Lecture 1: Introduction

Math 98

math.berkeley.edu/~tom_schang

Introduction

Instructor: Tom Schang
Email: tom_schang@berkeley.edu
Office hours: ~~Tu/Th 3-5 PM, 1041 Evans Hall~~
Website: math.berkeley.edu/ tom_schang/my-website/teaching

- Per Section 102.23: Course Materials of the UC Berkeley Student Code of Conduct - **don't post any of these materials to CourseHero (or any similar sites)**

Lab computer login information can be found on bCourses

Lectures, Grading and Deliverables

1. 5 Lectures
   - Every Tu/Th, starting Tuesday (09/03)
2. 4 Homeworks
   - Graded largely on effort and completion
   - Assignment Submission: on bCourses
     - ★ Solutions appear 24 hours after deadline
   - You are allowed 1 late day per assignment, 3 late days total
     - ★ In your best interest to complete course quickly
3. 1 Project
   - Graded for accuracy
   - Expectation is that you get this *completely* correct as warmup for 128A assignments
4. Receiving a P in this class should not be hard
   - Each HW worth 15 points, Project worth 40
   - Must receive minimum of 70% average on the HW assignments and 70% on the project
   - Start early, work hard, and seek help on the assignments (come to office hours!!)

Who should take this course

1. If you have no/minimal programming experience:
   - This is the target audience of this course
   - We will build things up from scratch
   - I also expect many of you to also feel a bit nervous about programming (and that's OK!)
2. If you have extensive programming experience in another language:
   - This course likely won't be a good use of your time
   - Easy to pick up MATLAB syntax yourself. Work through an online tutorial (see suggestions on my webpage)
3. For everyone else in between.....
   - Take a look at some of the homeworks and see if you could mostly code them up in another language

To everyone: You may find it more efficient to learn yourself once you've gotten started. Everyone is highly encouraged to read/study ahead.

Why you should take this course

Why take this course? (vs. some other course)

1. Condensed focus that covers the necessary material for Math 128a
2. Assignments that are well aligned with Math 128a
   - Especially the final project
3. Availability of customized help (from me!) in office hours
   - Even if you can figure out the problem, it is extremely useful to get feedback on your thought process and areas for improvement. So make sure to ask for it!

There is no real need to formally enroll for 1 unit (unless you want it). I will be happy to grant access to the course material to auditors and speak with them in office hours.

Course Plan

Plan: Spend 50-60 minutes lecturing (depends on the day) and do lots of exercises

- Have MATLAB open to experiment and try things out
- Will stop for many in class exercises, and I will go around and offer help

Course Expectations

This is a fairly intense course. In 3 weeks you will go from no programming experience to being able to complete a 128A Programming Assignment.

What you should do:

- Try to set aside a block of ∼2-3 hours at least every other day to practice programming (including the lectures)
  - Like taking a foreign language class. Immersion
- Interrupt me as frequently as needed for clarification and questions
- Talk to neighbors (at a respectful volume) and ask for help
- **Important:** Actually try to figure things out and stay to work on the exercises
- **More Important:** Practice. The best way to learn how to code is to **write lots of code.**

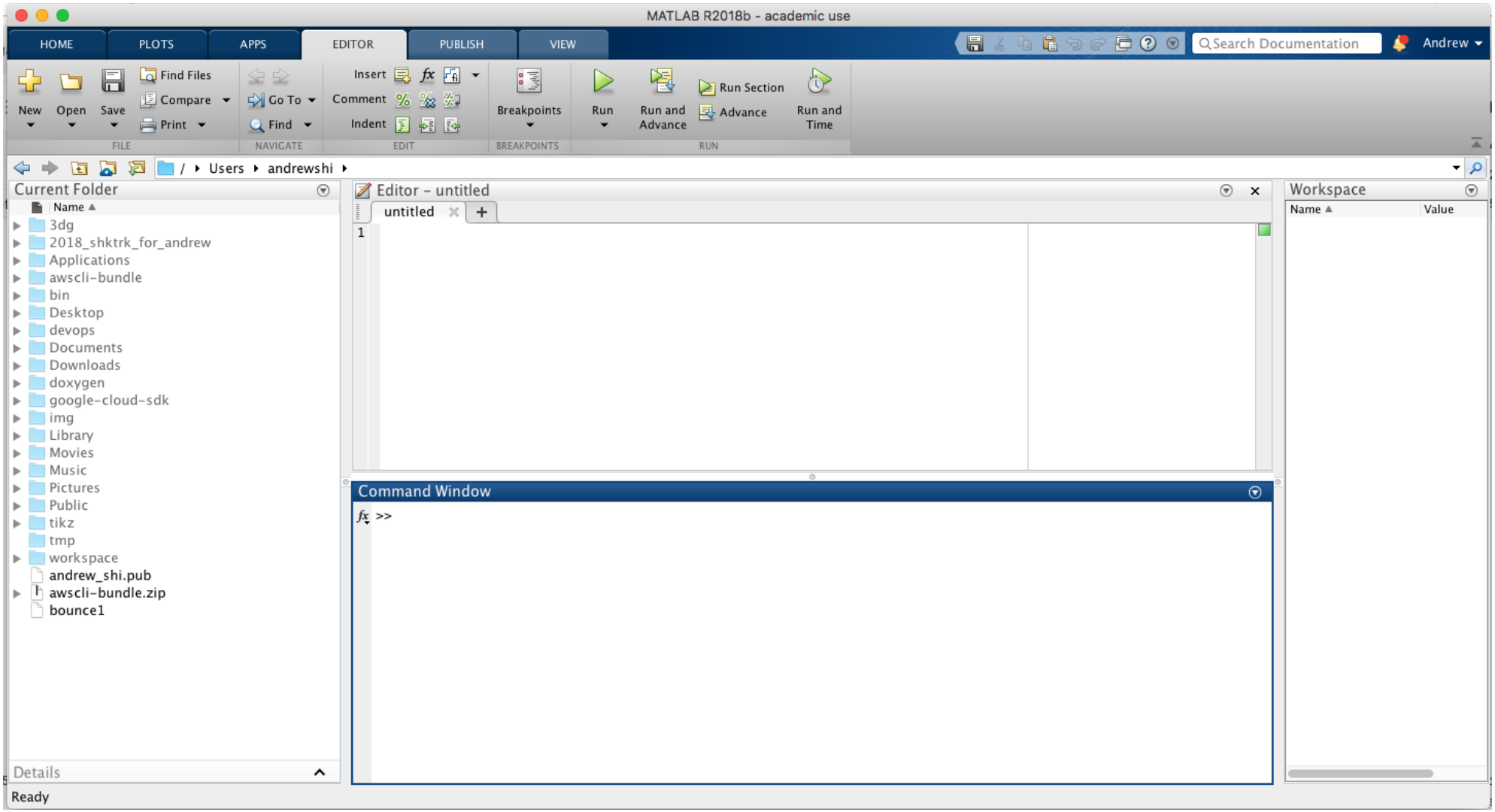
Agenda

- Go over items on the course webpage
- MATLAB Introduction
- Variables and Formatting
- Vectors and Matrices
- Relations
- Scripts
- Exercises

Downloading MATLAB

- MATLAB is available for free for UC Berkeley students via a campus license
- Make sure to download ASAP in case issues arise

MATLAB Intro: Opening up MATLAB



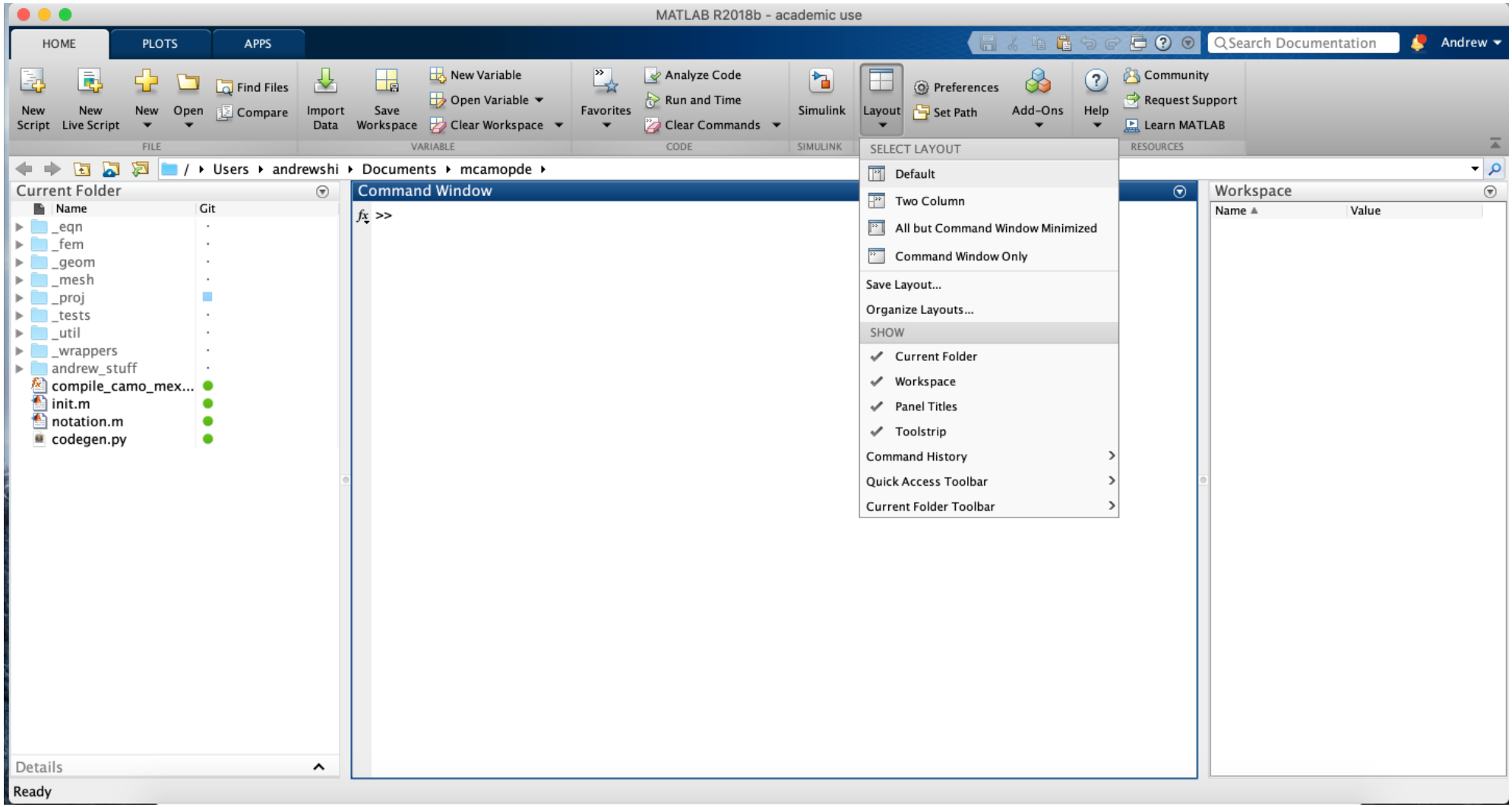
MATLAB Intro: Getting started with MATLAB

Matlab has five features:

1. **Current Folder** shows the files that MATLAB is accessing. By default MATLAB cannot execute files contained in other folders.
2. **Command Window** Here we can define variables, perform calculations, and much more.
3. **Workspace** is a MAT-file containing the variables you have defined in your session.
4. **Editor** allows us to save collections of commands as M-files.
5. **Command History** can be accessed using the up arrow.

It's OK if this doesn't make much sense yet.....we will revisit this.

MATLAB Intro: Reset Layout



If something happens to this, you can reset the layout to the default configuration.

MATLAB Intro: Command Window

To begin with, think of MATLAB as a giant calculator.

Operations: $+, -, *, /, \hat{\cdot}, \exp(\cdot), \sqrt{\cdot}, \log(\cdot)$

```
>> 2 - 3*(1+2)/2
ans =
    -2.5000
>> 3^4
ans =
    81
>> log(4)
ans =
    1.3863
>> sqrt(9.01)
ans =
    3.0017
```

MATLAB Intro: Help

Wait, what kind of logarithm is that?

```
  >> help log
log    Natural logarithm.
   log(X) is the natural logarithm of the elements of X.
   Complex results are produced if X is not positive.

   See also log1p, log2, log10, exp, logm, reallog.

   Reference page for log
   Other functions named log
```

help is going to be your best friend when using MATLAB.

MATLAB Intro: MATLAB Documentation

- The online documentation is also excellent with many examples.
- Google search "matlab log"
  - https://www.mathworks.com/help/matlab/ref/log.html
- Knowing where to find the answer is much more important than knowing the answer.

MATLAB Intro: Built-in MATLAB Functions

- Many built in functions in MATLAB to do math.

`cos(x)`	Cosine	`abs(x)`	Absolute value
`sin(x)`	Sine	`sign(x)`	Signum function
`tan(x)`	Tangent	`max(x)`	Maximum value
`acos(x)`	Arc cosine	`min(x)`	Minimum value
`asin(x)`	Arc sine	`ceil(x)`	Round towards $+\infty$
`atan(x)`	Arc tangent	`floor(x)`	Round towards $-\infty$
`exp(x)`	Exponential	`round(x)`	Round to nearest integer
`sqrt(x)`	Square root	`rem(x)`	Remainder after division
`log(x)`	Natural logarithm	`angle(x)`	Phase angle
`log10(x)`	Common logarithm	`conj(x)`	Complex conjugate

Variables: ans

The workspace shows variables that have been defined in the current session. In particular, ans is by default the value of the last arithmetic computation we made. We can check the value of a variable by entering its name in the command window.

```
>> 1 + 3
ans =
     4
>> ans
ans =
     4
>> 3 * 8
ans =
     24
>> ans
ans =
     24
```

Variables: Defining Your Own

We can define our own variables, too! Variable names must start with a letter and can contain letters, digits, and underscores. MATLAB is case sensitive but all built-in MATLAB functions use lowercase letters, so if you use at least one capital letter in your variable names you can be sure to avoid any name conflicts.

```
>> x1 = 5.337
>> my_variable = ‘‘howdy"
>> frodoBaggins33 = sqrt(2)*pi
>> x2 = x1 + 1
```

Variables: Built-in Variables I

There are many built in variables too.

```
>> pi
ans =
    3.1416
>> eps
ans =
    2.2204e-16
```

You can override these if you want....

```
>> pi = 4
>> pi
pi =
    4
```

.....but this is usually a bad idea. Try to avoid ambiguity.

Variables: Built-in Variables II

There are two more special built in variables: `Inf` and `NaN` (not a number)

```
>> 1/0
ans =
    Inf
>> -1/0
ans =
    -Inf
>> 0/0
ans =
    NaN
>> Inf/Inf
ans =
    NaN
```

What is `Inf/0`? What about `exp(exp(7))`? Compare to `exp(exp(6))`.

Variables: Using Informative Names

Give your variables informative names. Good.

```
>> radius = 4
>> area = pi*radius^2
area =
    50.2655
```

Bad.

```
>> foo = 4
>> blah = pi*foo^2
blah =
    50.2655
```

Variables: Using Informative Names

Use a semicolon to suppress the output of a command. Using `disp` will suppress the `ans =` text, but will also not save the output to `ans`. Multiple commands can be placed on a line separated by semicolons.

```
>> x = 5; y = 6; disp(x+y)
11
```

Use SHIFT-ENTER to start a new line. Use ellipses (...) and SHIFT-ENTER to continue a line.

```
>> sqrt(5 + 7 + ...
13)
ans =
    5
```

Variables: Clearing

Use `clear` to clear all variables from the workspace. Use `clear VAR1 VAR2` to clear specific variables `VAR1` and `VAR2`.

```
>> x = 5; clear x;
>> x
Undefined function or variable 'x'.
```

Use `clc` to clear the command line.

Formatting: `long` and `short`

Doesn't `pi` have more digits than this?

```
>> pi
ans =
    3.1416
```

This is just a formatting/visual thing. Try format long to show 15 digits.

```
>> format long
pi
ans =
    3.141592653589793
```

Then try format short.

```
>> format short
pi
```

Formatting: `compact` and `loose`

Go into MATLAB and do stuff in the command window. Do you notice all those blank lines in between the command window output?
Experiment with `format compact` and `format loose`.

Vectors: Introduction

In addition to strings and scalars, we can define variables that represent vectors.

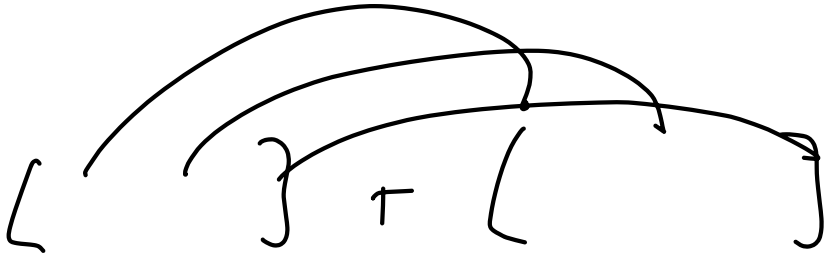
```
>> v = [1, 5, -2];
>> y = [2; 3; 4];
>> z = [-1 10 4 2];
>> max(v)
ans =
     5
>> length(y)
ans =
     4
```

$v + y$

$$\begin{bmatrix} 0 & 5 & -2 \end{bmatrix} \quad \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}$$

$$\begin{bmatrix} 3 & 7 & 0 \\ 4 & 8 & 1 \\ 5 & 9 & 2 \end{bmatrix}$$

$v + v$



$v + 2$