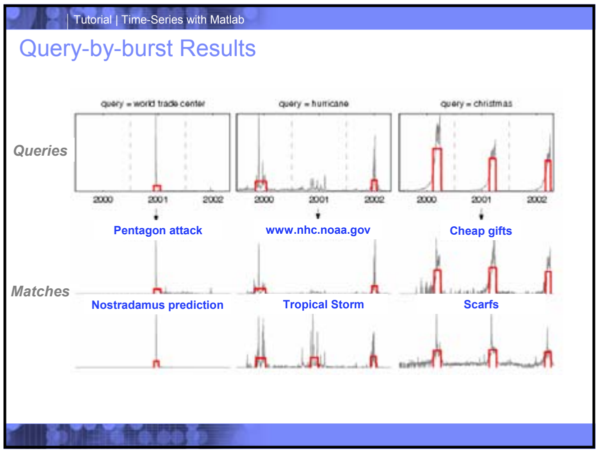
- *Create distribution of values (eg gaussian model)*
- *Any value that deviates from the observed distribution (eg more than 3 std) can be considered as burst.*



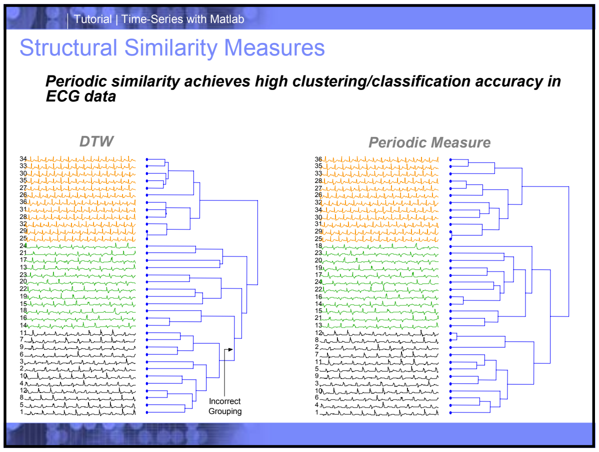
Query-by-burst

To perform 'query-by-burst' we can perform the following steps:

1. ***Find burst regions in given query***
2. ***Represent query bursts as time segments***
3. ***Find which sequences in DB have*** *overlapping burst regions*.

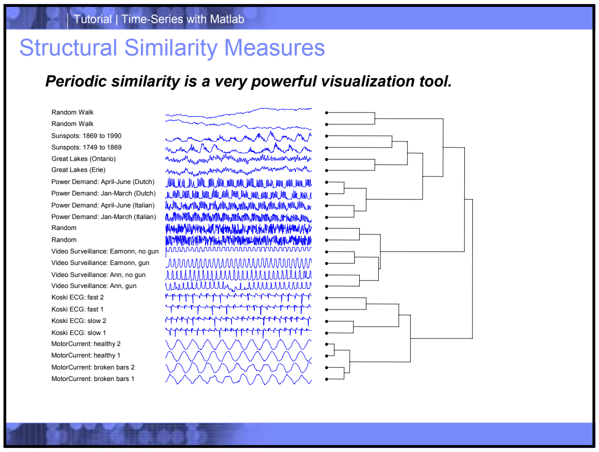


Query-by-burst Results



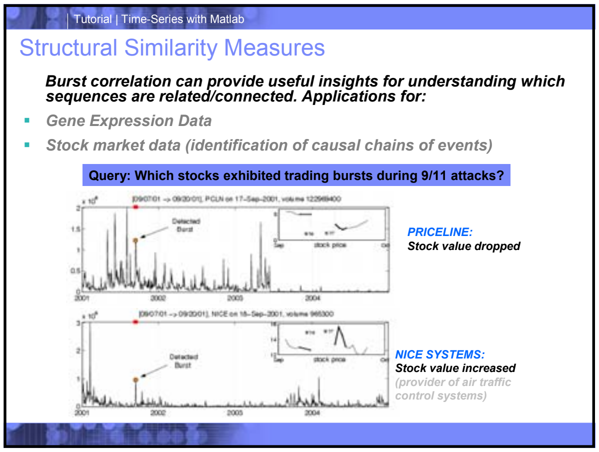
Structural Similarity Measures

Periodic similarity achieves high clustering/classification accuracy in ECG data



Structural Similarity Measures

Periodic similarity is a very powerful visualization tool.



Structural Similarity Measures

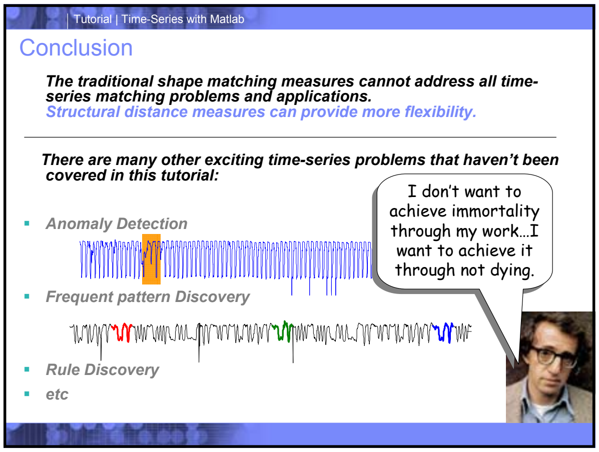
Burst correlation can provide useful insights for understanding which sequences are related/connected. Applications for:

- *Gene Expression Data*
- *Stock market data (identification of causal chains of events)*

Query: Which stocks exhibited trading bursts during 9/11 attacks?

PRICELINE:
Stock value dropped

NICE SYSTEMS:
Stock value increased
(provider of air traffic control systems)



Conclusion

The traditional shape matching measures cannot address all time-series matching problems and applications.
Structural distance measures can provide more flexibility.

There are many other exciting time-series problems that haven't been covered in this tutorial:

- *Anomaly Detection*
- *Frequent pattern Discovery*
- *Rule Discovery*
- *etc*

I don't want to achieve immortality through my work…I want to achieve it through not dying.