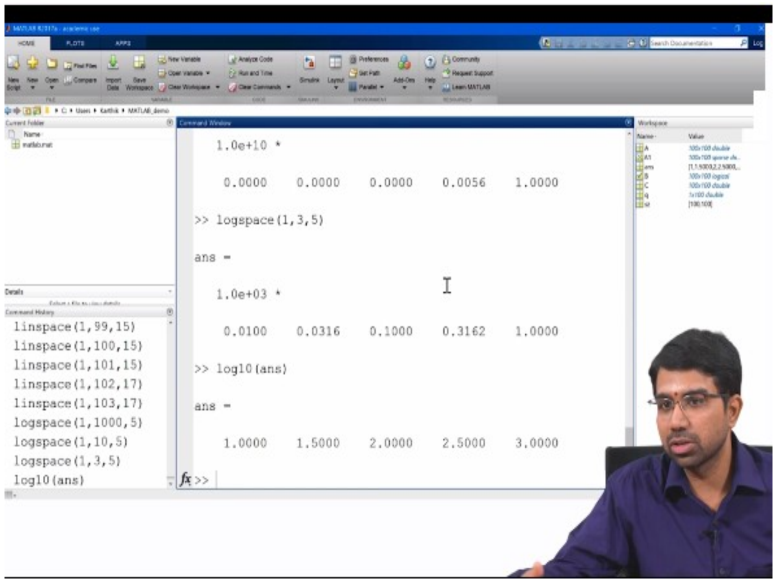
Computational Systems Biology

Karthik Raman
Department of Biotechnology
Indian Institute of Technology – Madras

Lecture – 09
Lab: MATLAB Basics

So, welcome to this lab video, so these videos are going to be more informal where I'm going to be sitting, chatting with my students, discussing various problems and so on, although in the first couple of videos, I will be giving more of an introduction but you will get a feel for, how one codes, how one make mistakes while coding, how do we correct those mistakes, how do we profile, how to we debug and so on.

(Refer Slide Time: 00:40)



```
   1.0e+10 *

    0.0000    0.0000    0.0000    0.0056    1.0000

>> logspace(1,3,5)

ans =

   1.0e+03 *

    0.0100    0.0316    0.1000    0.3162    1.0000

>> log10(ans)

ans =

    1.0000    1.5000    2.0000    2.5000    3.0000

>>
```

```
linspace(1,99,15)
linspace(1,100,15)
linspace(1,101,15)
linspace(1,102,17)
linspace(1,103,17)
logspace(1,1000,5)
logspace(1,10,5)
logspace(1,3,5)
log10(ans)
```

So, in this video, we will focus on some of the basics of MATLAB, so we will do a quick MATLAB overview. We will just go over some basic things, okay. I think you are familiar with many of these concepts but let's just skim through all of these again. So, there are a bunch of basic MATLAB commands that you need to know,

clc- clear screen

clear- removes all the variables. You would have seen that the workspace becomes empty once you type clear.

clear all-removes all variables, functions and everything from the memory, anything that you have defined.

ls-lists whatever is in the current directory.

You can also use various shell commands. One useful thing I often use is, if you are on windows especially is to use

!explorer . - It will open windows folder in the current directory. It will show the directory, you can copy files there, you can add files there and so on.

pwd-shows the current working directory.

These are all almost basic commands,

help-gives you help for a given command and you may often want to use 'doc' because for example, even if you click something in, you can say

doc ls- will open the MATLAB help browser with the details of the command 'ls'. It will give you examples, it's hyperlinked, it's nicely formatted and you can understand a lot more by looking at the documentation, so that's using doc.

And if you want to search for a particular thing, suppose you want to figure out how to create a sparse matrix, you just say

lookfor sparse- It will return a bunch of functions which have sparse mentioned somewhere in the name. So, you will see that there are a bunch of so in this case, I get mostly a lot of examples from different toolboxes actually including the COBRA toolbox and so. We can clear it.

Let's say, we want to create a matrix of zeros, so you can say

A = zeros(10)

and then if you say,

who- will say your variables are whatever and

whos -will give you more details. It will tell you what is the size of the A matrix and how much space is being taken up and so on.

So, let's say,

A = zeros(100) so this will create a 100x100 matrix of zeros, so you see it is taking up around how much space? 10,000 elements. 8 bytes per element, 80,000.

Now if I say,

A1 = sparse(A), A1 takes just 824 bytes. So, that also sounds like a lot, it basically stores the entire same matrix in about 1/100th of the space. We will look at sparse in a moment, so let us look at some of the other basic commands. So let's just save it, it will save everything to MATLAB.mat. You can also use, whos -file filename.mat. So, if you now say

whos -file MATLAB.mat- it will tell you what are all the variables present in that file.

These are basic useful functions that one needs to know. Without these your everyday MATLAB cannot run. Let's say, sz = size(A). Now,

class(sz)- It just says double

size(sz)- returns 1 2 , as it is actually a 1 x 2 matrix.

length(sz)- will return the longest dimension which is 2.

So, make sure you understand what it returns, it is equivalent to max size x for non-empty arrays and zero for empty ones.

numel(sz)- tells you the number of elements which in this case will be 10000 because we created a 100 x 100 matrix a little while ago.

Transpose the matrix, so let's create a random matrix B = rand (10).

I will just do a small trick here. B = rand (100) > 0.9,

spy(B) shows you the sparsity pattern. This is how the matrix looks and if you now do,

spy (B+B'), you will be able to see this is actually symmetric.

There's a small trick I will use,

q = symrcm(B+B') (Note: symrcm returns the symmetric reverse Cuthill-McKee ordering. Refer MATLAB help for more details)

C = B+B'

spy(C(q,q))-you will actually see that now this looks really symmetric, this is actually a matrix reordering thing, which is very useful for several kinds of algorithms.

But essentially, you will be able to see that this is now a symmetric matrix, **"Student conversation"** What is this? This is just a, this just tells you what are the nonzero elements in the matrix, that is all.

imagesc(C(q,q)) (Note: imagesc function scales image data to the full range of the current colormap and displays the image. Refer MATLAB help for further information)

We are getting a little ahead of ourselves, right and you can also say,

colorbar - displays a vertical colorbar to the right of the current axes or chart. Colorbars display the current colormap and indicate the mapping of data values into the colormap. You can have different built-in colour maps.

colormap jet- Most usual one is jet. I think that's been changed in the more recent MATLAB versions. Something called parula,
colormap parula- this is the default colour map.

So, essentially I did all this to show you how you can transpose a matrix and get a symmetric matrix, right.
A = A' gives you the transpose of A.
So more other commonly useful things include,
1:7:100 returns every seventh number.
And if you want something say,
linspace(1,99,15)- 15 equally spaced numbers between 1 and 99. You will observe the same result as 1:7:100, right?
linspace(1,100,15)- This is different though, so if you put 100, you will get equally spaced numbers. Similarly for, linspace(1,101,15), linspace(1,102,17) and so on.

You can also have
logspace()- Gives you points that are log spaced, say logpace(1,3,5) and you get an answer, ans. So, if you actually do log10(ans), you can see they are equally spaced numbers between 1 and 3. This can actually be useful especially when you are doing parameter estimation and so on. You may want to do a log sampling of the parameter space because whenever, you need to sample across orders of magnitude, you would need to resort to log sampling.
(Refer Slide Time: 12:06)

Recap

Topics covered

- Basics of MATLAB

In the next video ...

- Indexing

So, in this first MATLAB lab video, we looked at some of the basics of MATLAB but in the next video, we will go on to some very interesting and in my opinion one of the most important advantages of using MATLAB for computation, which is the various ways one can index data in MATLAB.