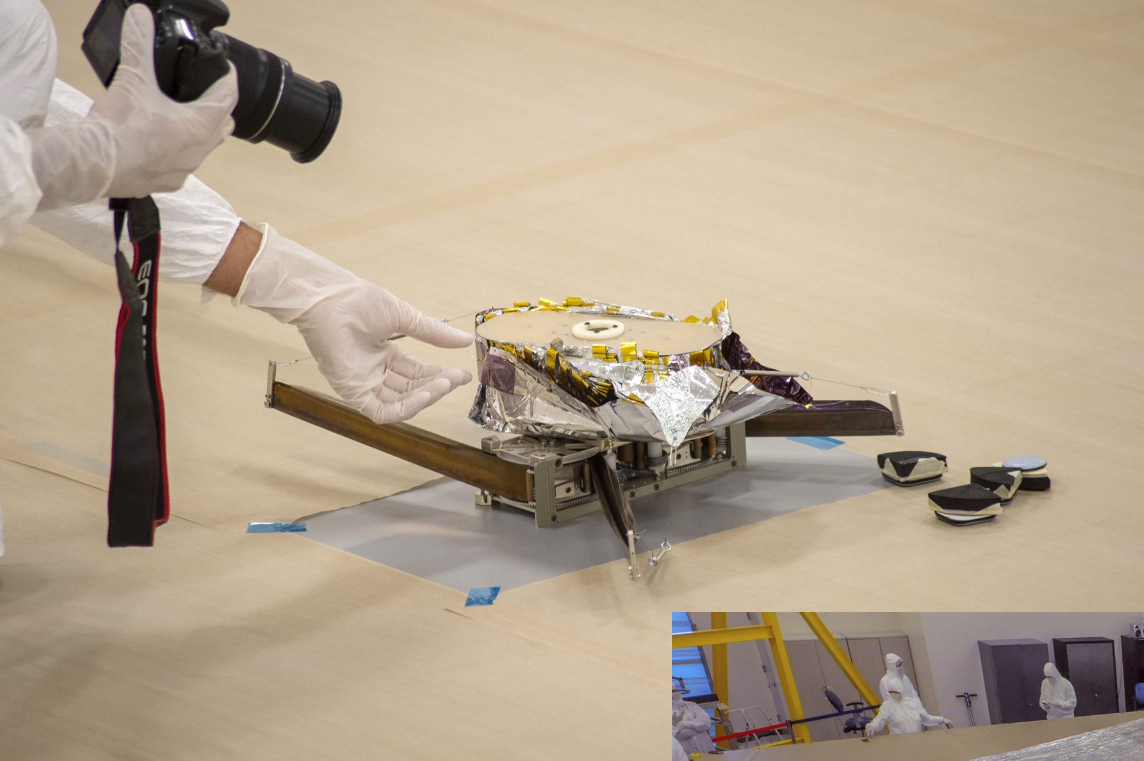
While you are waiting

Michigan Engineering
University of Michigan ■ College of Engineering

ANAGRAM

Re-arrange the letters to reveal the related word...

aloft



Solar Sail Test

ENGR 101 - Lecture 9

Data Analysis

Laura Alford, James Juett, Rick Niciejewski 10/4/17

Announcements

- Project 2 Due Thursday, 12-OCT-2017 at 11:59pm
- I shall hold office hours after lecture behind Stamps

- Todays lecture: check **00_Todays_Lecture**
  - **2 exercises, and you must Add-On a MATLAB statistics toolbox if you did not do so with Lab 00**

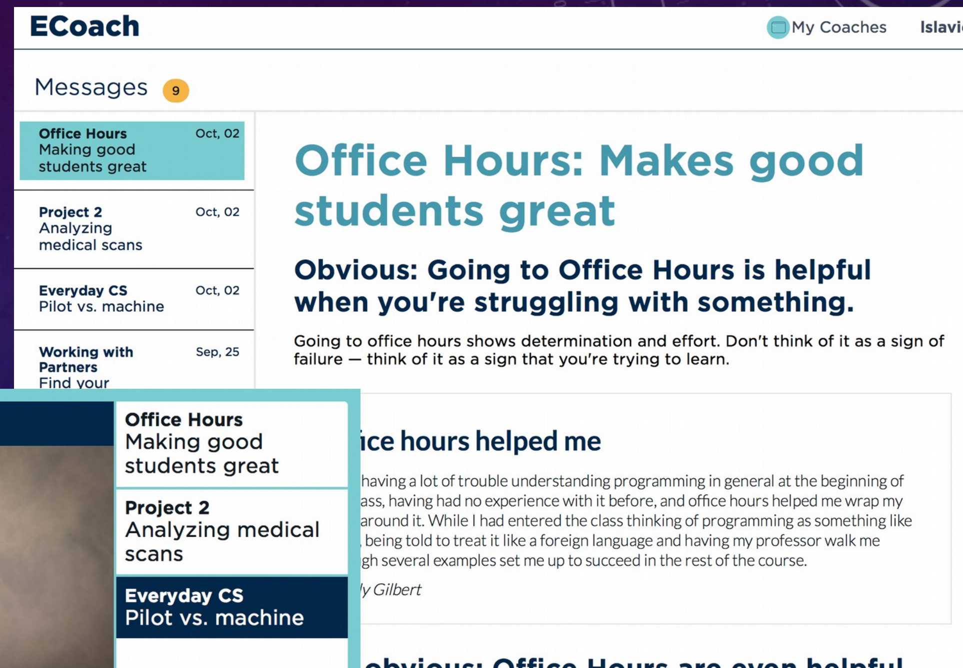
Lecture Goals

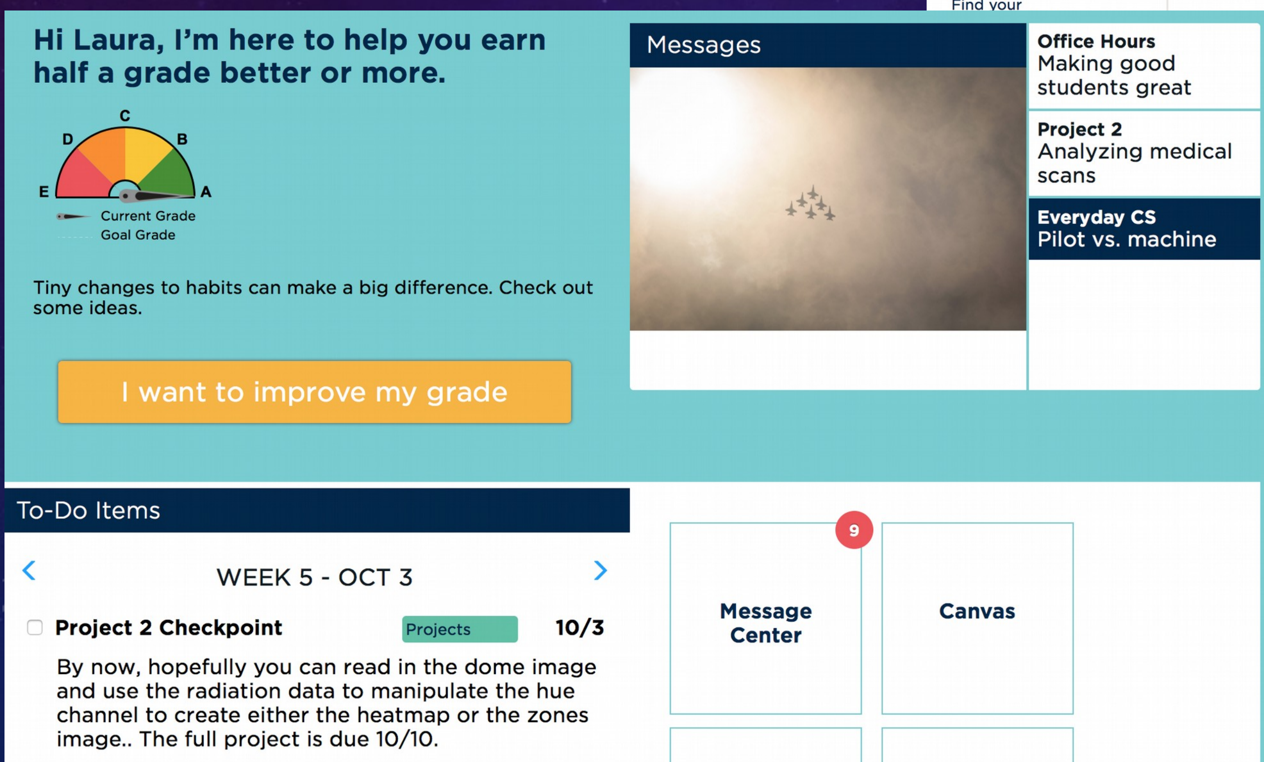
- **Today's lecture: Data Analysis**
  - **Statistical functions**
  - **Examples**
    - **Cantilever beam deflection, vertical only**
    - **Histograms**
    - **Random simulations**
  - **Suggested readings, Attaway, Chap 12.1**

- **Download today's lecture and 'beamData.mat' support files from**
  - **00_Todays_Lecture**

ECoach: Try it out if you haven't already!

The grades aren't *quite* linked up correctly with Canvas yet (this is a work in progress) but there is a TON of useful information on ECoach.





We HIGHLY recommend you sign up before Exam 1! There's some special stuff just for exam prep.

Review: `min/max`

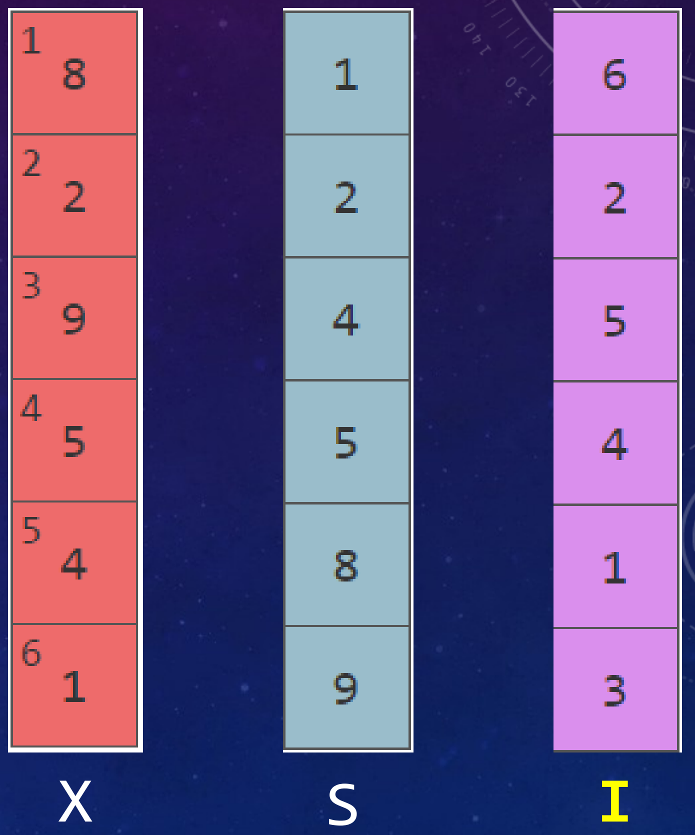
- The `min` and `max` functions can be used to find the smallest or largest elements in a vector. They too, work in each column first and must be applied twice for a whole matrix.
- `min` and `max` have a compound return value. They return both the value found, and also the **index** where it was found.

```
[m, i] = max(X)
```

Review: The `sort` Function

- By default, the `sort` function works with column vectors. (If you have a matrix, each column is sorted individually.)
- `sort` uses a compound return to give us both a sorted version of the vector AND a vector of sorted **indices**.
- You can provide 'ascend' or 'descend' as a another argument to specify order.

```
[S, I] = sort(X, 'ascend')
```



mean

- The mean function returns the column-by-column mean of a matrix. (Or for a single row, the mean of that row.)

2	3	5	3
4	2	3	3

`A = [2,3,5,3; 4,2,3,3]` or **`[2 3 5 3`**
`4 2 3 3]`

3	2.5	4	3

`mean(A)`

3.125

`mean(mean(A))`
`or mean(A(:))`

3.25
3

`mean(A,2)`

Specifying the 2nd dimension allows you to do a row-by-row mean instead.

`median`

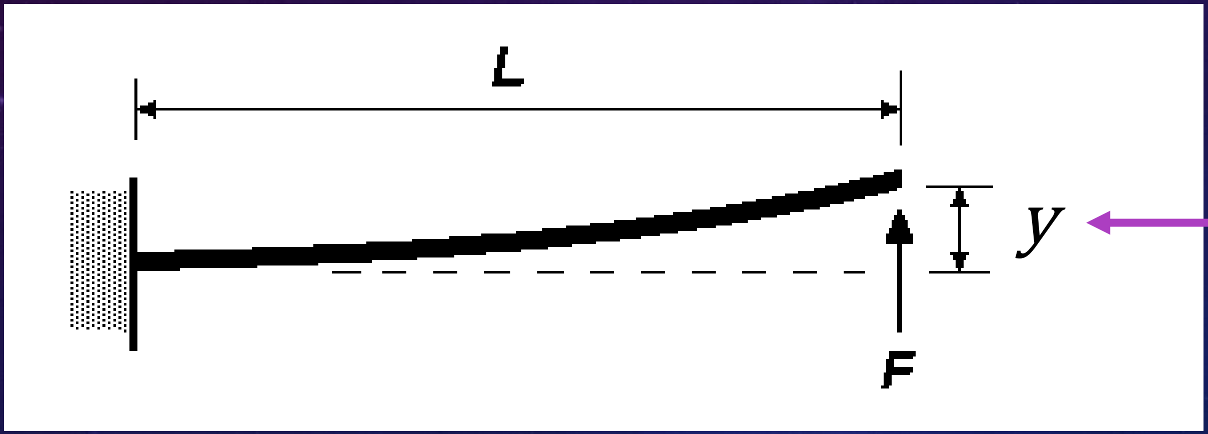
- The `median` function computes the median of a dataset.
- It works with arrays in the same way as the mean function.
  - i.e. column-by-column, selecting dimensions, etc.

- The median of a dataset is the number that would appear in the middle if the numbers were sorted.
  - In case of an even number of elements, average the two in the middle.

B	5	0	8	9	6	7	7	3	5	1
sort(B)	0	1	3	5	5	6	7	7	8	9
median(B)	5.5									

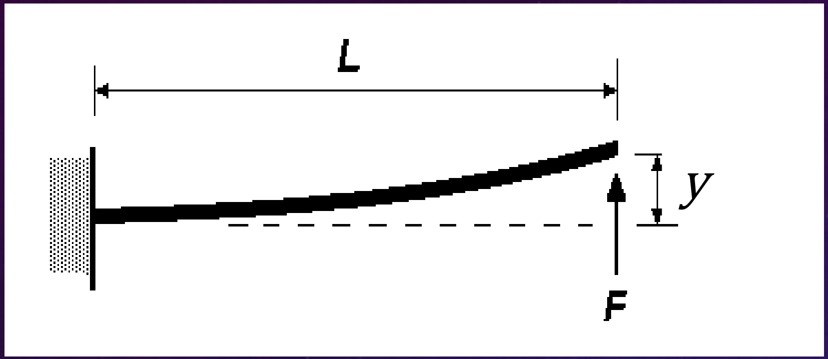
Example: Cantilevered beam deflection

push up on the end of the beam with force, F,
release, measure deflection as a function of time, t



the end will vibrate up and down a distance y

Example: Theoretical prediction



solution to *y(t)* is:

$$y(t) = y_o e^{-d\,\omega_n\,t} \cos(\sqrt{1-d}\;\omega_n t)$$

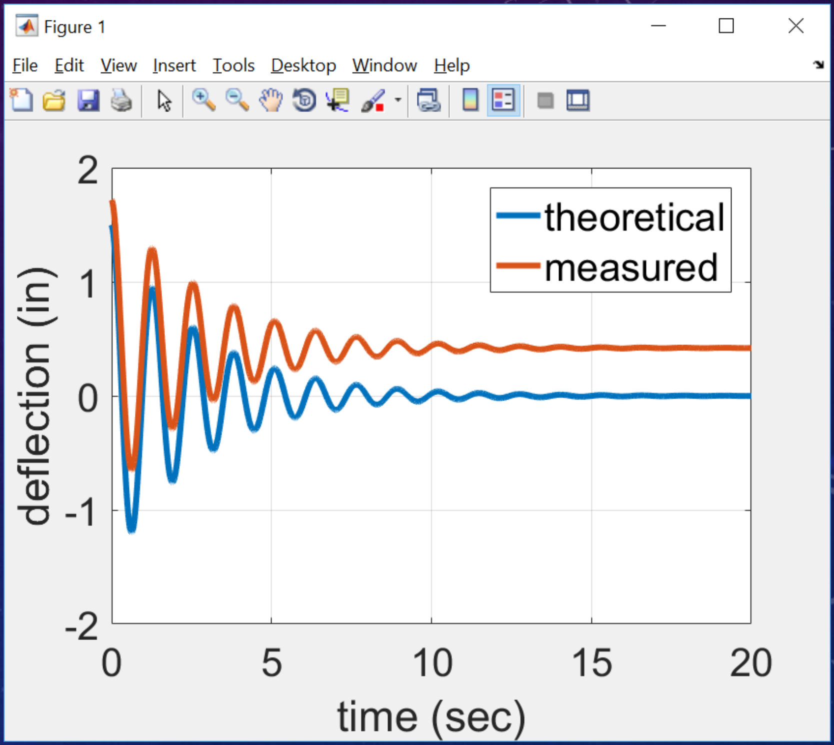
the initial distance the end of the beam was pushed up

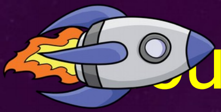
the *"damping ratio"*

the natural frequency of the beam

Example: Cantilevered beam deflection

- We want to compare lab data (00_Todays_Lecture/beamData.mat) with the theoretical prediction.
- Known problem: Our sensor is slightly off due to calibration issues.
  - In the plot, it's hard to tell if the data are aligning with theory due to an offset.
- Software fix: Find the mean of the measured data and remove it from each data point.
  - Since we assume the beam is symmetric, this will re-center the deflection data.





our turn: Beam Deflection

7 min

- Download previously measured data from a beam bending experiment: beamData.mat
  - Load it into MATLAB
- Measured data: deflection (y) and time (t)
  - Re-center the measured data by subtracting the mean
- Compare to the theoretical solution

$$y(t) = y_o e^{-d\,\omega_n\,t} \cos(\sqrt{1-d}\,\omega_n t)$$

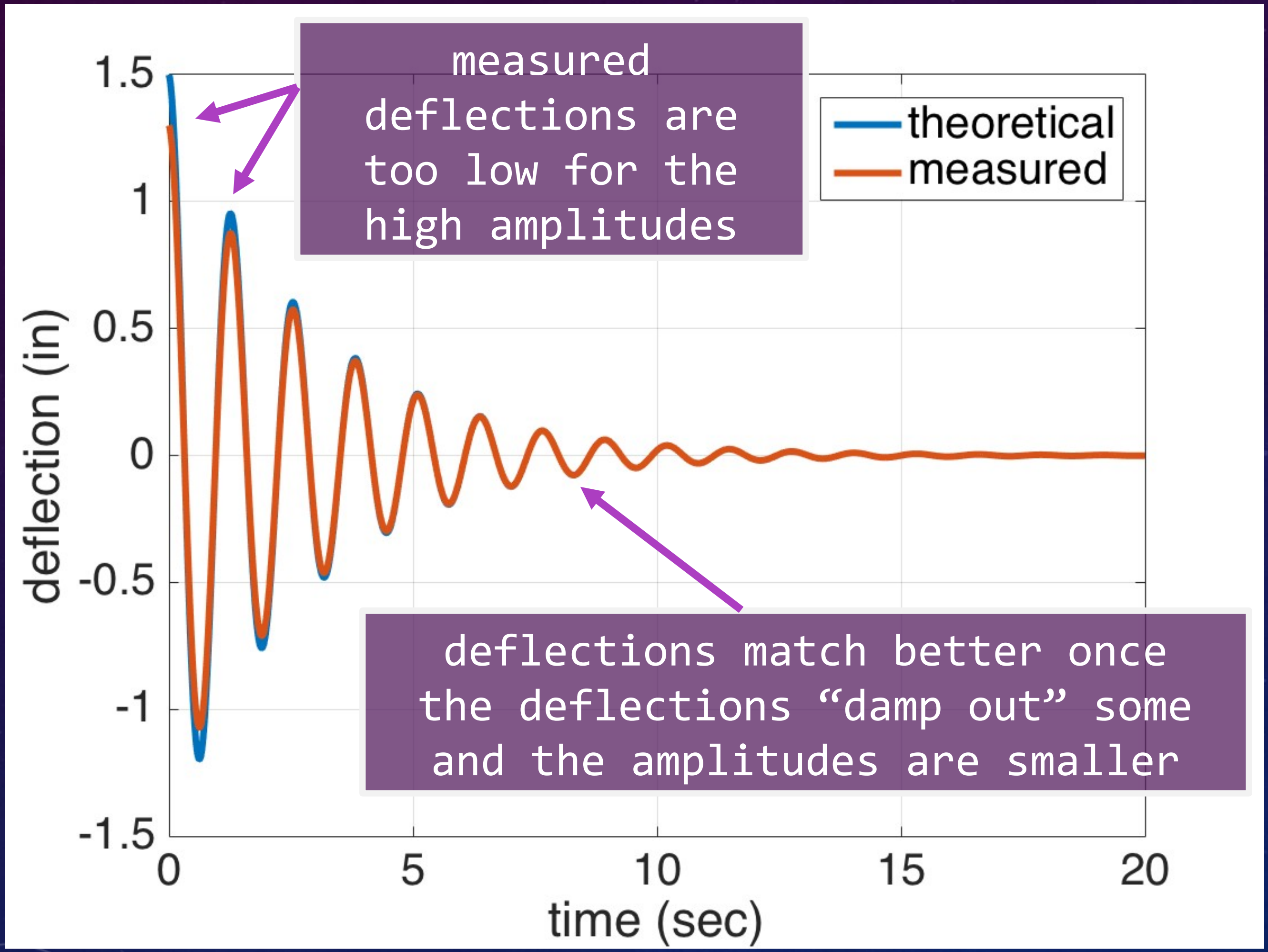
with $y_o = 1.5, d = 0.07,$ and $\omega_n = 5.1$

- Question to answer: Are $d = 0.07,$ and $\omega_n = 5.1$ the correct properties of this beam?
  - i.e. Does the measured data match the theoretical data?
  - Plot both the measured and theoretical data on the same axes to find out!

Solution: Beam Deflection

```
load('beamData');

% Substract the mean to re-center data
y = y-mean(y);

% Compute theoretical deflection over time
y0 = 1.5;
d = 0.07;
omega_n = 5.1;
yTrue = y0.*exp(-t.*d.*omega_n).*cos(sqrt(1-d).*omega_n.*t);

% Plot measured and theoretical data
p = plot(t,yTrue,t,y);

% Annotate the graph
xlabel('time (sec)','FontSize', 20);
ylabel('deflection (in)','FontSize', 20);
legend({'theoretical','measured'},'FontSize', 20);
```

Interpretation: Beam Deflection

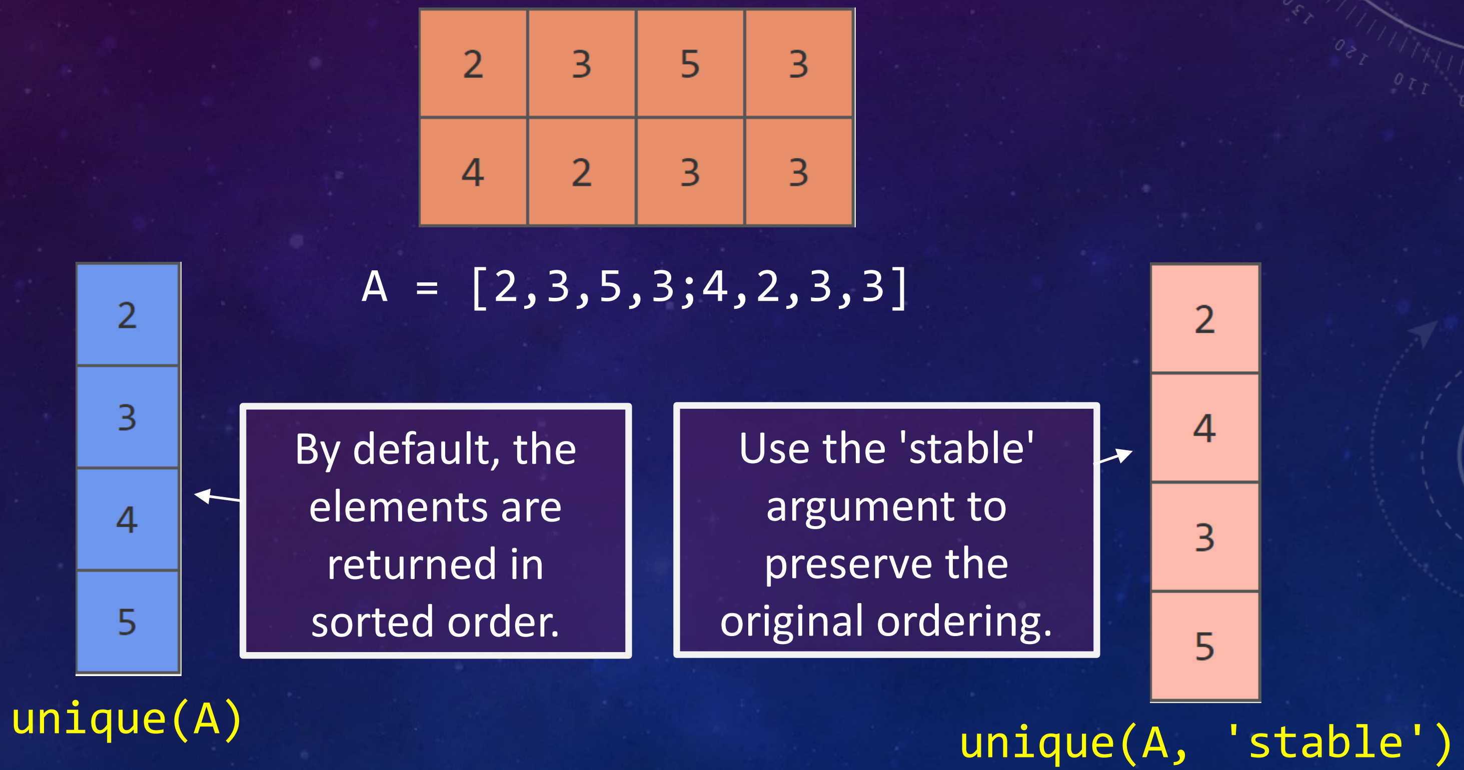


answer:
either these beam properties are not *quite* correct, or the sensor is non-linear at large deflection

need more tests to determine which one it is!

unique

- The unique function takes an array as input and returns a vector of its unique elements (i.e. with duplicates removed).



mode

- The mode of a dataset is the value that occurs most often.
- The mode function returns this value.

X	1	9	0	7	8	8	0	3	7	8

`mode(X)` 8

- Using a compound return, you can also get the frequency.

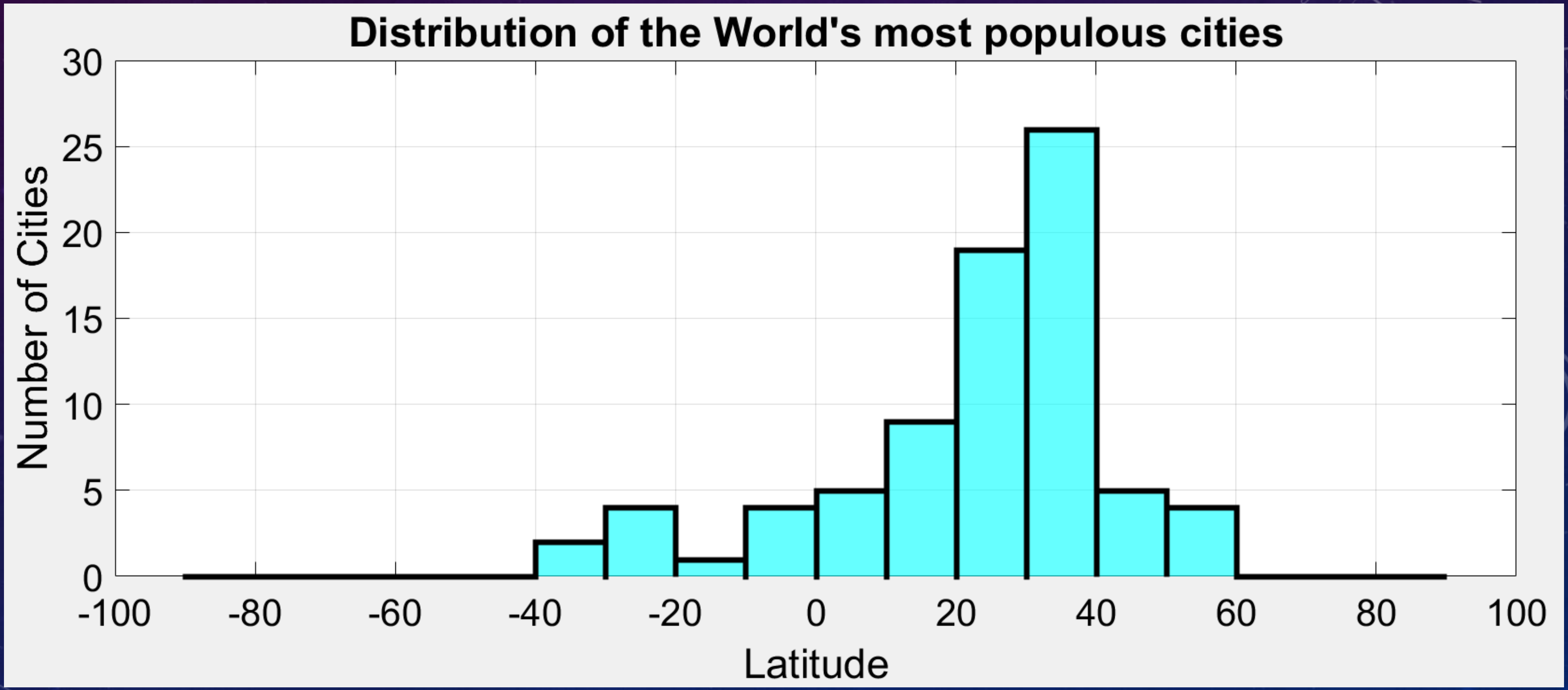
X	1	9	0	7	8	8	0	3	7	8

```
[m, freq] = mode(X)
```

m 8 freq 3

Histograms

- In general, a histogram is a visualization of the frequency of occurrence for certain values in a dataset.



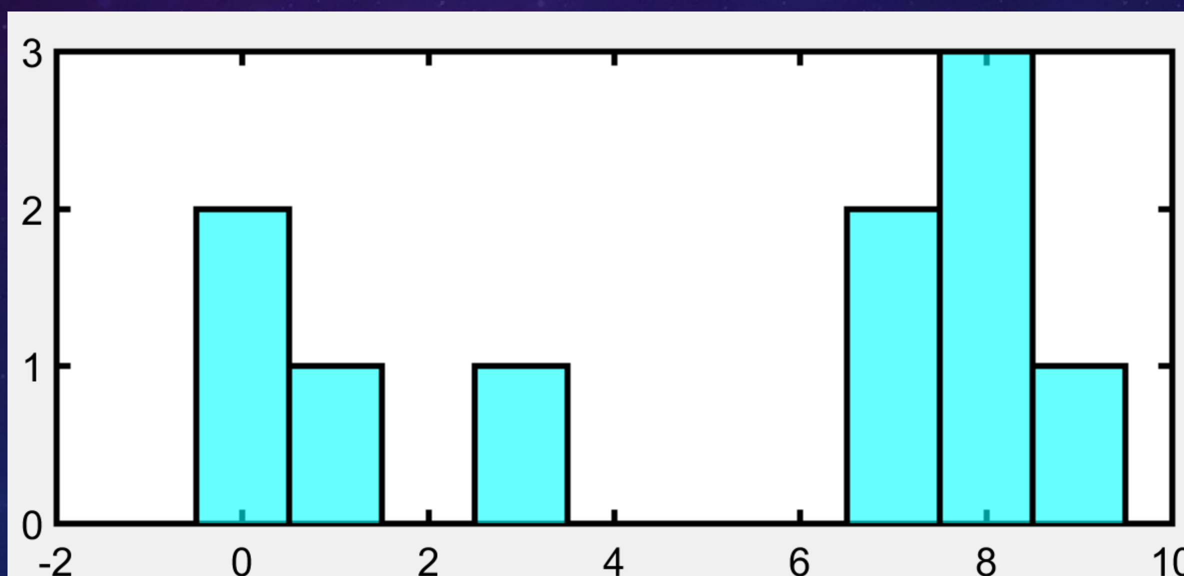
We'll make this chart in a future lecture!

Creating a Histogram

X	1	9	0	7	8	8	0	3	7	8

- Use the `histogram` function:

```
histogram(X);
```



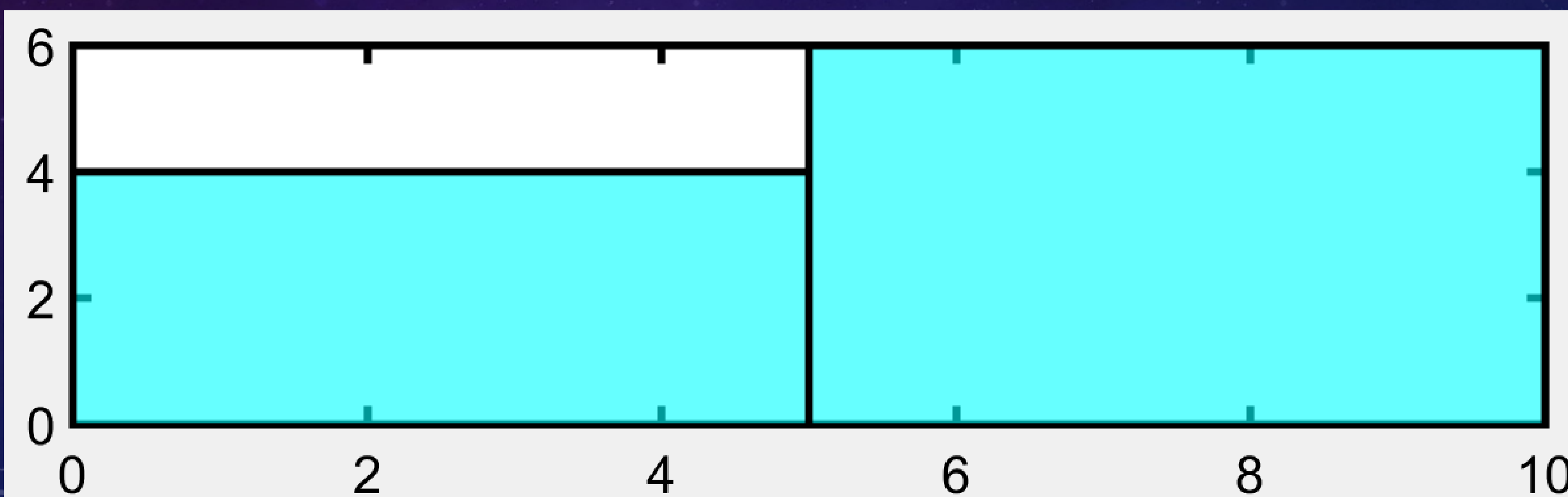
Note: There's also a `hist` function that does some of the same things, but `histogram` is generally better.

Histogram Bins

X										
X	1	9	0	7	8	8	0	3	7	8

- You can specify the number of "bins" you want to use:

```
histogram(X, 2);
```



Note: There's also a `hist` function that does some of the same things, but `histogram` is generally better.

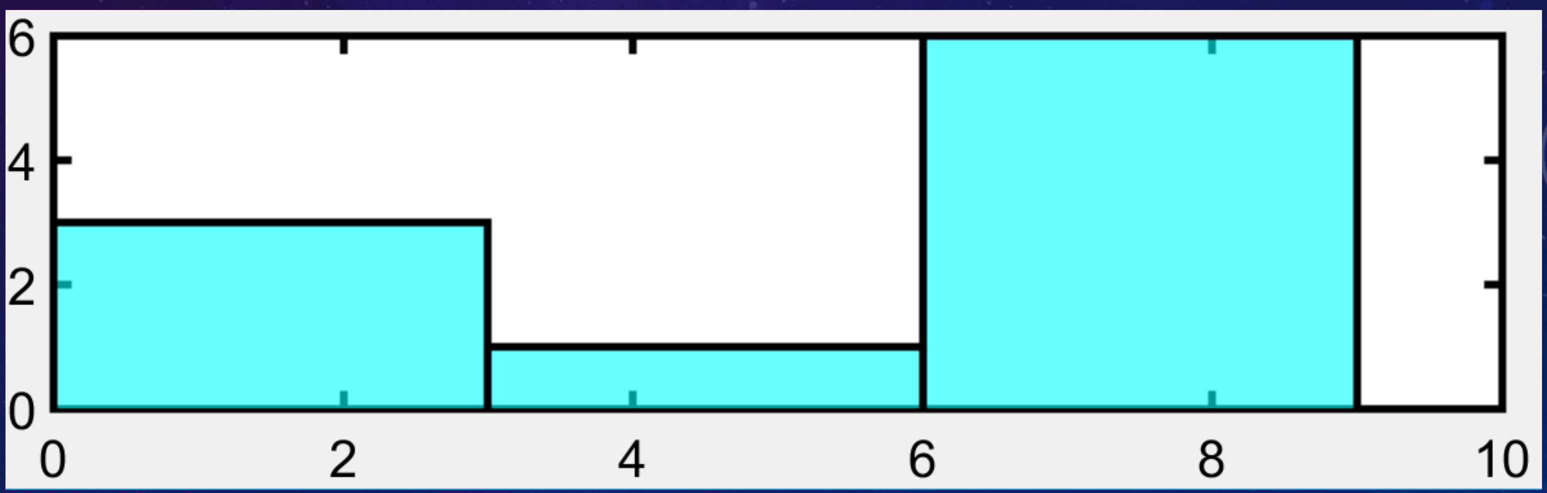
Customizing Histogram Bins

X	1	9	0	7	8	8	0	3	7	8

- MATLAB will try to pick reasonable "bins" for you, but you can also specify the bounds explicitly.

Bars for 0-2, 3-5, 6-9.

```
histogram(X, 0:3:9)
```

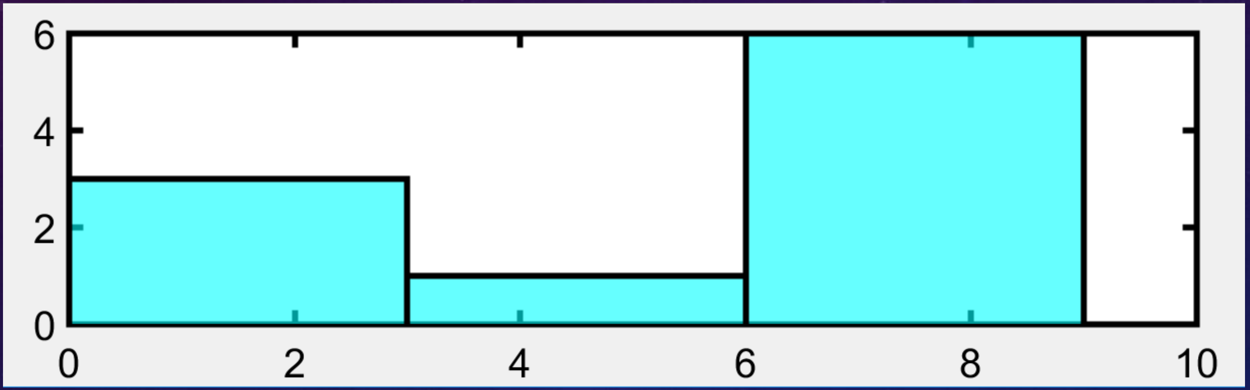


Note: There's also a `hist` function that does some of the same things, but `histogram` is generally better.

histcounts

X									
1	9	0	7	8	8	0	3	7	8

`histogram(X, 0:3:9);`



- The `histcounts` function gives you the number of elements belonging to each histogram bin.

`histcounts(X, 0:3:9);`

3	1	6

Obtaining Frequency Counts

X

1	9	0	7	8	8	0	3	7	8

- To ensure that you get a single bin for each value, first use the unique and max functions to create a vector of bins for each element.

`unique(X)`

0	1	3	7	8	9

`bins = [unique(X), max(X)+1]`

0	1	3	7	8	9	10

Need one extra for the upper bound (10) on the last bin

- Then give these bins to the histcounts function

`histcounts(X, bins);`

2	1	1	2	3	1

QUESTION: How has bins been constructed ? Start with a row vector that is the length of unique(X), and add one more element to the row vector !!

A “Working Break” Time

We'll start again in 5 minutes.

Use MATLAB “Add-Ons” which may be found in the MATLAB desktop HOME toolstrip

- **Install “Statistics and Machine Learning Toolbox”**

Variance and Standard Deviation

- Variance and standard deviation are parameters that describe statistics in a dataset.
- MATLAB has functions to calculate these:
  - `var` – variance
  - `std` – standard deviation

Sampling From Random Distributions

- MATLAB provides functions for sampling from a variety of probability distributions.
- This allows us to run simulations of random phenomena if we know the parameters of their distributions.
- For example, to simulate a uniform discrete distribution:

3	1	6	6	3

`unidrnd(6,1,5)`

Simulate rolling a 6-sided die 5 times

10	7	19	8	3

`unidrnd(20,1,5)`

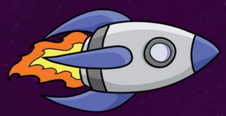
Simulate rolling a 20-sided die 5 times.

Probability is Hard. Vectorization is Easy.

- Question: If I roll a 6-sided die and a 20-sided die, what is the probability that the sum is greater than 15?
- Math Answer:
  - There's certainly an answer, but this isn't a probability class.
  - Sometimes (e.g. election example from lecture 6), where we're simulating over 200 different countries, there is no clean math answer.

Probability is Hard. Vectorization is Easy.

- Question: If I roll a 6-sided die and a 20-sided die, what is the probability that the sum is greater than 15?
- Math Answer:
  - There's certainly an answer, but this isn't a probability class.
  - Sometimes (e.g. election example from lecture 6), where we're simulating over 200 different countries, there is no clean math answer.
- **Quick ENGR101 Answer:**
  - **Simulate this a big bunch of times with MATLAB and see how often we get a sum that's greater than 15.**
  - **Because this involves "repetition", <u>vectorization</u> is our tool of choice.**



5 min

Your turn: Computational Random Simulation

- Question: If I roll a 6-sided die and a 20-sided die, what is the probability that the sum is greater than 15?
- **Use MATLAB "Add-Ons"**
  - **Install "Statistics and Machine Learning Toolbox"**
- <u>Write a script file</u> that takes these steps to find the answer:
  - Generate row vectors for Nt = 10000 rolls of each kind of die using `unidrnd(6,1,Nt)` and `unidrnd(20,1,Nt)`.
  - **Add the row vectors** together to get the sum.
  - Use **logical indexing to select sums greater than 15** and tally them.
  - Divide by Nt.
  - Probability =(number of sums greater than 15)/(number total).

Solution: Computational Random Simulation

➢ Question: If I roll a 6-sided die and a 20-sided die, what is the probability that the sum is greater than 15?

```
numTrials = 10000;

% roll the dice
rolls6 = unidrnd(6,1, numTrials);
rolls20 = unidrnd(20,1, numTrials);

% compute the sum for each roll
rollsSum = rolls6 + rolls20;

% count how many are > 15
numSuccess = sum(rollsSum > 15);

% compute probability
prob = numSuccess / numTrials
```

Other Distributions

- MATLAB supports many different random distributions.

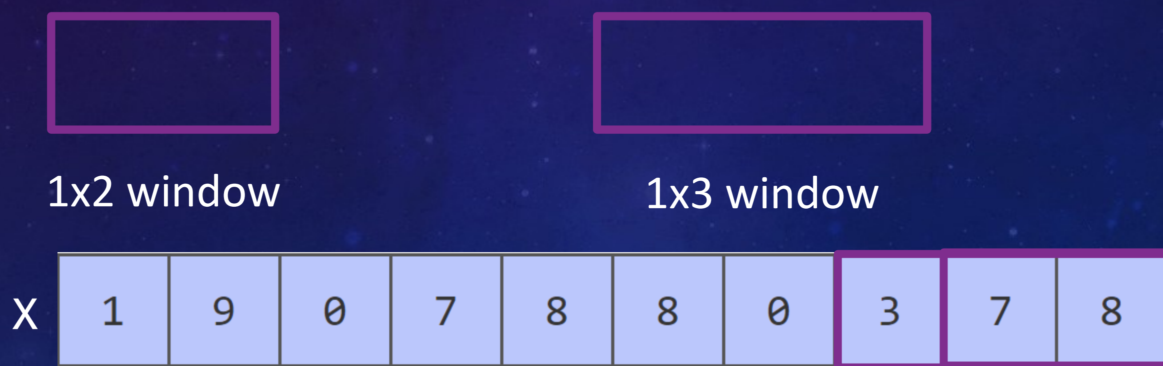
- Discrete:
  - Binomial
  - Multinomial
  - Geometric
  - Poisson
  - Etc…

- Continuous:
  - Beta
  - Gamma
  - Exponential
  - Chi-square
  - etc...

- As always, consult the documentation for more details!

Convolution

- Convolution is a mathematical operation applied to a vector or matrix that computes values based on small "neighborhoods" of elements.
- We'll only consider vectors for now.
- The neighborhoods we consider are determined by a "sliding window" that moves across a vector:



Moving Averages

- We can use convolution to compute moving averages.
  - These are useful for removing noise from measurements of a signal over time.
- Create a window vector with values that add to 1.

0.5	0.5

1x2 window

```
conv(X,ones(1,2)./2,'valid')
```

X	1	9	0	7	8	8	0	3	7	8

5	4.5	3.5	7.5	8	4	1.5	5	7.5

See you Monday