MFE204TC
ARTIFICIAL INTELLIGENCE AND DATA ANALYSIS

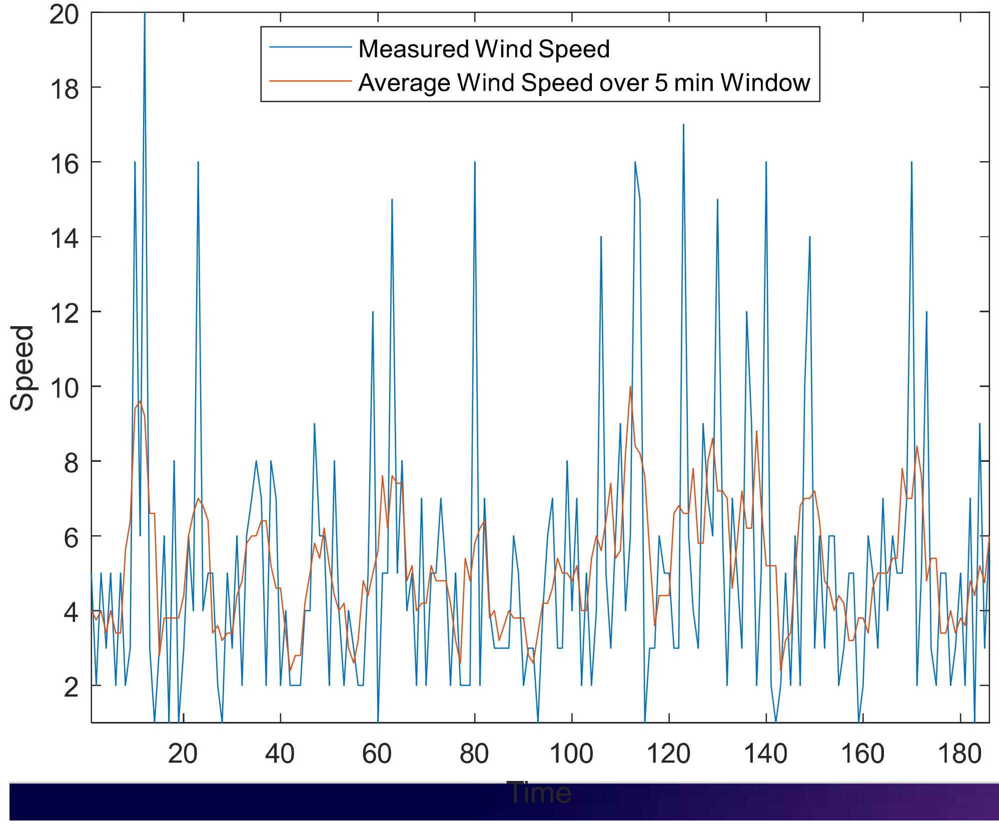
LECTURE 2

LONG HUANG

Xi’an Jiaotong-Liverpool University
西交利物浦大学

DATA SMOOTHING – MOVING WINDOW METHODS

- Data smoothing: eliminating unwanted noise or behaviors in data
- E.g Wind speed measurements taken every minute for 3 hours.

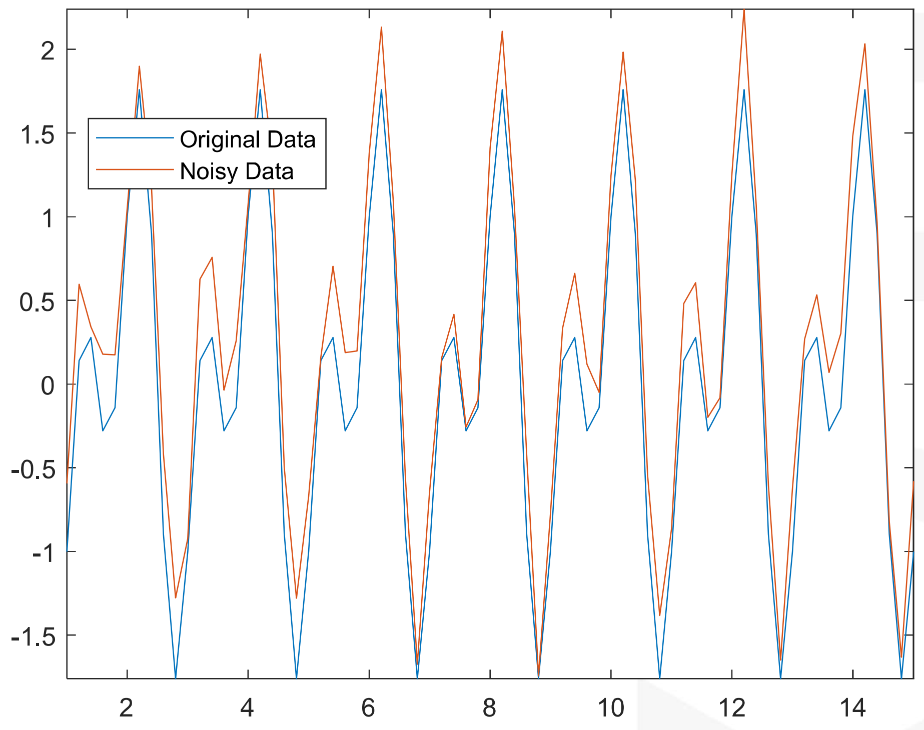


```
load windData.mat
mins = 1:length(speed);
window = 5;
meanspeed = movmean(speed,window);
plot(mins,speed,mins,meanspeed)
axis tight
legend('Measured Wind Speed','Average Wind Speed over 5 min Window','location','best')
xlabel('Time')
ylabel('Speed')
```

MOVING WINDOW METHODS – NOT ALWAYS SUITABLE

- Consider a sinusoidal signal with injected random noise.

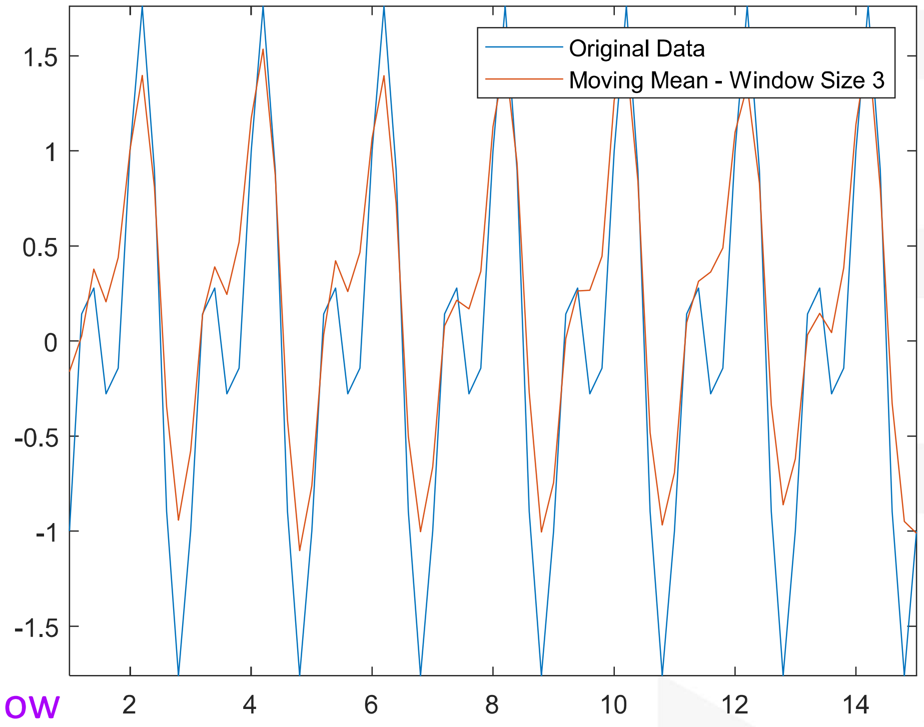
```
t = 1:0.2:15;
A = sin(2*pi*t) + cos(2*pi*0.5*t);
Anoise = A + 0.5*rand(1,length(t));
plot(t,A,t,Anoise)
axis tight
legend('Original Data','Noisy
Data','location','best')
```



MOVING WINDOW METHODS – NOT ALWAYS SUITABLE

- Use a moving mean with a window size of 3 to smooth the noisy data.

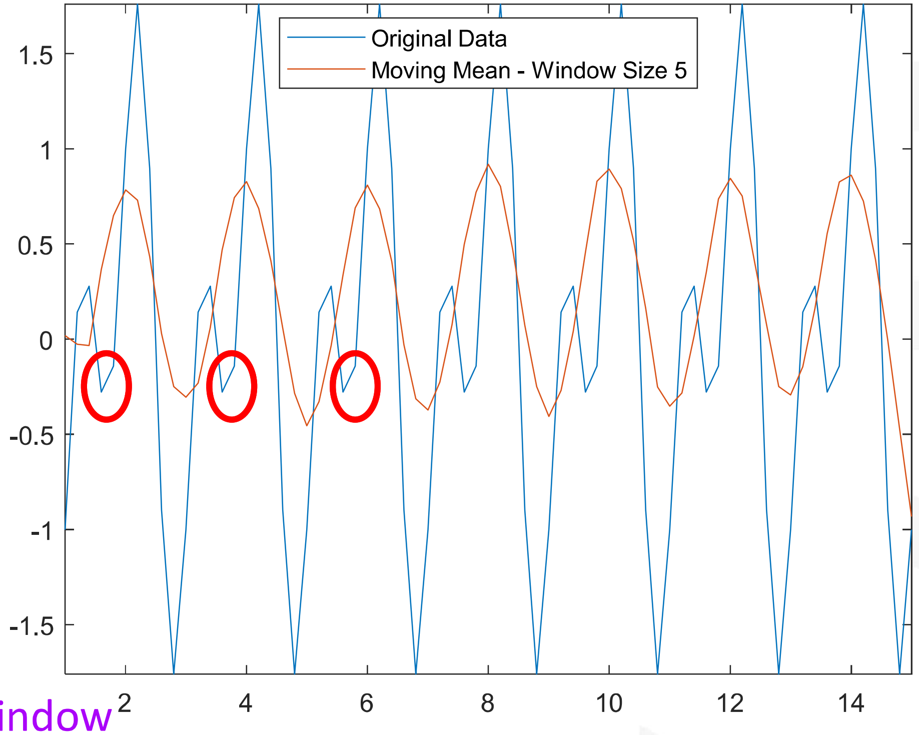
```
t = 1:0.2:15;
A = sin(2*pi*t) + cos(2*pi*0.5*t);
Anoise = A + 0.5*rand(1,length(t));
plot(t,A,t,Anoise)
axis tight
legend('Original Data','Noisy
Data','location','best')
window = 3;
Amean = movmean(Anoise,window);
plot(t,A,t,Amean)
axis tight
legend('Original Data','Moving Mean - Window
Size 3')
```



MOVING WINDOW METHODS – NOT ALWAYS SUITABLE

- Larger window? Smoothed but the valley points are gone..

```
t = 1:0.2:15;
A = sin(2*pi*t) + cos(2*pi*0.5*t);
Anoise = A + 0.5*rand(1,length(t));
plot(t,A,t,Anoise)
axis tight
legend('Original Data','Noisy
Data','location','best')


Amean = movmean(Anoise,5);
plot(t,A,t,Amean)
axis tight
legend('Original Data','Moving Mean - Window
Size 5','location','best')
```

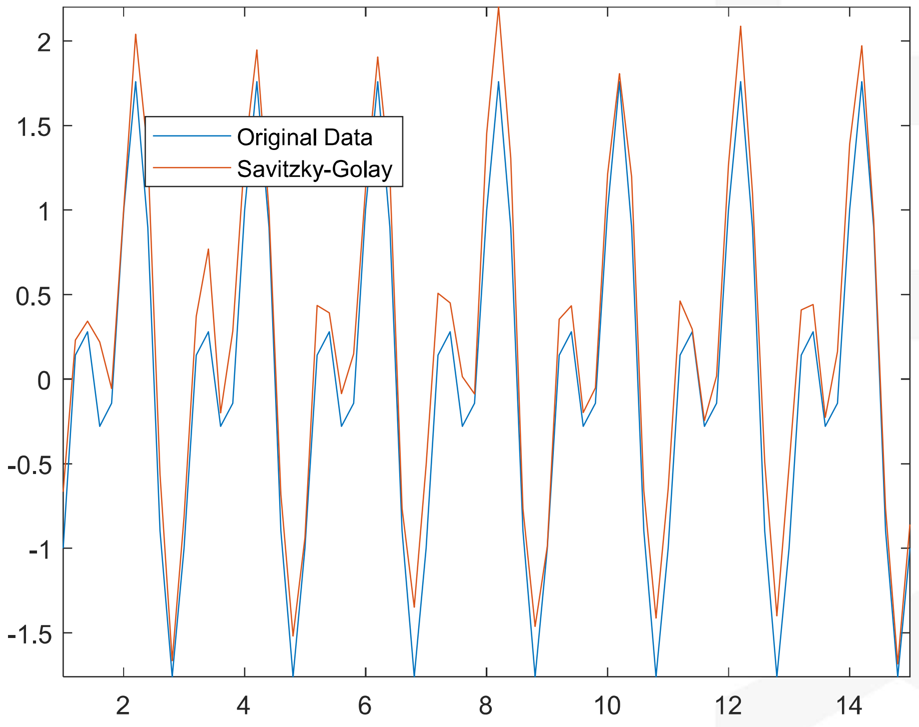


- <u>HW: Research and try different smoothing methods for this.</u>

COMMON SMOOTHING METHODS

- The *smoothdata* function provides several smoothing options
  - i.e. the Savitzky-Golay method,
  - By default, *smoothdata* chooses a best-guess window size

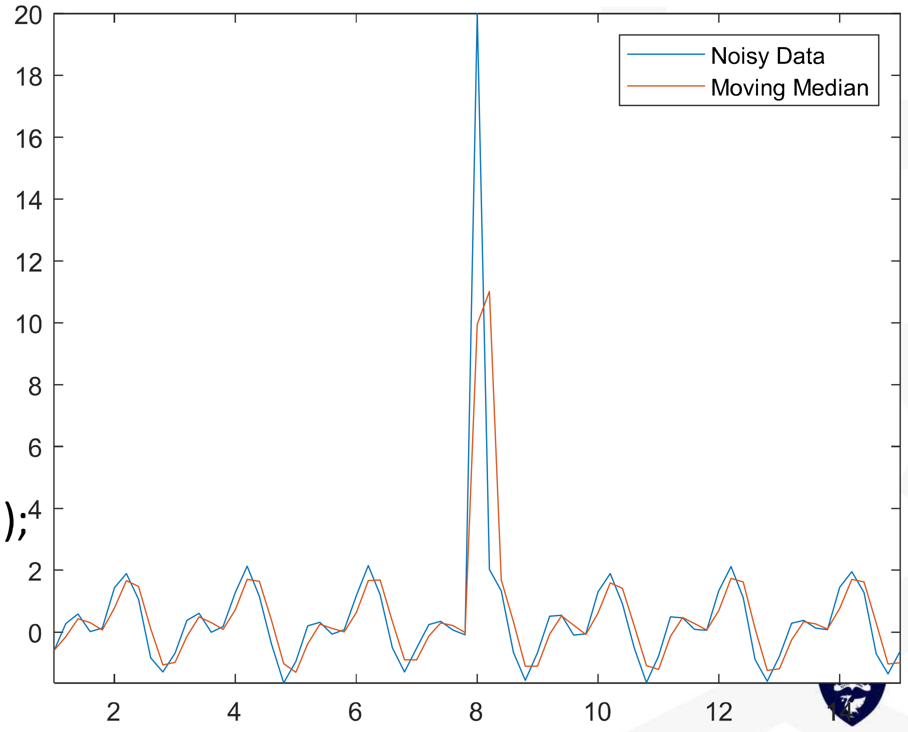
```
t = 1:0.2:15;
A = sin(2*pi*t) + cos(2*pi*0.5*t);
Anoise = A + 0.5*rand(1,length(t));
plot(t,A,t,Anoise)
axis tight
legend('Original Data','Noisy Data','location','best')


[Asgolay,window] = smoothdata(Anoise,'sgolay');
plot(t,A,t,Asgolay)
axis tight
legend('Original Data','Savitzky-
Golay','location','best')
```



OUTLIER DETECTION

- Outlier detection identifies data points that are significantly different from the rest of the data.

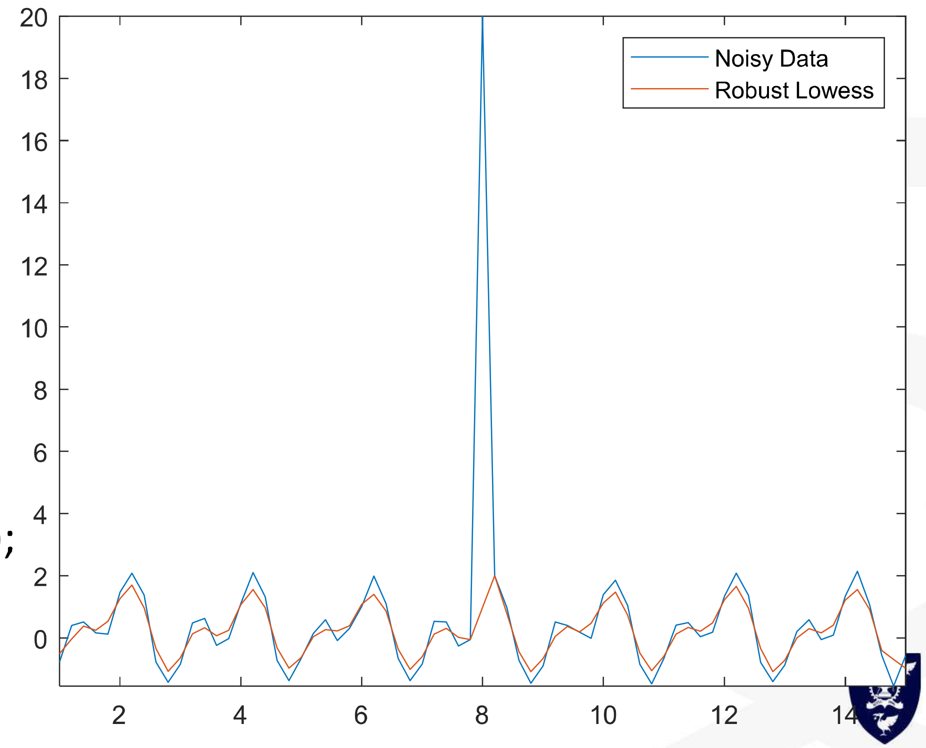
```
t = 1:0.2:15;
A = sin(2*pi*t) + cos(2*pi*0.5*t);
Anoise = A + 0.5*rand(1,length(t));
plot(t,A,t,Anoise)
axis tight
legend('Original Data','Noisy
Data','location','best')


Anoise(36) = 20;
Amedian = smoothdata(Anoise,'movmedian');
plot(t,Anoise,t,Amedian)
axis tight
legend('Noisy Data','Moving Median')
```



OUTLIER DETECTION

- Robust Lowess method is another smoothing method that is particularly helpful when outliers are present in the data in addition to noise.

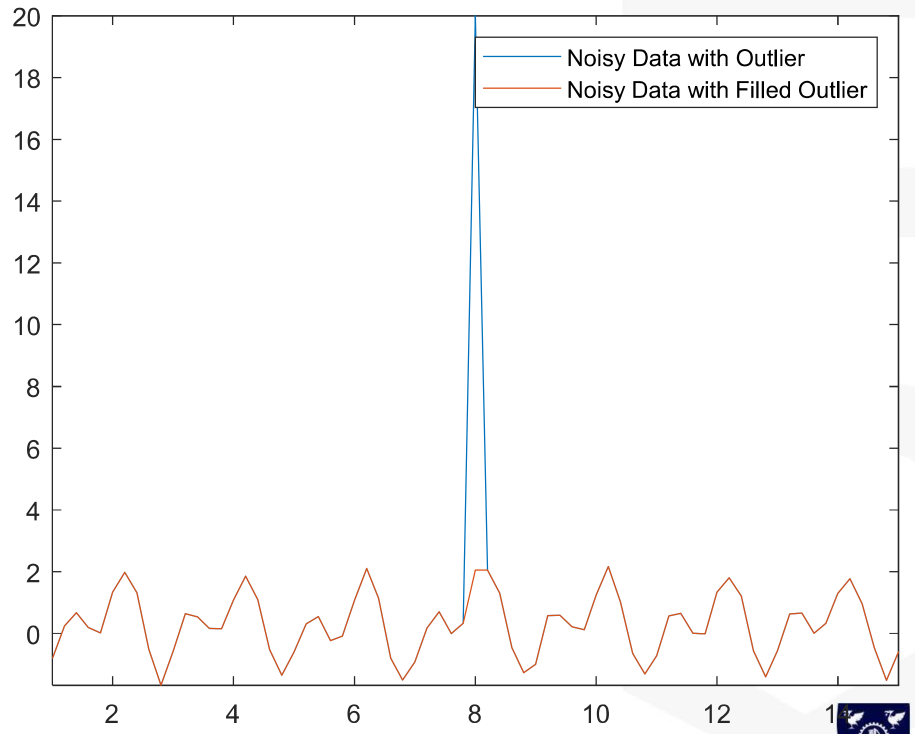
```
t = 1:0.2:15;
A = sin(2*pi*t) + cos(2*pi*0.5*t);
Anoise = A + 0.5*rand(1,length(t));
plot(t,A,t,Anoise)
axis tight
legend('Original Data','Noisy
Data','location','best')


Anoise(36) = 20;
Arlowess = smoothdata(Anoise,'rlowess',5);
plot(t,Anoise,t,Arlowess)
axis tight
legend('Noisy Data','Robust Lowess')
```



OUTLIER DETECTION

- Use the *filloutliers* function to replace outliers in your data by specifying a fill method.

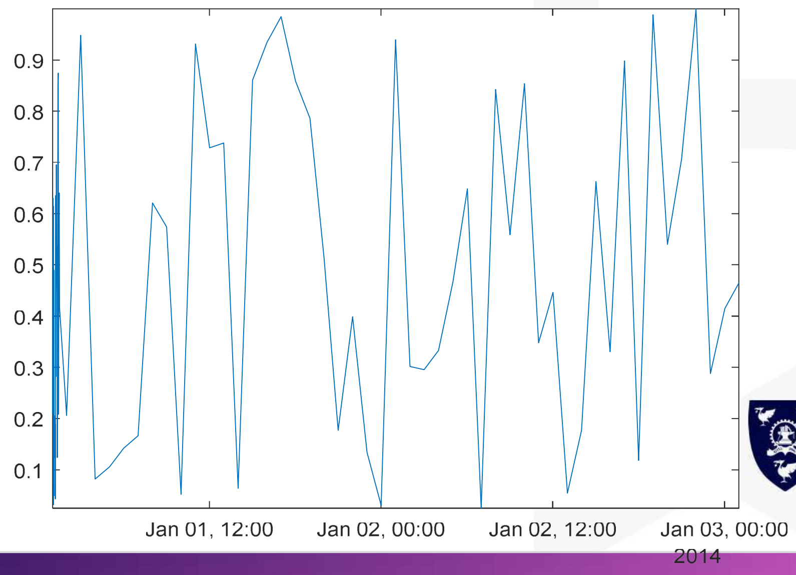
```
t = 1:0.2:15;
A = sin(2*pi*t) + cos(2*pi*0.5*t);
Anoise = A + 0.5*rand(1,length(t));
plot(t,A,t,Anoise)
axis tight
legend('Original Data','Noisy Data','location','best')


Anoise(36) = 20;
TF = isoutlier(Anoise);
ind = find(TF)
Aoutlier = Anoise(ind)
Afill = filloutliers(Anoise,'next');
plot(t,Anoise,t,Afill)
axis tight
legend('Noisy Data with Outlier','Noisy Data with Filled
Outlier')
```



NONUNIFORM DATA

- Not all data consists of equally spaced points, which can affect methods for data processing.
- Datetime vector that contains irregular sampling times for the data in Airreg.
- The time vector represents samples taken every minute for the first 30 minutes, then hourly over two days.

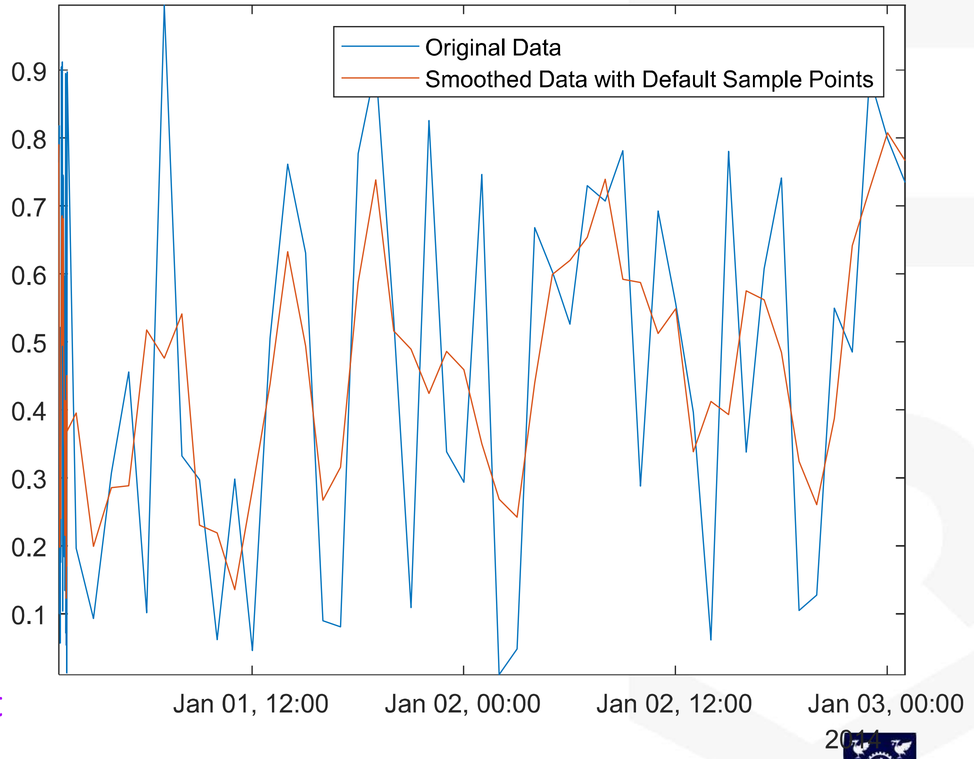
```
t0 = datetime(2014,1,1,1,1,1);
timeminutes = sort(t0 + minutes(1:30));
timehours = t0 + hours(1:48);
time = [timeminutes timehours];
Airreg = rand(1,length(time));
plot(time,Airreg)
axis tight
```



NONUNIFORM DATA

- If we still sooth with respect to equally spaced integers

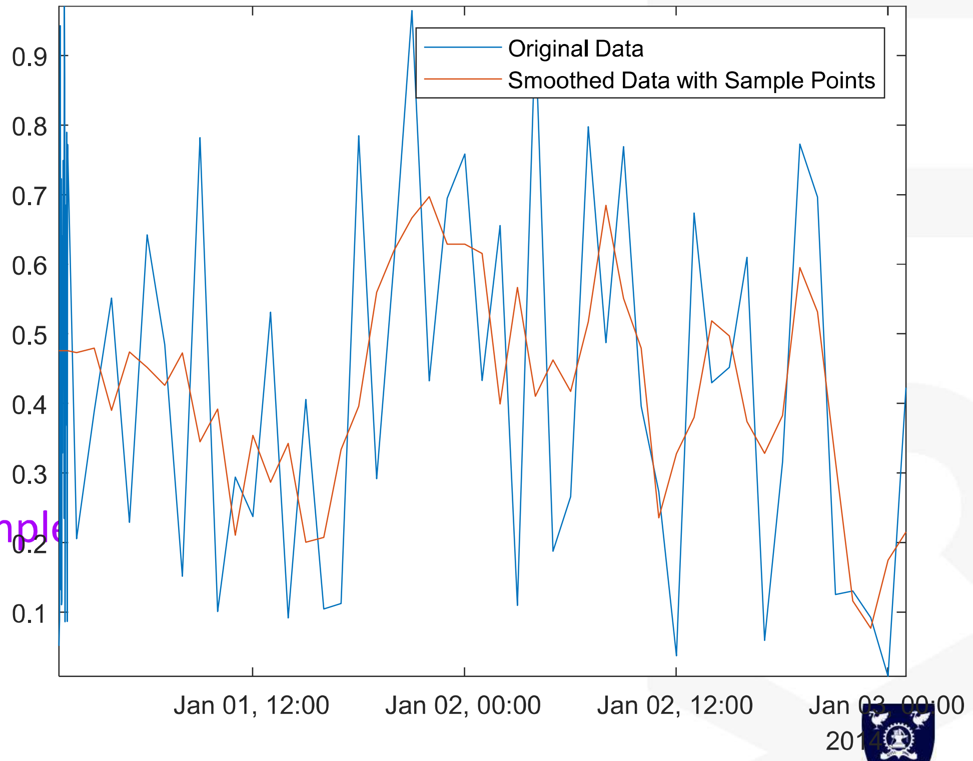
```
t0 = datetime(2014,1,1,1,1,1);
timeminutes = sort(t0 + minutes(1:30));
timehours = t0 + hours(1:48);
time = [timeminutes timehours];
Airreg = rand(1,length(time));
plot(time,Airreg)
axis tight


Adefault = smoothdata(Airreg,'movmean',3);
plot(time,Airreg,time,Adefault)
axis tight
legend('Original Data','Smoothed Data with Default
Sample Points')
```



NONUNIFORM DATA

- To remove the high-frequency variation in the first half hour of data, use the *'SamplePoints'* option with the time stamps in time.

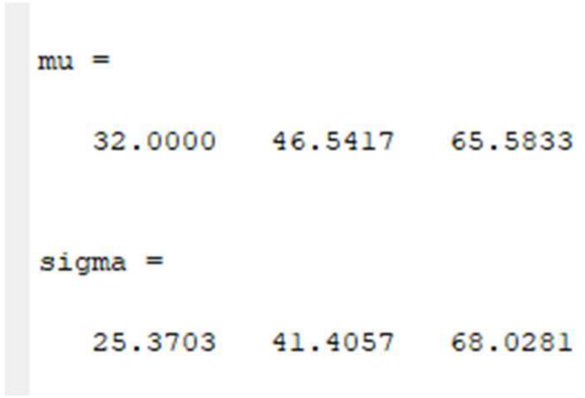
```
t0 = datetime(2014,1,1,1,1,1);
timeminutes = sort(t0 + minutes(1:30));
timehours = t0 + hours(1:48);
time = [timeminutes timehours];
Airreg = rand(1,length(time));
plot(time,Airreg)
axis tight
Asamplepoints =
smoothdata(Airreg,'movmean',hours(3),'Sample
Points',time);
plot(time,Airreg,time,Asamplepoints)
axis tight
legend('Original Data','Smoothed Data with
Sample Points')
```



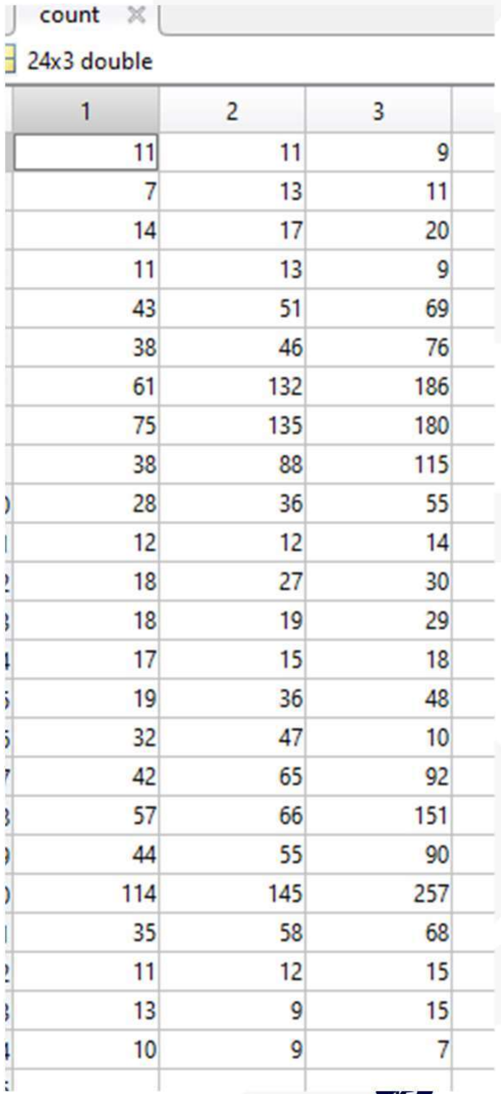
INCONSISTENT DATA

- Similar considerations with outliers.
  - Be cautious about the actions taken.
  - Try understanding the source of the issue.

```
% Import the sample data
load count.dat;
% Calculate the mean and the standard
deviation
% of each data column in the matrix
mu = mean(count)
sigma = std(count)
```



```
mu =

   32.0000   46.5417   65.5833


sigma =

   25.3703   41.4057   68.0281
```

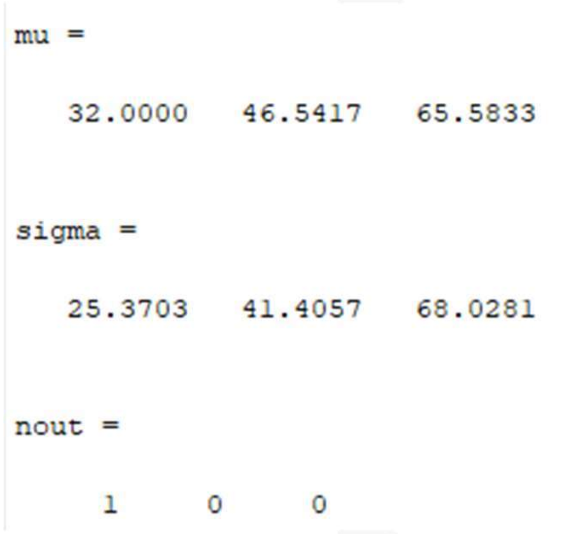


1	2	3
11	11	9
7	13	11
14	17	20
11	13	9
43	51	69
38	46	76
61	132	186
75	135	180
38	88	115
28	36	55
12	12	14
18	27	30
18	19	29
17	15	18
19	36	48
32	47	10
42	65	92
57	66	151
44	55	90
114	145	257
35	58	68
11	12	15
13	9	15
10	9	7

INCONSISTENT DATA

- Find out the number of outliers

```
% Import the sample data
load count.dat;
% Calculate the mean and the standard deviation
% of each data column in the matrix
mu = mean(count)
sigma = std(count)


[n,p] = size(count);
% Create a matrix of mean values by
% replicating the mu vector for n rows
MeanMat = repmat(mu,n,1);
% Create a matrix of standard deviation values by
% replicating the sigma vector for n rows
SigmaMat = repmat(sigma,n,1);
% Create a matrix of zeros and ones, where ones indicate
% the location of outliers
outliers = abs(count - MeanMat) > 3*SigmaMat;
% Calculate the number of outliers in each column
nout = sum(outliers)
```



```
mu =

   32.0000   46.5417   65.5833


sigma =

   25.3703   41.4057   68.0281


nout =

     1     0     0
```

FILTER DATA

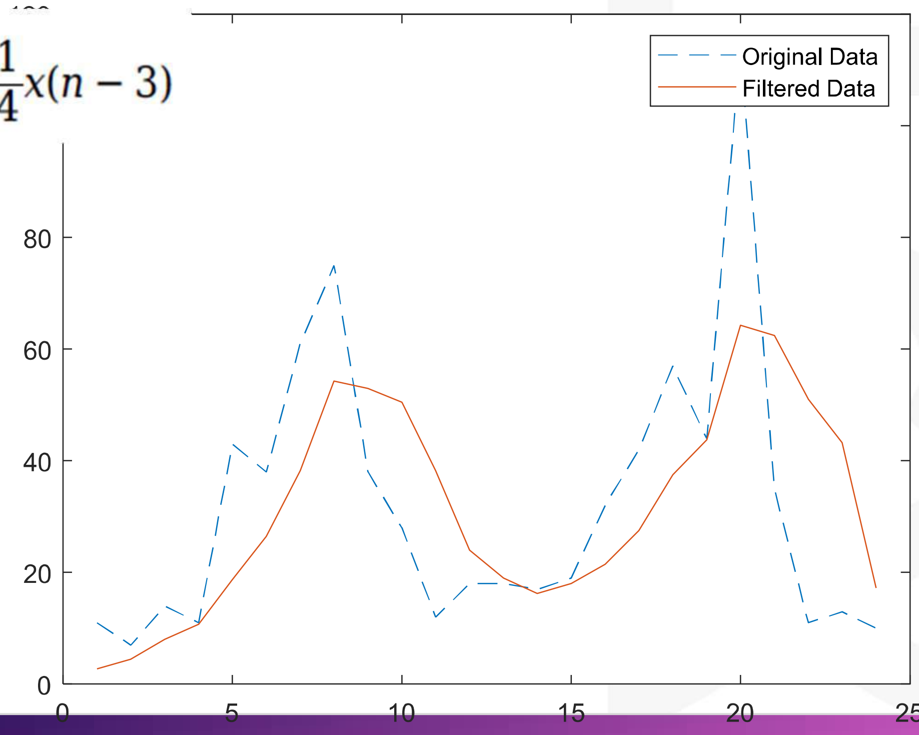
- Filter Difference Equation

$$a(1)y(n) = b(1)x(n) + b(2)x(n-1) + \ldots + b(N_b)x(n - N_b + 1)$$
$$-a(2)y(n-1) - \ldots - a(N_a)y(n - N_a + 1)$$

- Put this into use to understand how the coefficients work…

$$y(n) = \frac{1}{4}x(n) + \frac{1}{4}x(n-1) + \frac{1}{4}x(n-2) + \frac{1}{4}x(n-3)$$

```
load count.dat
x = count(:,1);
a = 1;
b = [1/4 1/4 1/4 1/4];
y = filter(b,a,x);
t = 1:length(x);
plot(t,x,'--',t,y,'-')
legend('Original Data','Filtered Data')
```



MODIFY AMPLITUDE OF DATA

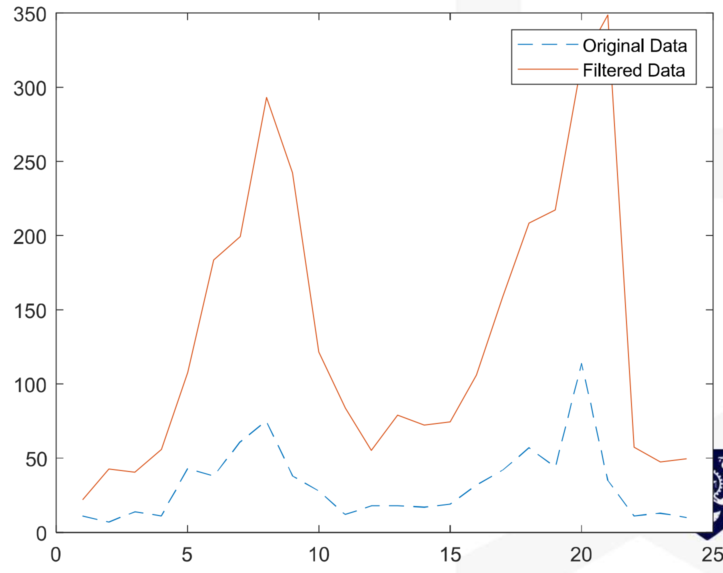
- For digital signal processing, the Z-transform of the difference equation

$$Y(z) = H(z^{-1})X(z) = \frac{b(1) + b(2)z^{-1} + \ldots + b(N_b)z^{-N_b+1}}{a(1) + a(2)z^{-1} + \ldots + a(N_a)z^{-N_a+1}}X(z)$$

- For the following transfer function:

$$H(z^{-1}) = \frac{b(z^{-1})}{a(z^{-1})} = \frac{2+3z^{-1}}{1+0.2z^{-1}}$$

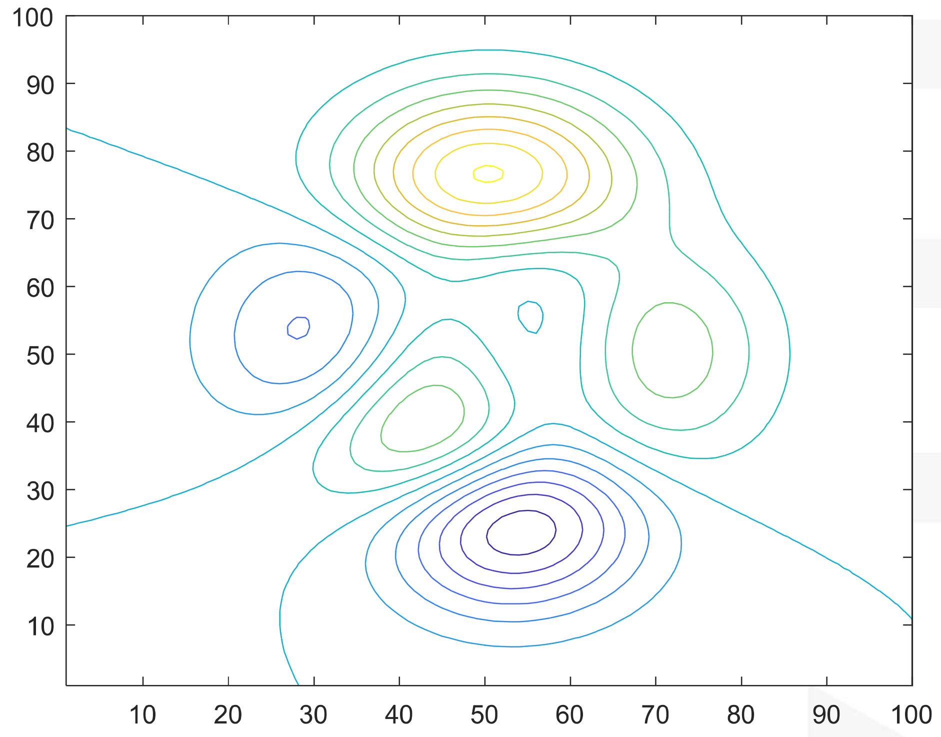
```
load count.dat
x = count(:,1);
a = [1 0.2];
b = [2 3];
y = filter(b,a,x);
t = 1:length(x);
plot(t,x,'--',t,y,'-')
legend('Original Data','Filtered Data')
```



SMOOTH DATA WITH CONVOLUTION

- Smooth 2-D data that contains high-frequency components.

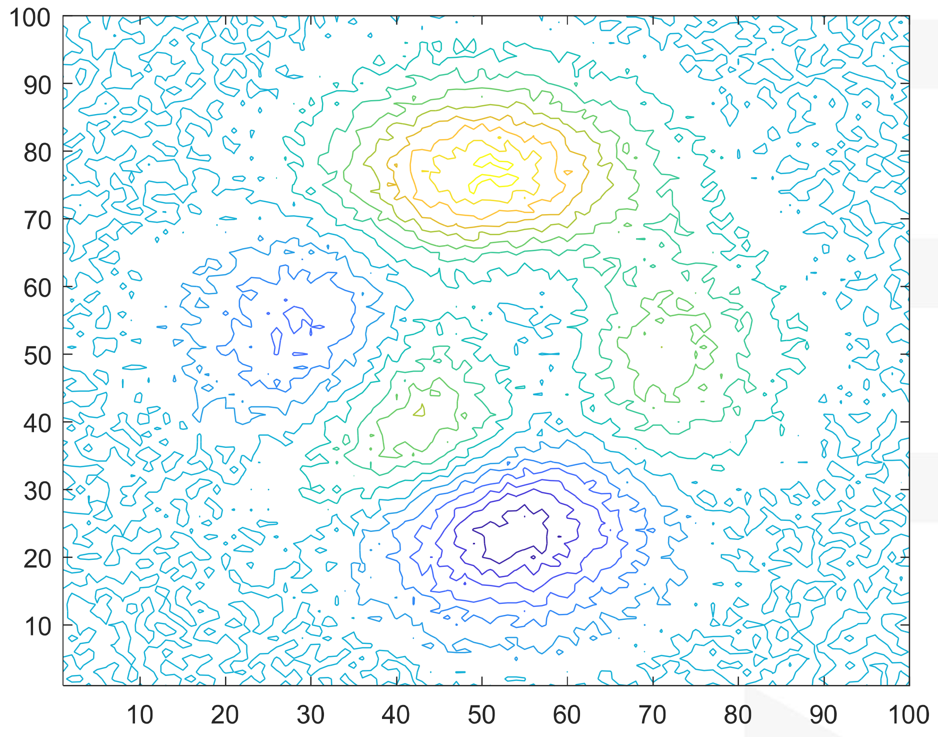
```
Z = peaks(100);
levels = -7:1:10;
contour(Z,levels)
```



SMOOTH DATA WITH CONVOLUTION

- Inject some random noise

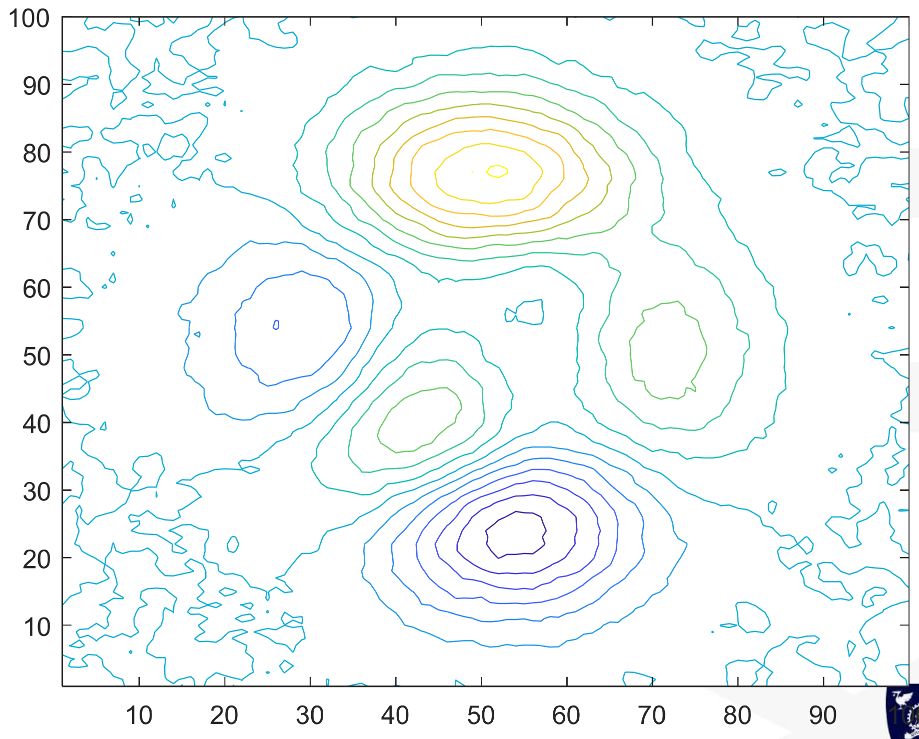
```
Z = peaks(100);
levels = -7:1:10;
Znoise = Z + rand(100) - 0.5;
contour(Znoise,levels)
```



SMOOTH DATA WITH CONVOLUTION

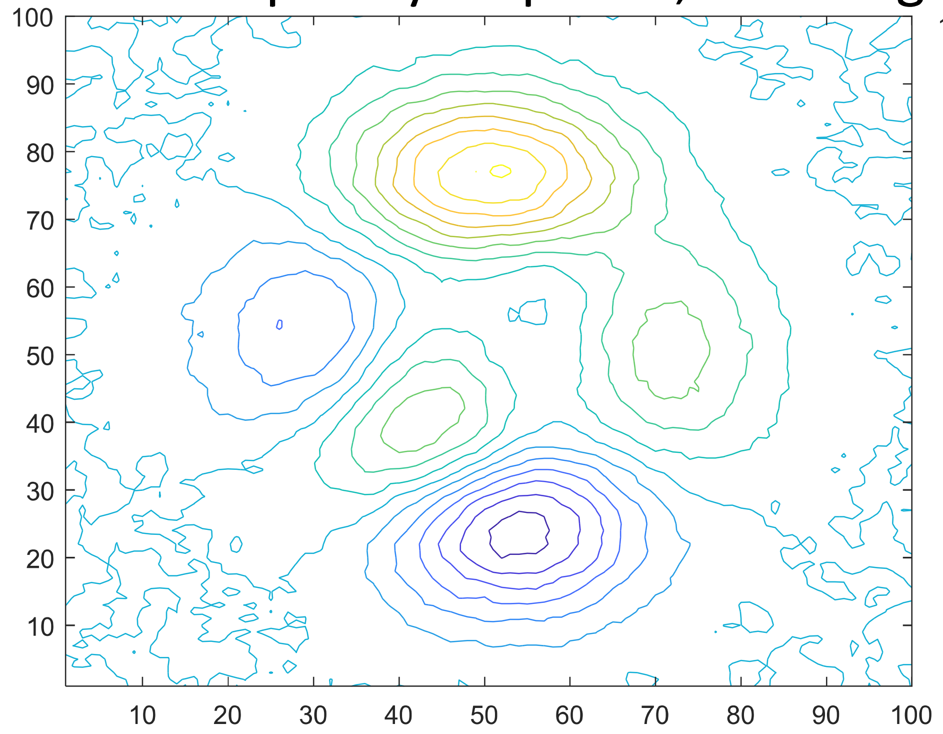
- The conv2 function in MATLAB® convolves 2-D data with a specified kernel whose elements define how to remove or enhance features of the original data.

```
Z = peaks(100);
levels = -7:1:10;
contour(Z,levels)
Znoise = Z + rand(100) - 0.5;
contour(Znoise,levels)
K = (1/9)*ones(3);
Zsmooth1 = conv2(Znoise,K,'same');
contour(Zsmooth1, levels)
```

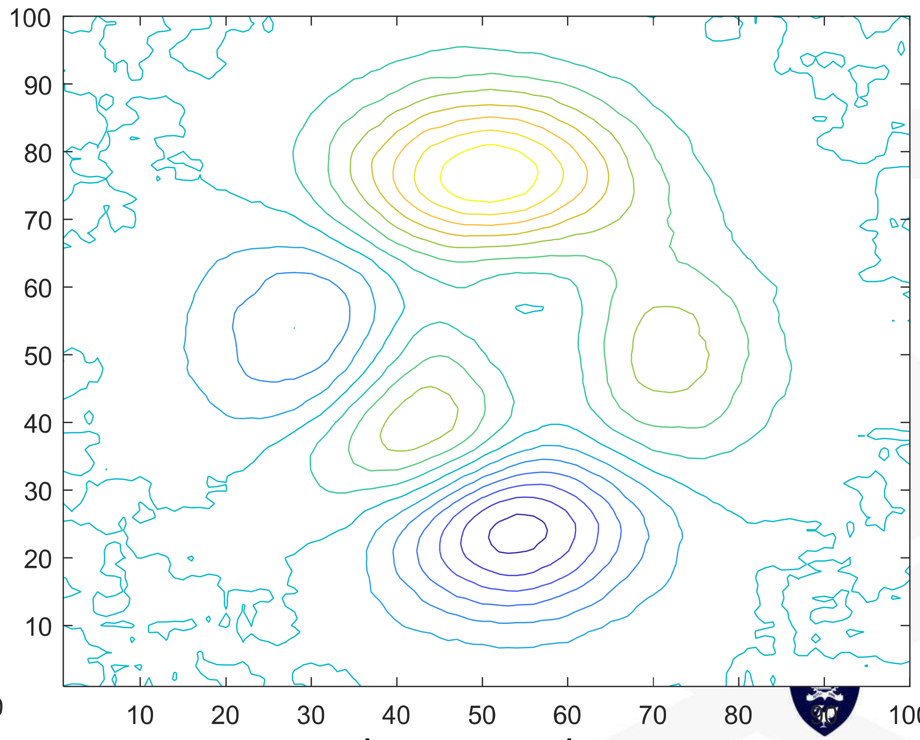


SMOOTH DATA WITH CONVOLUTION

- Small-sized kernels can be sufficient to smooth data containing only a few frequency components.
- Larger sized kernels can provide more precision for tuning frequency response, resulting in smoother output.



3 by 3 Kernel

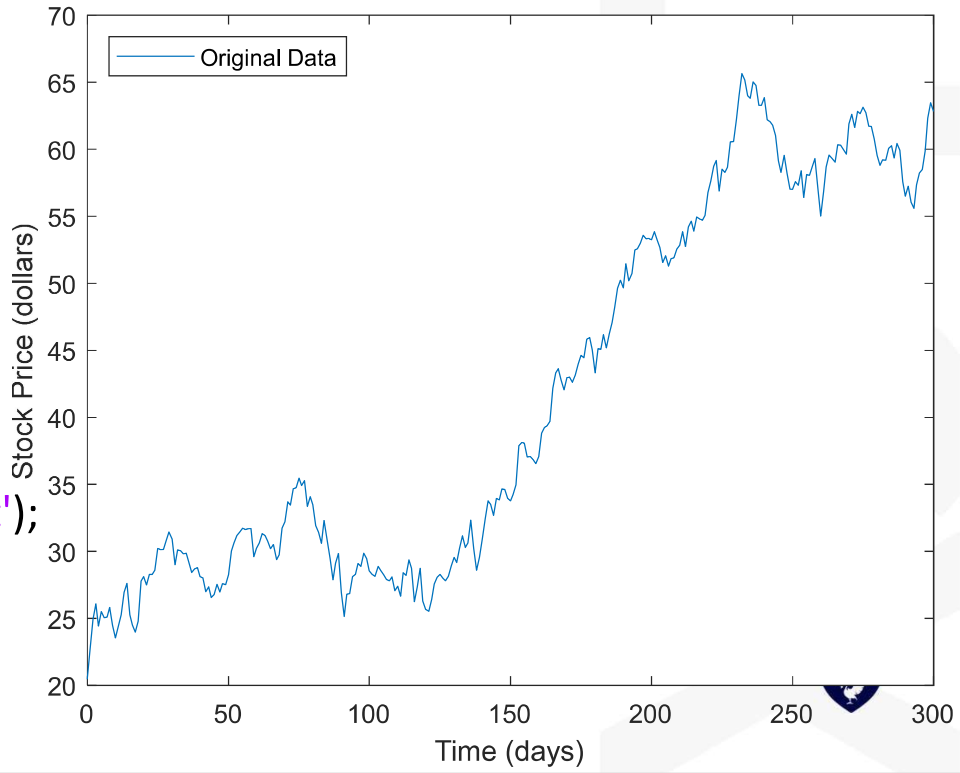


5 by 5 Kernel

DETRENDING DATA

- *detrend* subtracts the mean or a best-fit line (in the least-squares sense) from your data.
- Consider a representation of the daily price changes of a stock

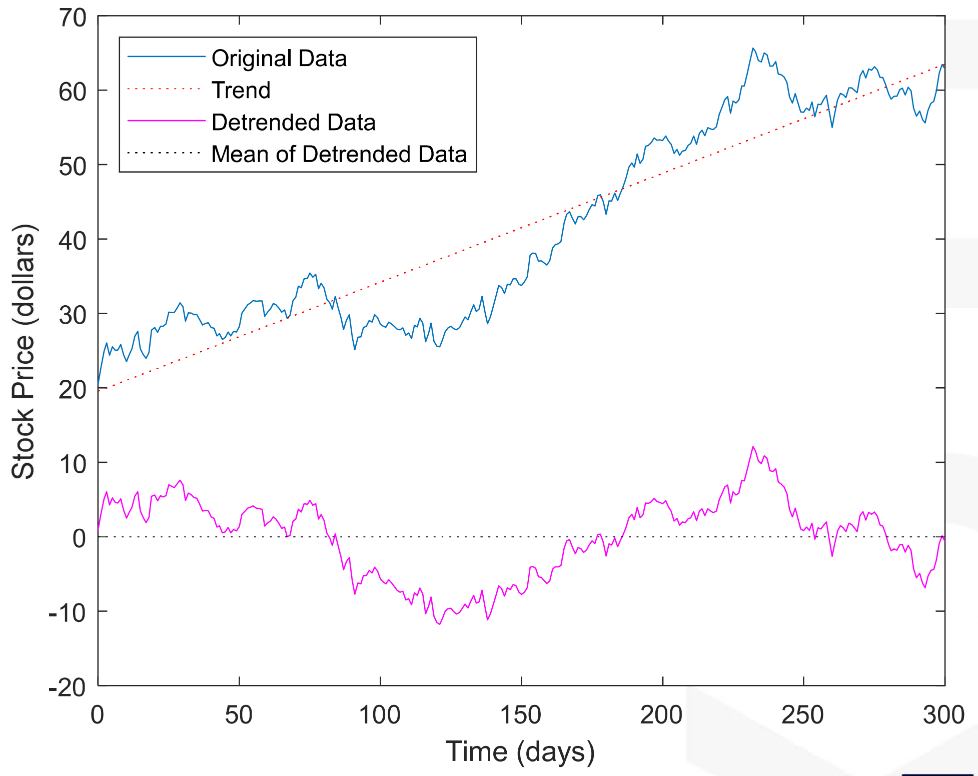
```
rng(20)
t = 0:300;
dailyFluct = randn(size(t));
sdata = cumsum(dailyFluct) + 20 + t/100;
mean(sdata)
figure
plot(t,sdata);
legend('Original Data','Location','northwest');
xlabel('Time (days)');
ylabel('Stock Price (dollars)');
```



DETRENDING DATA

- Remove Linear Trends from Data

```
rng(20)
t = 0:300;
dailyFluct = randn(size(t));
sdata = cumsum(dailyFluct) + 20 + t/100;
mean(sdata)
figure
plot(t,sdata);
legend('Original Data','Location','northwest');
xlabel('Time (days)');
ylabel('Stock Price (dollars)');
detrend_sdata = detrend(sdata);
trend = sdata - detrend_sdata;
hold on
plot(t,trend,':r')
plot(t,detrend_sdata,'m')
plot(t,zeros(size(t)),':k')
legend('Original Data','Trend','Detrended Data',...
'Mean of Detrended Data','Location','northwest')
xlabel('Time (days)');
ylabel('Stock Price (dollars)');
```


Original Data
Trend
Detrended Data
Mean of Detrended Data
Stock Price (dollars)
Time (days)