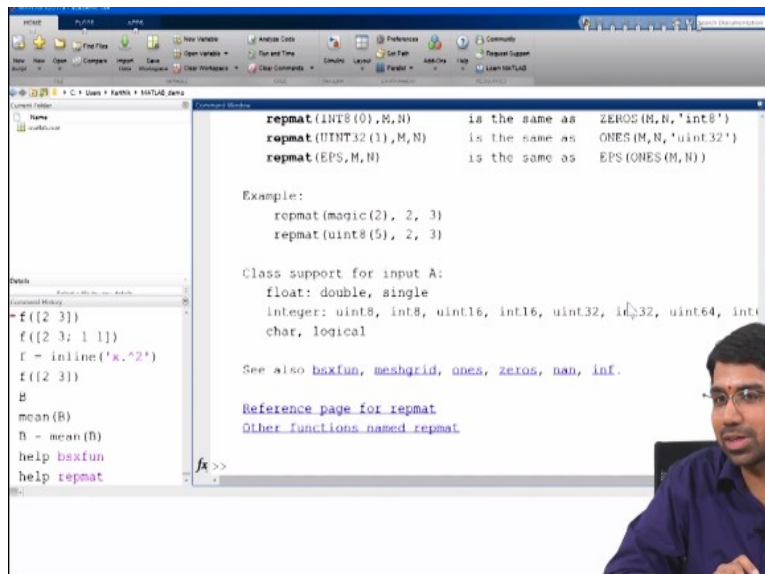


**Computational Systems Biology**  
**Karthik Raman**  
**Department of Biotechnology**  
**Indian Institute of Technology – Madras**

**Lecture – 11**  
**Lab: MATLAB Basics**

So, in today's lab video, will look at matrices and functions in MATLAB, what kind of special matrices are available in MATLAB and how one writes functions, passes multiple arguments, returns multiple values and so on.

**(Refer Slide Time: 00:27)**



Yeah, so find A is another very important function but it is often abused, you shouldn't be using it in a lot of cases, you should instead just use logical indexing, right, so one way to do is, let's say B is this, now I want to change every element that is -5 back to 0, right, so I would first do  $B(\text{find}(B == -5)) = 0$

but I might as well have just done

$B(B == -5) = 0$

This will no longer work because there are no more -5s but so, in fact if you type this code out in the MATLAB editor, it will underline it and tell you that's possible improvement you can make.

Because find is a slower function, so it is the classic suggestion that MATLAB will give, please consider using logical indexing instead of find, so

whos i,

i does not exist, right,

If you type i enter, it becomes, now if I say whos i, similarly j, right, so avoid using, in general, avoid using i and j in loops and so on, so you say  $x = i + 1$  automatically, i and j are automatically complex.

If you said  $i = 0$  and then say  $x = i + 1$ , then things are sensible. But generally avoid it, you don't want to play around with i and j, when it comes to MATLAB, use other loop indices. And you can make inline functions, so it's like those of you familiar with python, it is like lambda functions in python but this is just, so you can just say,

`f = inline("x^2")`, then `f(3)` will give you 9 as the answer.

Now, `f([2 3])`, will this work?

No, because x is always a matrix in MATLAB, so here we are saying x squared, it has to be matrix multipliable, which means it has to have equal number of rows. It should be a square matrix, right

So, if you give `f([2 3; 1 1])`, it will work.

But this may not be what you want, if you want element wise squaring, you use something called the dot operator, you use

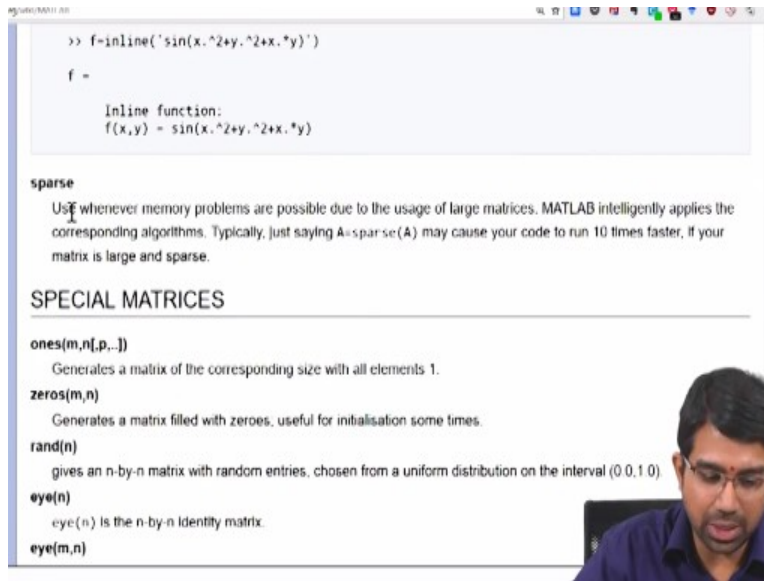
`f = inline("x.^2")` (x dot raised to 2)

And now if you say `f([2 3])`, it will work, so dot is quite important. So, these are all the small tricks, the use of the dot operator, use of logical indexing and there is also a very important function called "repmat", to replicate a matrix in a particular shape.

So, let's go back to this example again, B, okay, so this is the mean, what if you wanted to subtract the mean of every row, every column from the matrix to you know, do some sort of normalisation? No that won't give you, so how do you, I think now it actually works in MATLAB, so you can do `B - mean(B)`, it automatically works, it wouldn't work a few years ago in a previous, you have to use something known as either "bsxfun" or "repmat".

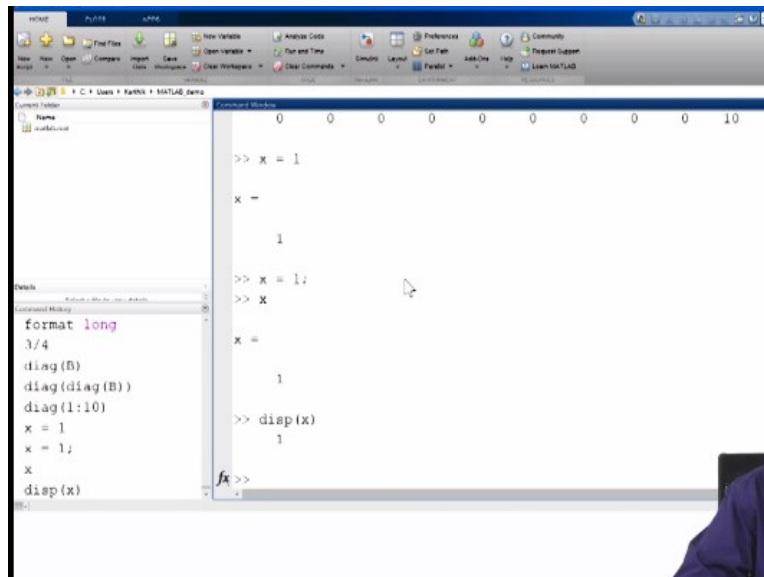
So, you should check these out. These are the tricks to make MATLAB code much faster. You may have actually written a loop to loop through all the columns, find the mean, subtract the mean from the column that kind of thing, instead you can just do it in one shot.

(Refer Slide Time: 05:45)



So, sparse is very important, so you need to use it a lot of times, there are some tools which actually requires sparse matrices only. We will find that MATLAB BGL-boost graph library that we will use for network analysis, accepts only sparse matrices as input for any of the functions.

(Refer Slide Time: 06:07)



“**Professor – student conversation**” No, so that is true but sparse matrix is basically a different data structure as far as MATLAB is concerned. So, that will make your codes run more efficient as long as the matrix is really sparse. You take a dense matrix and make it sparse, your code will run worse but if you take a sparse matrix and you make it sparse your code will run much faster.

Then there are a bunch of special matrices that you need to know.

`ones(3);`

`ones(3,2);`

`zeros(4);`

`zeros(3,2);`

`eye(5);`

`eye(4)`- it is an identity matrix, eye, right and you can also have

`eye(3,4)`- it is an identity matrix with more zeros, and of course

`rand(4);`

`format;`

`rand(4);` Now the 'format' is useful to change how the format looks, just `format`, resets the format. But you can say

`format rat;` which will keep things as a rational and just "format" will keep them as small floats and `format long` and so on.

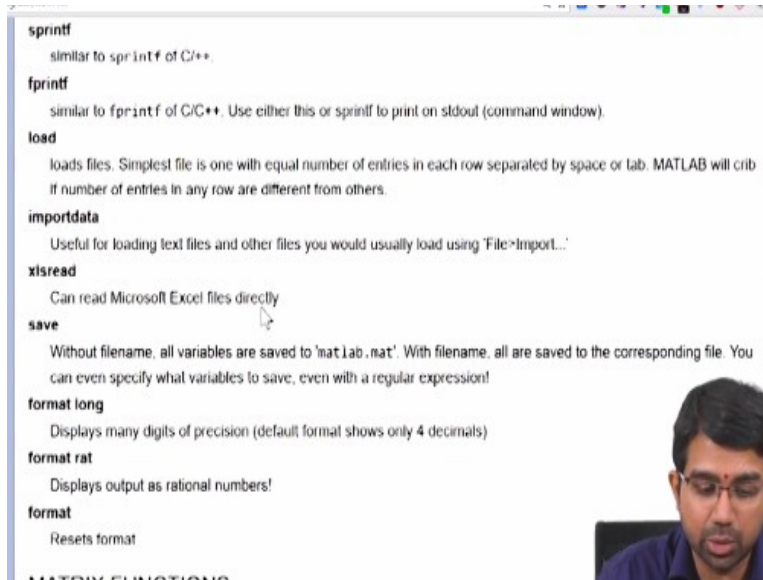
`diag(B)` - you can get the diagonal of a matrix like this

`diag(diag(B))`-this is how you create a diagonal matrix. `Diag` of a vector would give you a diagonal matrix.

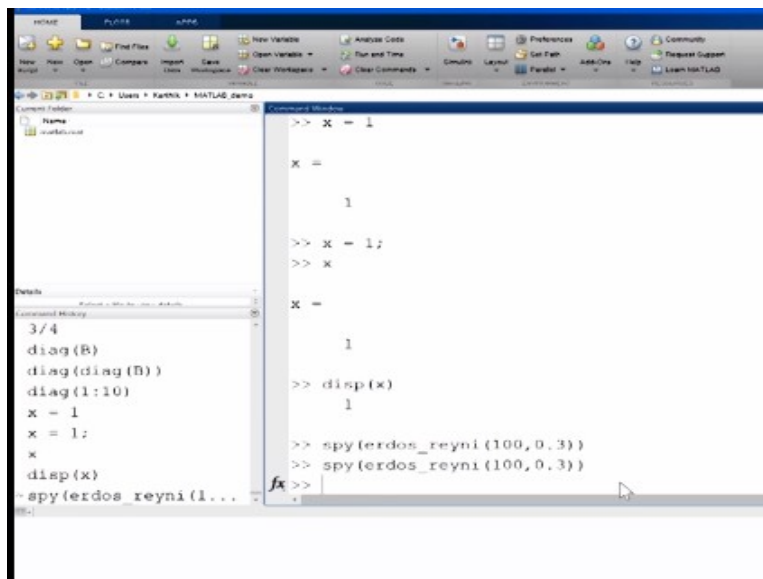
Of course, basic things like ending the command with a semicolon say `x=1;` suppresses output, right and if you say `x`, it will give you this big like 1 new line, `x = one` more new line, 1, one more new line.

So, you can say just `disp(x)` and it gives you much more compact representation.

**(Refer Slide Time: 08:50)**



And you have a sprintf and fprintf which are similar to printf in C. Loading files, you can import data, you can even read xls, you can save files, different types of formatting. “**Professor – student conversation**” Cobra tool box is for constraint-based modelling, it is for network analysis, so we will need BGL immediately. **(Refer Slide Time: 09:21)**



So, BGL will give you things like, `spy(erdos_reyni(100,0.3))` - this is an Erdos\_reyni matrix, this is a random matrix with roughly 0.3% of the edges, so you see there are 2994 roughly, 3000. We will look at what these models are tomorrow.

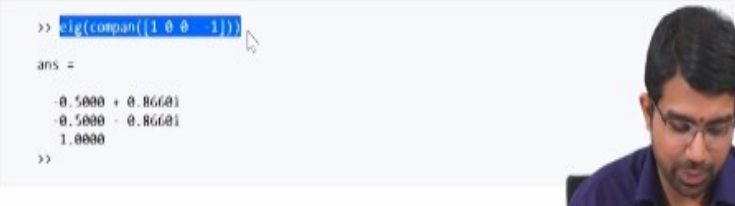
**(Refer Slide Time: 10:07)**

Cholesky Decomposition

## POLYNOMIAL FUNCTIONS

**polyval(P,X)**  
 returns the value of a polynomial P evaluated at X. P is a vector of length N+1 whose elements are the coefficients of the polynomial in descending powers.  $Y = P(1) * X^N + P(2) * X^{N-1} + \dots + P(N) * X + P(N+1)$ . If X is a matrix or vector, the polynomial is evaluated at all points in X.

**compan**  
 Can be used for solving polynomial equations. Beautiful function! The eigenvalues of the companion matrix of a polynomial are the roots of the polynomial!  
 Example. Calculate the cube roots of unity, i.e. the roots of the equation  $x^3 - 1 = 0$ .



```

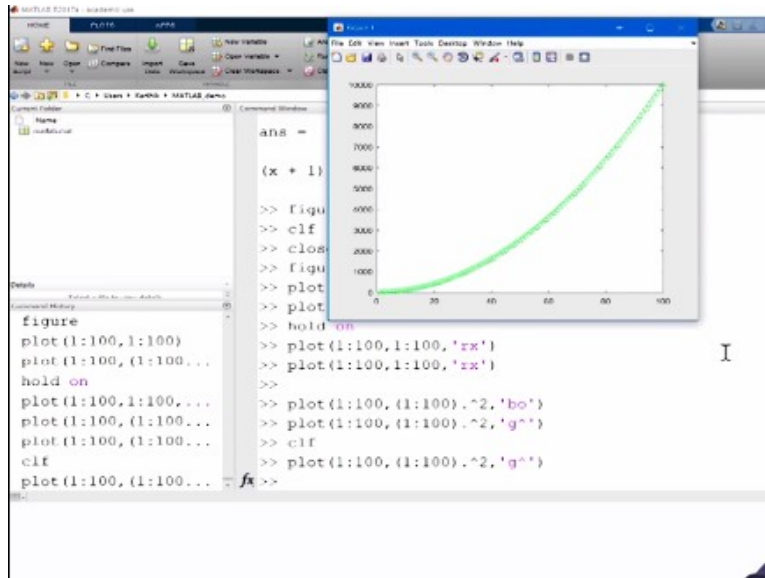
>> eig(compan([1 0 0 -1]))
ans =
    0.5000 + 0.8660i
    0.5000 - 0.8660i
    1.0000
>>
  
```

You can invert a matrix, you can also use this what's known as a divide in MATLAB, right, so it's a special matrix divide, so try to look at this as b over a, right, just like  $ax = b$ ,  $x = b/a$ .

So, here if you have  $Ax = b$  as an equation, you have x as b 'over' A, difference is, this is usually used, so instead of actually computing the inverse, this might be solved more efficiently,  $x = b/A$ . Yeah, I think this solves the equation without computing the inverse.

Then, matrix multiplication, element wise multiplication and dot does element wise operations. Norm, you can compute various norms of a matrix, then expm, as we did discuss earlier, it gives you this and you have different norms, you can have 1 norm, 2 norm, infinite norm so on and so forth. You know eigenvalues, Cholesky decomposition and so on, this is for polynomial function, I think I will skip some of these.

**(Refer Slide Time: 12:02)**



But I do like this command, so you can do  
`eig(compan([1 0 0 -1]));` which gives

```

ans =
-0.5000 + 0.8660i
-0.5000 - 0.8660i
1.0000 + 0.0000i

```

I don't know if anybody recognises these numbers. “**Professor – student conversation**” Cube roots of unity, right? And this is basically done from a matrix, eigenvalues of the companion matrix of this polynomial. This is a polynomial  $x^3+0x^2+0x-1=0$  and if you produce the companion matrix for that, it is that matrix for which the characteristic equation is the polynomial, so  $\lambda^3 = 1$ , so like the simplest matrix, it is called actually a companion matrix. So, companion matrix of polynomial with, I don't know this is not a very helpful, help command. So, basically solving any polynomial is just the same as computing the eigenvalues of the companion matrix, it is quite simple.

Then, one very useful thing in MATLAB is what's known as a cell. So, you basically say `a = {1}`, it is now a cell and the good part is you can say `a={1; 'sysbio'}`, you have different kinds of things mixed and matched within cells.

And you can say  $a\{3\} = [1\ 1; 2\ 1]$ ; you can just mix all kinds of things within a cell. And some very useful MATLAB functions are,

`cellfun(@length, a)`- for elemental cell, run function given

`ans =`

1

6

2

So, you have `cellfun` and different kinds of such useful things. And you also have symbolic, it is completely symbolic, so it is, again, algebraically calculate, you can have cancellations and things like that. We will actually see this is useful when we look at constraint-based modelling. Because you want to usually compute this matrix, a product of a stoichiometric matrix and a bunch of vectors.

Solving differential equations is quite tough. I am not too sure. And then you have a bunch of things for working with figures;

`Clf`- clears the figure ;

Most useful command is `close all`- you close all the figures that are currently open and because invariably you will have a loop where in you keep generating multiple figures and so on.

And so, let's say

`figure`

`plot(1:100, 1:100)`- you now have a line and now

`plot(1:100, (1:100).^2)`- you have this but the old line is gone, so instead you say,

`hold on`

`plot(1:100, 1:100, 'rx')`-so this line of course, is really here, whereas

`plot(1:100, (1:100).^2, 'bo')`- 'bo' will give you blue circles, so you can learn those shorthands, 'rx' will give you red crosses, okay.

**(Refer Slide Time: 18:10)**



Sparsity plot of a matrix — puts blue dots wherever entries are non-zero. Shows number of non-zeros as well.

## PROGRAMMING

- function name and file name should be the same
- % comments
- help and comments
  - help myfun will return the first block of comments in myfun.m (see the following example as well)
- return is implicit. For example

```

% my function to calculate max of two numbers
%
function a=mymax(x,y)
if (x>y)
    a=x;
else
    a=y;
end
return %optional: whatever a is at the end of the
%function file, that value will be returned.

```

- programs vs scripts
  - Scripts don't have a function keyword. They're just 'scripts', which run a series of commands.

So, there are subplots, waitforbuttonpress, ezplots, then this is important title, x label, y label, colour map and sparsity. I have shown you all this already. This is how you write a function, function a is you know, so this is very different from any other languages, what is the, okay let's do this exercise.

**(Refer Slide Time: 18:47)**

```

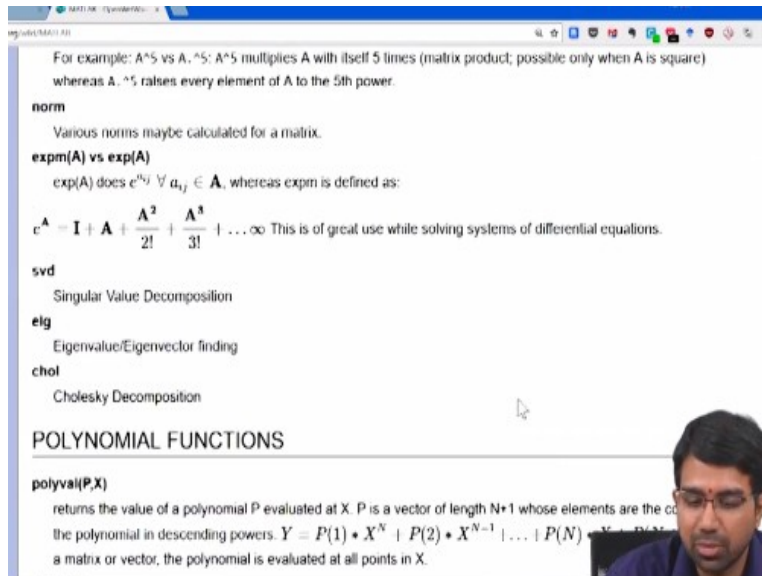
1 % my function to calculate max of two numbers
2
3 function a=mymax(x,y)
4     if (x>y)
5         a=x;
6     else
7         a=y;
8     end
9     return %optional: whatever a is at the end of the
10          %function file, that value will be returned.

```

Okay, so now if I say, mymax ([3,5]), will this work? It did not because I deliberately had a typo in the filename, so the functions are addressed by filename not what you call them here, right, so I could still have this abc, save it as mymax.m and then say mymax(3,5). You should have suppressed this and if I mouse over this, function name abc is known to MATLAB by its

filename mymax, fix, so the correct way to write the code of course is this but you should just be aware, so you can have this weird kinds of errors, okay.

**(Refer Slide Time: 20:31)**



The image shows a MATLAB help window with the following content:

For example:  $A^5$  vs  $A.^5$ :  $A^5$  multiplies  $A$  with itself 5 times (matrix product; possible only when  $A$  is square) whereas  $A.^5$  raises every element of  $A$  to the 5th power.

**norm**  
Various norms may be calculated for a matrix.

**expm(A) vs exp(A)**  
 $\exp(A)$  does  $e^{a_{ij}}$   $\forall a_{ij} \in A$ , whereas  $\expm$  is defined as:  
$$e^A = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \dots \infty$$
 This is of great use while solving systems of differential equations.

**svd**  
Singular Value Decomposition

**eig**  
Eigenvalue/Eigenvector finding

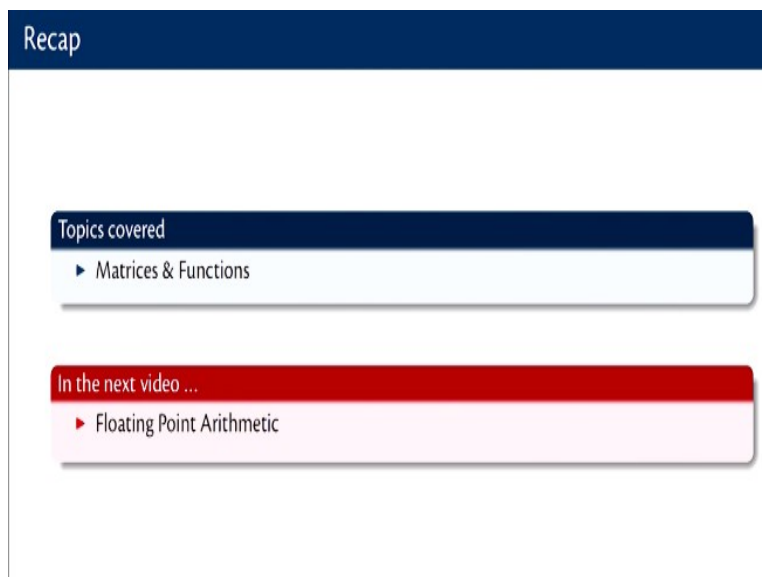
**chol**  
Cholesky Decomposition

**POLYNOMIAL FUNCTIONS**

**polyval(P,X)**  
returns the value of a polynomial  $P$  evaluated at  $X$ .  $P$  is a vector of length  $N+1$  whose elements are the coefficients of the polynomial in descending powers.  $Y = P(1) \cdot X^N + P(2) \cdot X^{N-1} + \dots + P(N) \cdot X + P(N+1)$ . If  $X$  is a matrix or vector, the polynomial is evaluated at all points in  $X$ .

You can also have scripts, scripts are basically just pooled up MATLAB statements like a long bunch of MATLAB statements. The best part is in MATLAB, you can have multiple arguments directly returned, multiple values directly returned, so something like `function(mean, median, mode, variance)=statistics(x)`, so whatever is the last set value of mean, median, mode, variance, it will be returned.

**(Refer Slide Time: 21:15)**



The image shows a recap slide with the following content:

**Recap**

**Topics covered**

- ▶ Matrices & Functions

**In the next video ...**

- ▶ Floating Point Arithmetic

You don't even need a return function in MATLAB. We will not worry about integrating ODEs and so on or some of the other toolboxes, so these are some of the basic things I wanted to show you. So, I hope you had a brief introduction to matrices and functions in MATLAB today, there is so many very good books on the topic, there are many interesting courses on this topic that you can find online.

In the next lecture, we will look at a very interesting aspect of not just MATLAB but computer arithmetic in general notably floating point arithmetic where I guarantee, you will have many surprises.