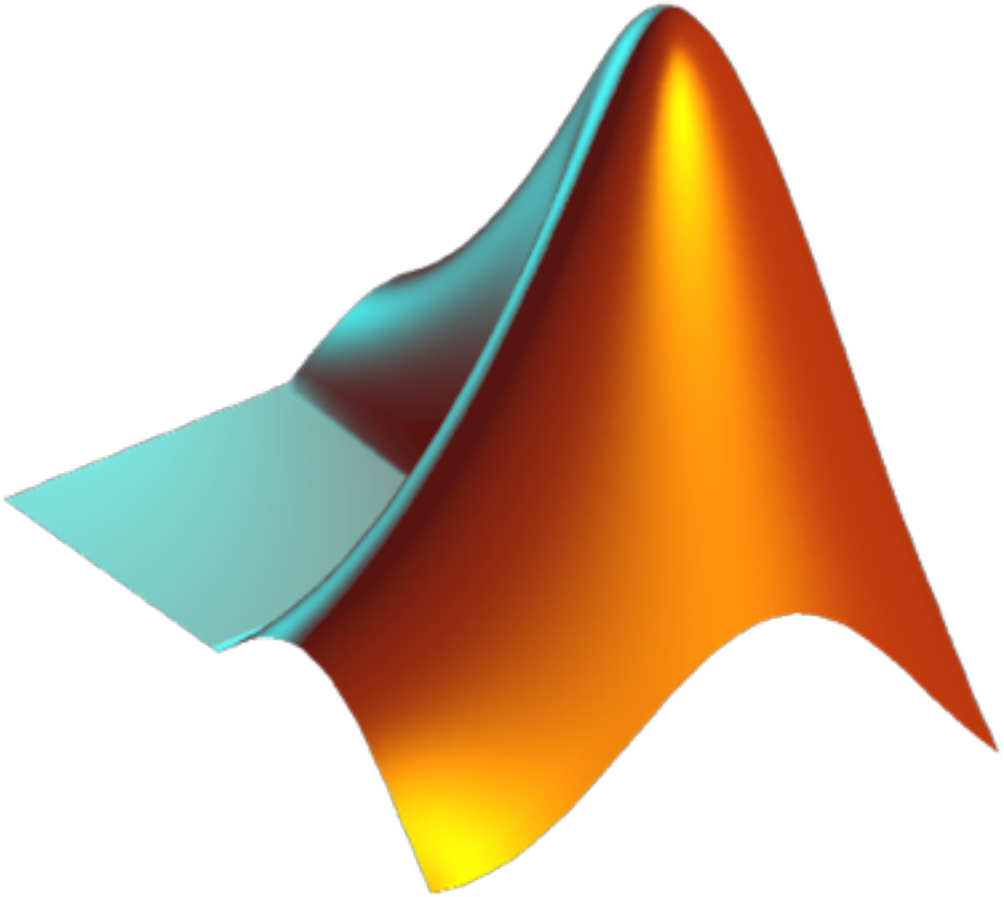
Lecture 1: Welcome to Matlab

Outline

1. What is Matlab and why should I care?
2. Administrative things
3. Getting started: Matlab basics
4. The “integrated development environment” (IDE) or Matlab desktop
5. Looking ahead

What is Matlab?



- **MAT**rix **LAB**oratory
- An general-purpose environment for doing *scientific computing*
- Allows you to acquire, process, digest, visualize, model, and communicate data
- This is done by writing code in the *MATLAB language*

Why should I care?

- Many of your colleagues are using it! Has become **the de facto standard for analyzing/visualizing data** in scientific environments
- Analyzing data via code is both **more principled** and **orders of magnitude faster** than doing things by hand or with specialized programs
- **Learn one language** for doing everything in your pipeline: processing data, running statistics, making figures, etc.
- Learning a programming language **will change the way you think** about data and data analysis

Demo:
Some examples

Course Aims

- Become proficient with programming in Matlab
- Teach you to recognize when/how Matlab can improve your workflow
- Learn how to think through implementing scientific analyses programmatically
- Develop practical data visualization skills
- Understand what resources are available for you to learn on your own

Grading Logistics

Weekly assignments consist of:

- Watching a set of video tutorials
- A programming assignment, graded on a 0, ✓, ✓+ basis:

0	Submission is broken or only a cursory attempt
✓	Submission mostly works, with minor flaws
✓+	Submitted code works and is well written

Sample solutions to the programming assignments will be posted after they are due. You can resubmit an assignment to get a ✓+

We'll assign a final project that serves as your assignments for the final two weeks of the quarter, it will be graded on the same basis.

Course grading is **satisfactory** or **no credit**. You must receive a ✓+ **on all but two** assignments to pass the course (and you must have a ✓ on those two - no zeros!)

Resources (help!)

- Your classmates
- E-mail any and all questions to the staff mailing list
- Office Hours! Held outside Peet's cafe in the Clark Center (times TBD)
- Course website: http://nens230.stanford.edu/

Accessing Matlab remotely

Let’s get started

Data Manipulation

In programming languages, we use a single equals sign to mean ‘assignment’:

```
x = 3;
```

The line above creates a new **variable**, which is named **x.** We can use this variable in subsequent calculations:

```
x + 1
x*x
```

What if we do the following?

```
x = 5;
y = x;
x = 10;
```

Then what is the value of **`y`**?

Command Line basics

List files in your current directory:	`ls`
List of variables in your workspace:	`who`
More info. on variables in the workspace:	`whos`
Clear all variables from the workspace:	`clear`
Clear the text in the command window:	`clc`
Change your current file directory:	`cd`
Open the Matlab documentation:	`doc`
Help for a specific function:	`help myfunction`

Arrays: sets of numbers

```
x = [1 3 9 77 55]
```

indexing

```
x(5)== 55
```

length

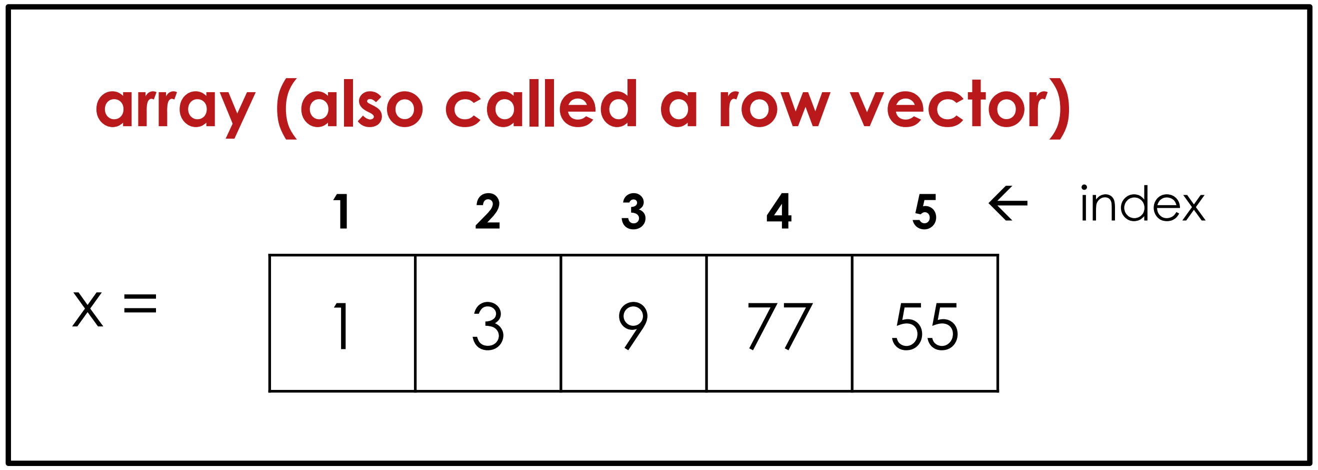
```
length(x) == 5
```

colon operator

```
x(1:3) == [1 3 9]
```

end indexing

```
x(4:end) == [77 55]
```



array (also called a row vector)

	1	2	3	4	5	← index
x =	1	3	9	77	55	

`length()` is a **function**

*The object in parentheses is called the **argument** of the function*

<- Ways to extract parts of an array: “index into an array”

How to “build” an array

colon operator

```
x = 1:10
x = [1 2 3 4 5 6 7 8 9 10]
```

returns values from 1 to 10, incrementing by 1

`linspace()`

```
x = linspace(0,1);
```

returns 100 (default) equally spaced values from 0 to 1

```
x = linspace(0,1,6)
x = [0 0.2 0.4 0.6 0.8 1]
```

returns 10 equally spaced values from 0 to 1

Some functions (like `linspace`)
can have a variable number of arguments!

Demo: Arrays and Functions

Looking Ahead

First 2/3 of the course:

- Data wrangling: how to load data into Matlab and get it into a useful format; variables and data types
- Basic programming: control flow, logic, and manipulation
- Data visualization: 2D and 3D plots and charts, customizing figures

Last 1/3 of the course (flexible):

- Specific algorithms/tools for analyzing data
- Signal and image processing, modeling and simulation, statistics, etc.