OIST mini course: MATLAB

Basics of programming and data analysis

Alexey Martyushev

Materials for the class

- Go to TIDA in your web-browser
- Find information about today's class in the announcements
- Click on it
- In the description, there should be a dropbox link

Materials for the class

- Go to TIDA in your web-browser
- Find information about the class in the announcements
- Click on it
- In the description, there should be a dropbox link

- Has everyone installed MATLAB to your laptop?

The content of the class

The content of the class

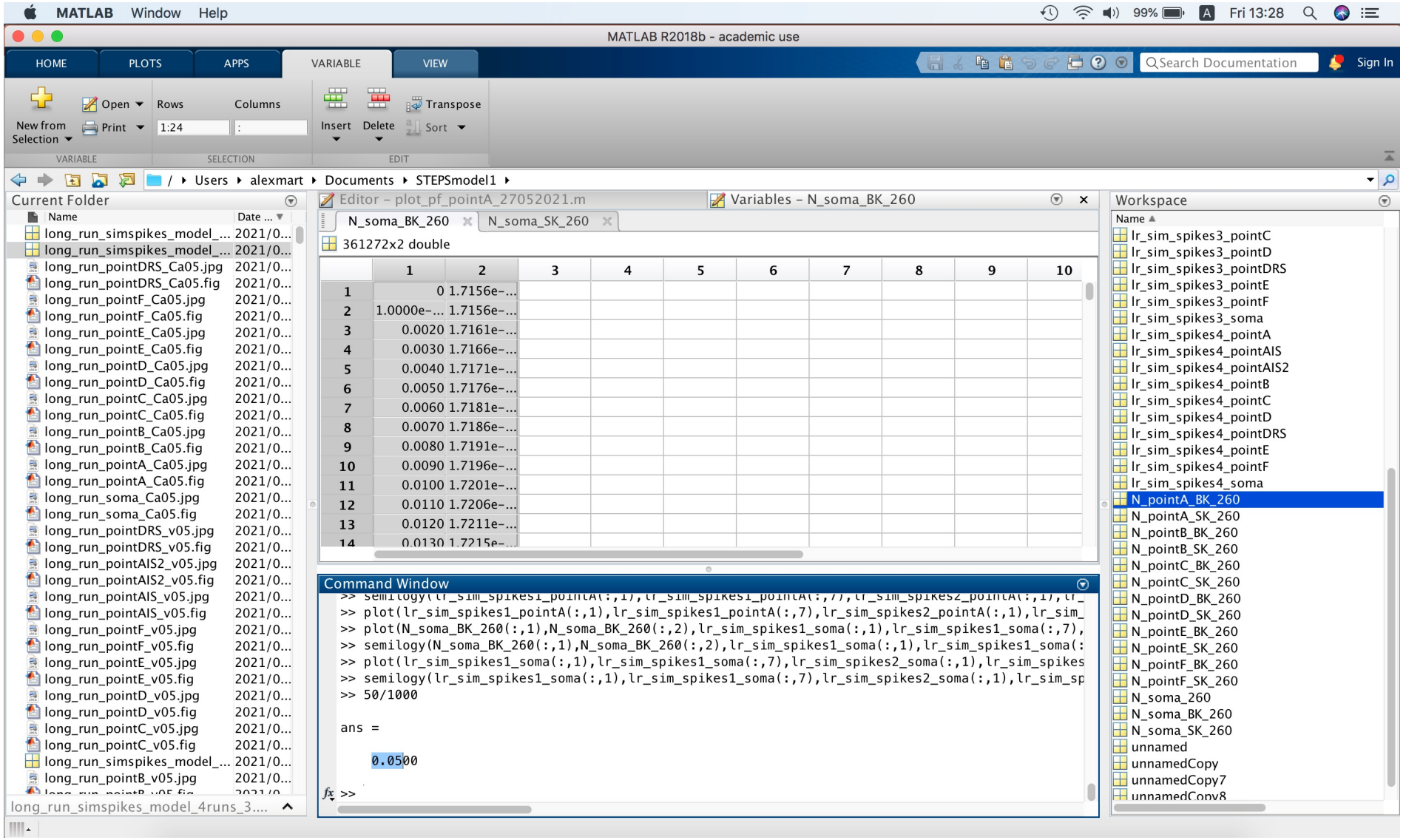
- MATLAB GUI (Graphical User Interface)
- Basics of programming in MATLAB
- Matrix computations
- Upload data to your computer
- Fitting of the model parameters to the data
- Plotting the results

The content of the class

- **MATLAB GUI (Graphical User Interface)**
- Basics of programming in MATLAB
- Matrix computations
- Upload data to your computer
- Fitting of the model parameters to the data
- Plotting the results

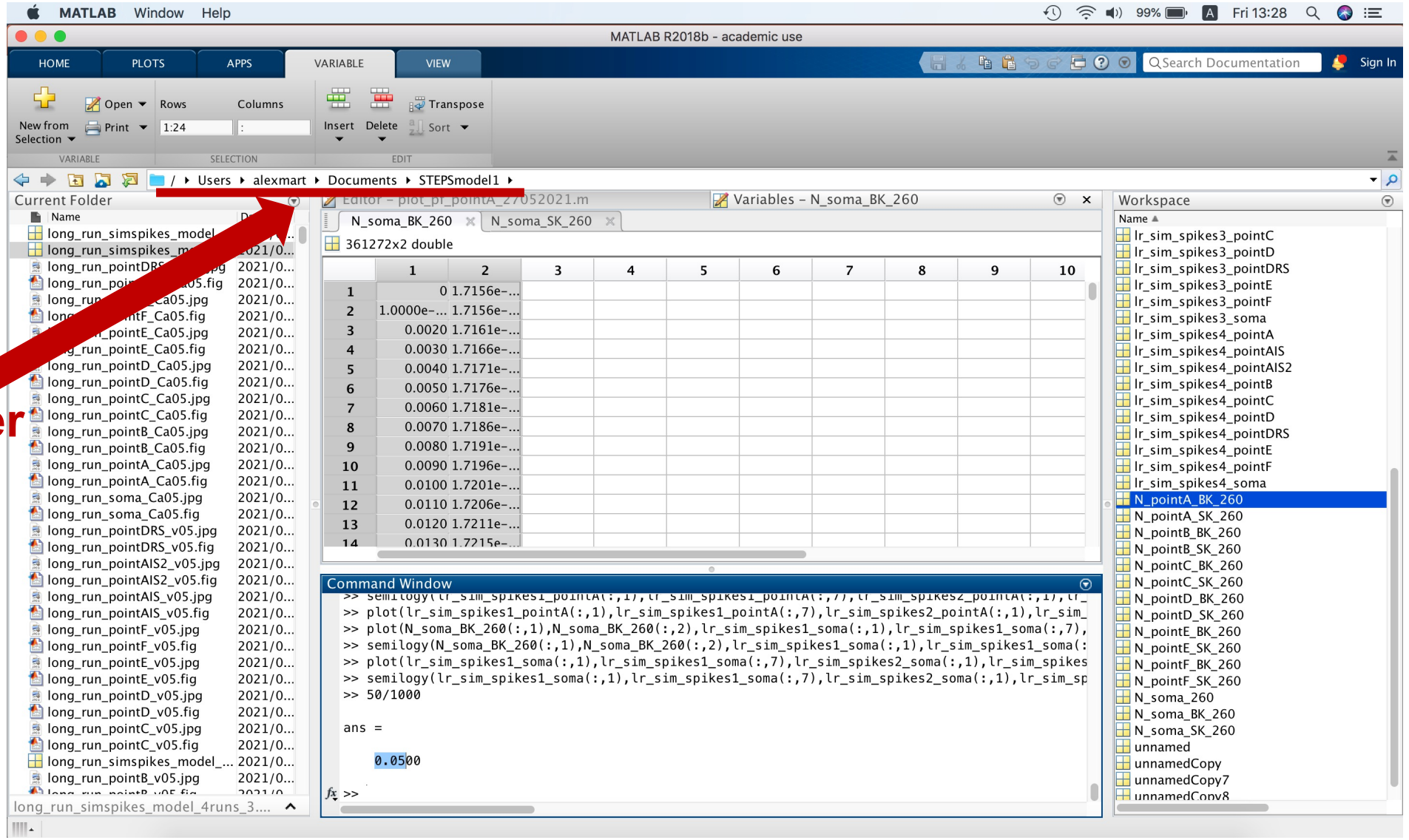
The content of the class

MATLAB GUI (Graphical User Interface)

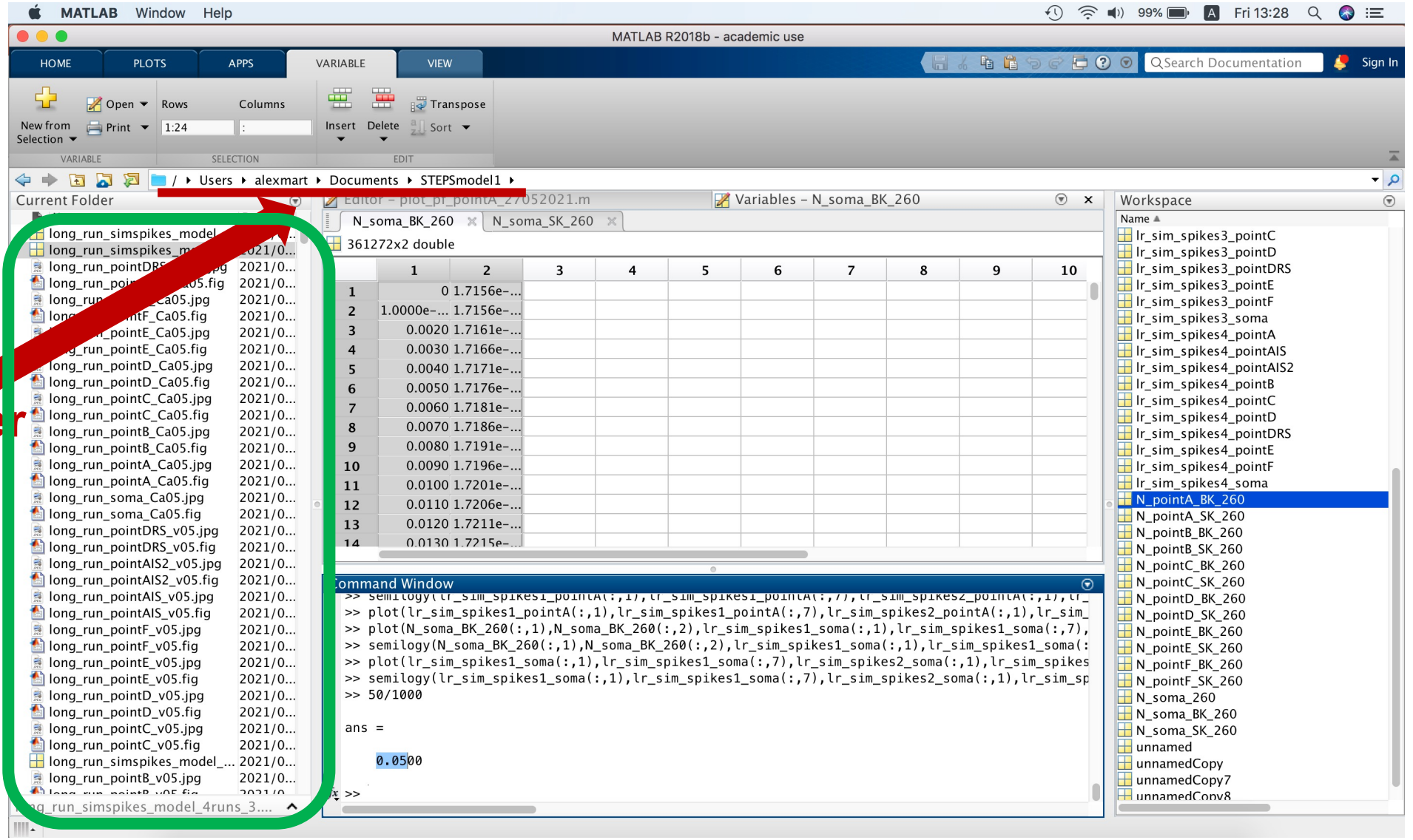


MATLAB GUI (Graphical User Interface)

Work folder



MATLAB GUI (Graphical User Interface)



Work folder

Folder content

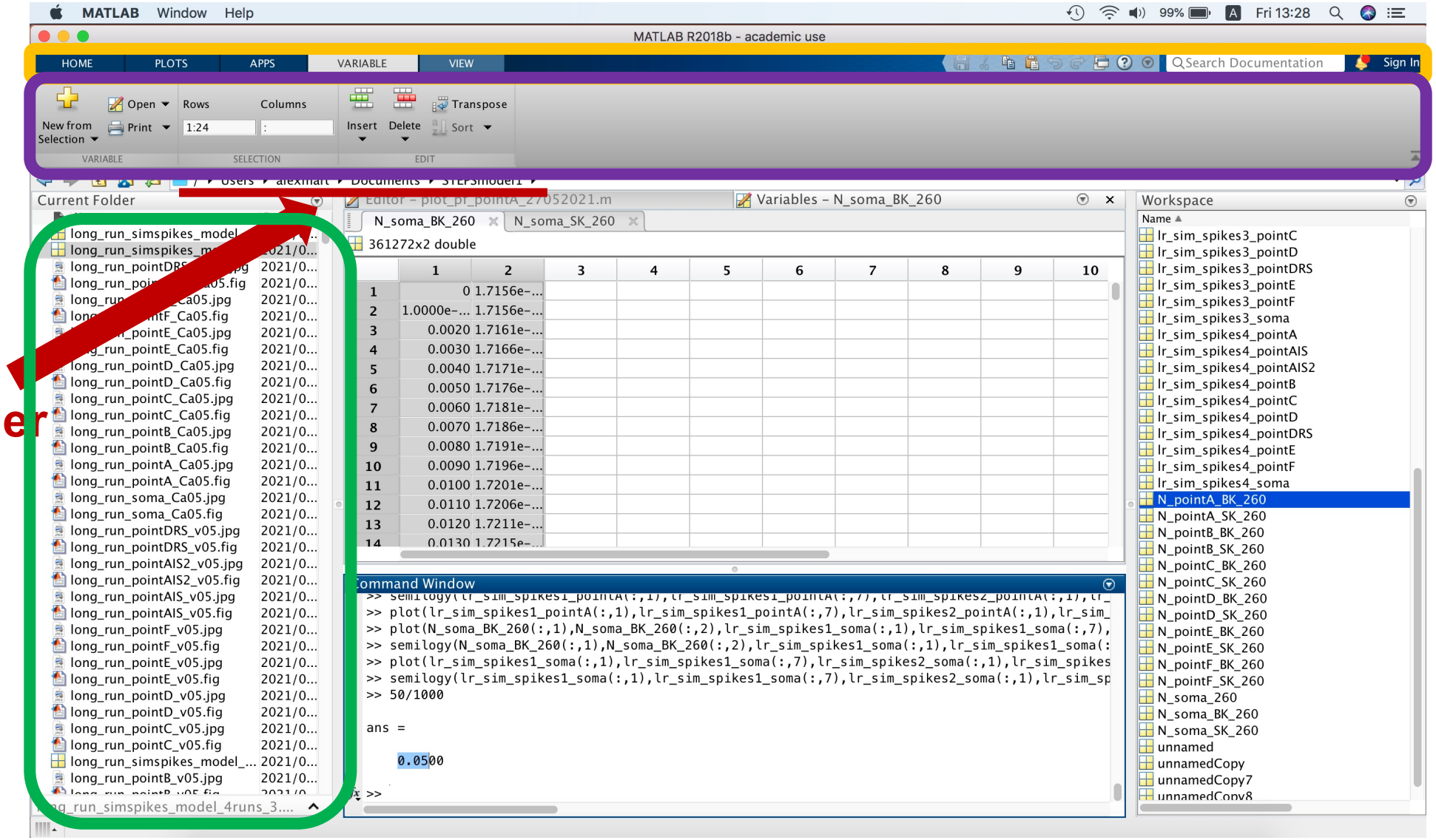
MATLAB GUI (Graphical User Interface)

Control tabs

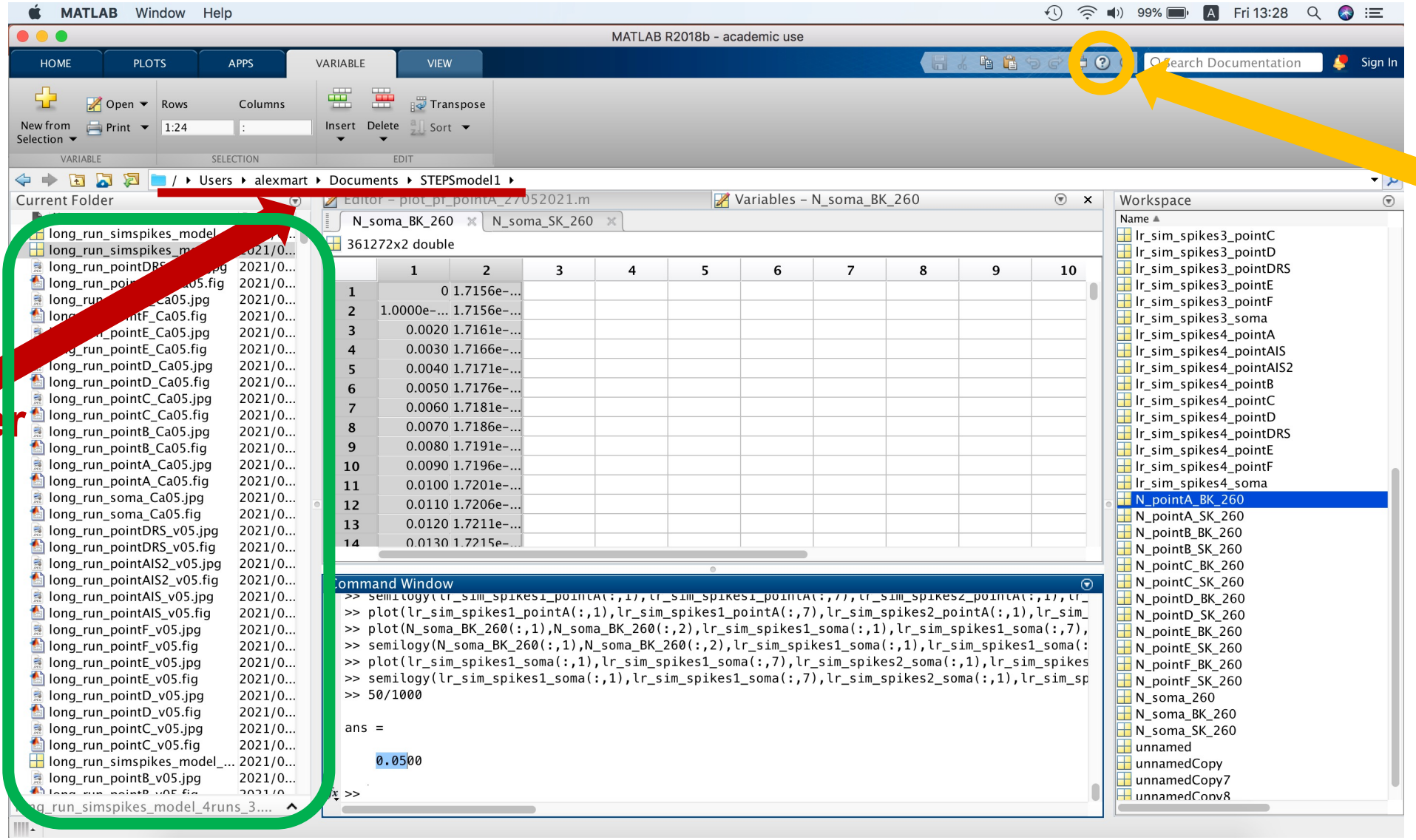
Buttons

Work folder

Folder content



MATLAB GUI (Graphical User Interface)

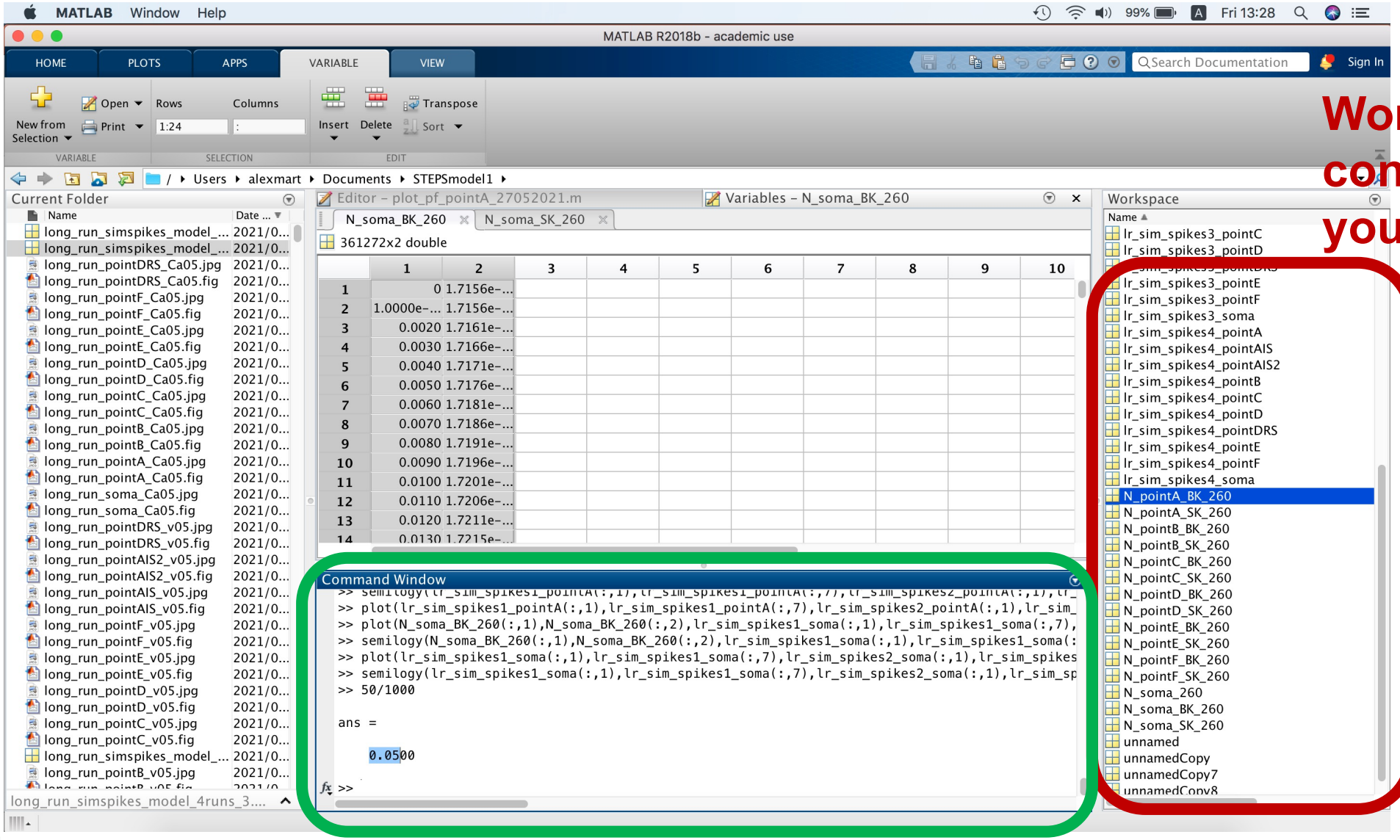


Work folder

Folder content

The most important button

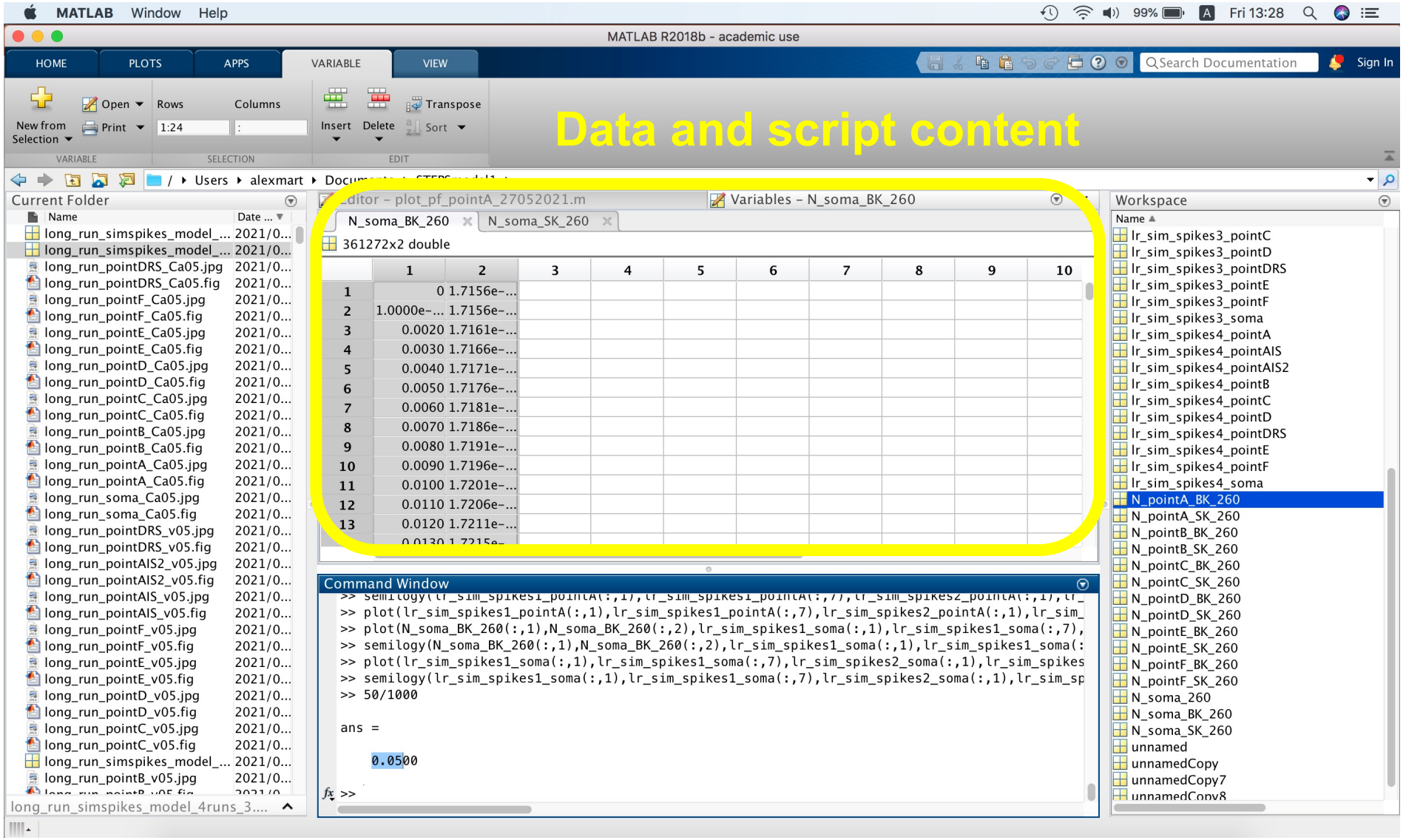
MATLAB GUI (Graphical User Interface)



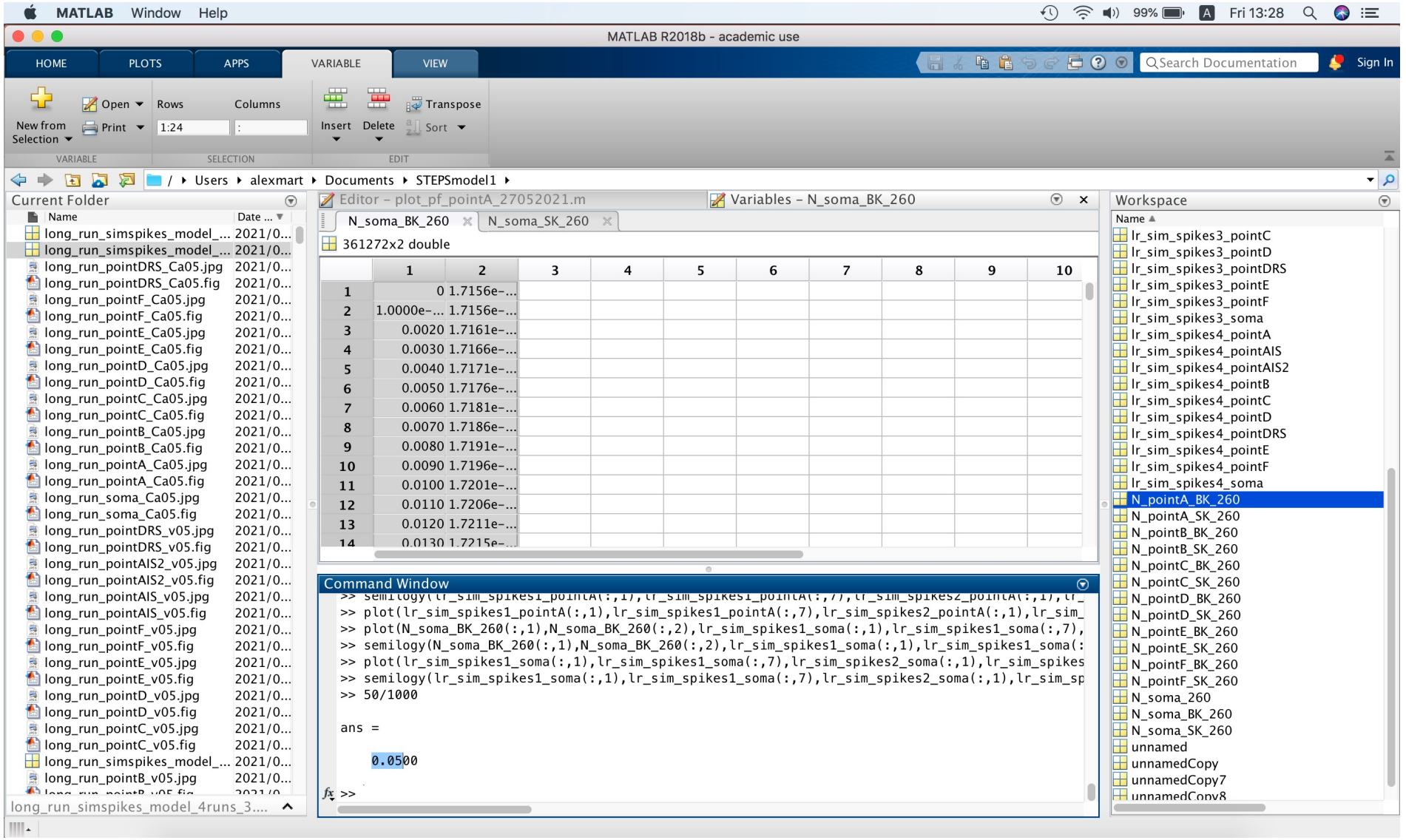
Workspace contains data you work with

Command window

MATLAB GUI (Graphical User Interface)



MATLAB GUI (Graphical User Interface)



The content of the class

- MATLAB GUI (Graphical User Interface)
- **Basics of programming in MATLAB**
- Matrix computations
- Upload data to your computer
- Fitting of the model parameters to the data
- Plotting the results

ADVANTAGES (list of some, not all):

- GUI: easy access to data, plot editor, and other functions
- Toolboxes (see APPS tab in your MATLAB): more than 20 are available for various scientific, engineering and data analysis needs (that requires you to do less coding)
- Easy debugging tools

DISADVANTAGES (list of some, not all):

- Organization of user-defined functions
- Some argue that it is slow, maybe…maybe…

BASICS OF CREATING YOUR MATLAB CODE:

- First, you need to create (or open an existing) script (function) file
- Data structures (VARIABLES) keep necessary information in your code such as single variables, arrays, strings.
- After you assigned some value to a variable(s), you provided an input to your code.
- If you wish the values of the variables been changed, as a result of some computations, you should state this in your code.

Basics of programming in MATLAB

DEFINING VARIABLES:

Not required, but may be necessary for some datatypes, like arrays, cells and other complex data structures

NAMES OF VARIABLES:

Can't begin with a number or other special characters (!@#$%^&*()_+=)

EXAMPLES (you may also type in the Command window):

```
>> n_v = 123; %integer variable
>> Duration_spikes = 0.785; %double variable
>> t_spikes = []; %empty array
```

VARIABLES CAN BE CONVERTED FROM ONE TYPE TO ANOTHER (with some conditions)

Basics of programming in MATLAB

If you want to print out a variable value, the simplest way is to type the name in you SCRIPT or in the COMMAND WINDOW:

For example:

```
>> n_v = 123
>> 123
```

or

```
>> n_v
>> 123
```

Basics of programming in MATLAB

For example:

```
>> s_v = 'SOME TEXT' %this is a string variable, and the text after % is just a comment stat will not be interpreted by MATLAB
>> 'SOME TEXT'
```

Basics of programming in MATLAB

If you want to display a text message to the command window, use this:

```
>> disp('SOME TEXT');
```

or if you want to display the content of a variable:

```
>> disp(s_v);
```

Using disp for displaying a variable means it must be some text, if a variable value is a number, you should convert it to string first:

```
>> n_v_text = num2str(n_v); disp(n_v_text);
```

More scientific ways of displaying a variables involve using:

```
>> fprintf('n_v is equal to %i.\n', n_v);
```

Or more complex:

```
>> float_point_variable = 123456789.123456789
>> 1.2346e+08
>> fprintf('Is equal to %6.2f.\n', float_point_variable);
>> 123456789.12
```

Briefly about arrays:

```
array1 = [1, 3, 4, 5, 7]; %is a row
array2 = [1; 2; 7; 8; 9]; %is a column
array3= [1, 2, 3, 4; 5, 6, 7, 8; 9, 10, 11, 12]; %is a 3x4 matrix
```

Access to data in arrays:

IMPOTANT: indexes start from 1 ~~not from 0~~.

```
>> array1(1)
>> 1
```

Briefly about arrays:

```
array1 = [1, 3, 4, 5, 7]; %is a row
array2 = [1; 2; 7; 8; 9]; %is a column
array3= [1, 2, 3, 4; 5, 6, 7, 8; 9, 10, 11, 12]; %is a 3x4 matrix
```

NOTE: using of **commas (,)** for inputting column elements in a row is NOT necessary, but **semicolon** is required to separate rows

```
array1 = [1 3 4 5 7]; %is a row
array2 = [1; 2; 7; 8; 9]; %is a column
array3= [1 2 3 4; 5 6 7 8; 9 10 11 12]; %is a 3x4 matrix
```

Basics of programming in MATLAB

```
array1 = [1, 3, 4, 5, 7]; %is a row
array2 = [1; 2; 7; 8; 9]; %is a column
array3= [1, 2, 3, 4; 5, 6, 7, 8; 9, 10, 11, 12]; %is a 3x4 matrix
```

```
>> array1(2)
>> 3
>> array2(3)
>> 7
```

array3(row number, column number) to access an element in array

```
>> array3(3,2)
>> 10
```

Basics of programming in MATLAB

```
array1 = [1, 3, 4, 5, 7]; %is a row
array2 = [1; 2; 7; 8; 9]; %is a column
array3= [1, 2, 3, 4; 5, 6, 7, 8; 9, 10, 11, 12]; %is a 3x4 matrix
```

If you want to change an element in the arrays:

```
>> array1(2) = 123;
>> array1(2)
>> 123
>> array3(3,2) = 111;
>> array3(3,2)
>> 111
```

MATRIX OR VECTOR TRANSPOSITION:

use “ ‘ ” apostrophe at the end of an array name

```
array1 = [1, 3, 4, 5, 7]; %is a row
array2 = [1; 2; 7; 8; 9]; %is a column
array3= [1, 2, 3, 4; 5, 6, 7, 8; 9, 10, 11, 12]; %is a 3x4 matrix
```

For example, try to apply transposition to the above arrays:

```
>> array1’
>> array2’
>> array3’
```

Basics of programming in MATLAB

If you need to test some condition in your code, for example, a>b, a<b, and a==b, then use the if-elseif-else statements.

IF, ELSEIF, ELSE STATEMENTS:

```
if (expression)
        statements;
elseif (expression)
        statements;
else
        statements;
end
```

```
%AN EXAMPLE:
a = 7;
b = 1;
if (a > b)
        disp('a is greater than b');
elseif (a == b)
        disp('a is equal to b');
else
        disp('a is less than b');
end
```

Basics of programming in MATLAB

If you need to test some condition in your code, for example, a>b, a<b, and a==b, then use the if-elseif-else statements.

IF, ELSEIF, ELSE STATEMENTS:

```
if (expression)
        statements;
elseif (expression)
        statements;
else
        statements;
end
```

IMPORTANT: the block must start with if and end up with end.
The key commands, as if, else, end are case-sensitive, don't use ~~CAPITAL~~ letter in them.
A statement should end up with semicolon ";" if you don't want the result been printed to the screen

Basics of programming in MATLAB

If you need to repeat, iterate or repeat doing something until a condition is true, then use loop statements.

LOOPS:

For, while, repeat until

AN EXAMPLE:

```
for index = values
        statements;
end
```

```
%AN EXAMPLE:
a = 7;
n = 10;
for i = 1:10
        if (i > a)
                fprintf('i = %i is greater than a = %i \n', i, a);
        elseif (i == a)
                fprintf('i = %i is equal to a = %i \n', i, a);
        else
                fprintf('i = %i is less than a = %i \n', i, a);
        end
end
```

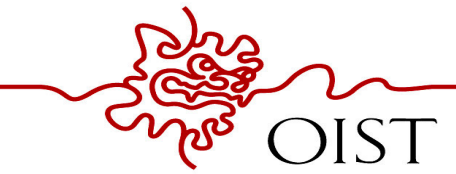
EXERCISE #1:

Let's imagine a situation you need to count the number of users in the age group between 18 and 24 (including person that are 18 and 24), and printout to the screen: 'The number of users of age between 18 and 24 is = the number'.

The input data comes as an array:

```
users_age = [18, 65, 34, 40, 25, 24, 45, 21, 34, 27, 22, 16, 21];
```

You can count directly for a small dataset, but if you have thousands of entries that is not easy.



EXERCISE #1

You have 10 minutes to do this task.

HINT:

Try to access the array elements in a “for” loop, use if-elseif-else to filter them, and “disp” or “fprintf” function to display the result.

The number of elements in users_age can be provided with ”length”, “size” functions, see manual

EXERCISE #2:

(For students who quickly completed Exercise #1, or for homework)

Try to pick first three elements from users_age that satisfy the age criteria from EXERCISE #1. Then, save to another array information about position of the picked elements in users_age. At the end, print to screen, the picked elements and their positions.

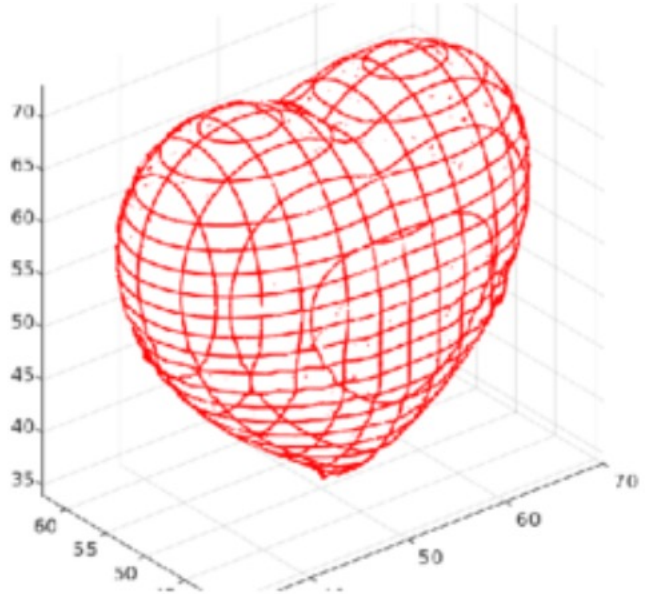
EXERCISE #2

You have 5 minutes to do this task.

HINT:

Try to use “while” or “repeat until” loops, and index variables to determine the position of picked elements.

Basics of programming in MATLAB



MY ANSWER (Exercise #1):

```
users_age = [18, 65, 34, 40, 25, 24, 45, 21, 34, 27, 22, 16, 21];

n = size(users_age,2); %or length(users_age)

count = 0;

for i=1:n
    if((users_age(i) >= 18) && (users_age(i) <= 24))
        count = count + 1;
    end
end

fprintf('The number of users between 18 and 24 years old is = %i \n', count);
```

Basics of programming in MATLAB

MY ANSWER (Exercise #2):

```
users_age = [18, 65, 34, 40, 25, 24, 45, 21, 34, 27, 22, 16, 21];
n = size(users_age,2); %or length(users_age)
i = 0;
count = 0;
age_array = zeros(1,3);
pos_array = zeros(1,3);
while (count<3)
    i = i + 1;
    if((users_age(i) >= 18) && (users_age(i) <= 24))
        count = count + 1;
        age_array(i) = users_age(i);
        pos_array(i) = i;
        fprintf('User of age %i in position of %i of the array is found!\n', age_array(i), pos_array(i));
    end
end
```

The content of the class

The content of the class

- MATLAB GUI (Graphical User Interface)
- Basics of programming in MATLAB
- **Matrix computations**
- Upload data to your computer
- Fitting of the model parameters to the data
- Plotting the results

Matrix computations in MATLAB

SIMPLEST OPERATORS:

EXMAPLE:

```
A = [1, 2, 3, 4; 5, 6, 7, 8; 9, 10, 11, 12];
b = [1; 2; 3; 4];

>> A*b
>> ans =
    30
    70
   110
```

A*b = [1*1 + 2*2 + 3*3 + 4*4; 5*1 + 6*2 + 7*3 + 8*4; 9*1 + 10*2 + 11*3 + 12*4];

Matrix computations in MATLAB

SIMPLEST OPERATORS:

EXMAPLE:

```
A = [1, 2, 3, 4; 5, 6, 7, 8; 9, 10, 11, 12];
b = [1; 2; 3; 4];
>> A + b
>> Matrix dimensions must agree.
```

Matrix computations in MATLAB

SIMPLEST OPERATORS:

EXMAPLE:

```
A = [1, 2, 3, 4; 5, 6, 7, 8; 9, 10, 11, 12];
b = [1; 2; 3; 4];
>> A + b
>> Matrix dimensions must agree. Why?
```

Matrix computations in MATLAB

SIMPLEST OPERATORS:

EXMAPLE:

```
A = [1, 2, 3, 4; 5, 6, 7, 8; 9, 10, 11, 12];
b = [1; 2; 3; 4];
>> A + b
>> Matrix dimensions must agree.
```

Why?

Because "+" operator sums single elements in A and b. If the number of single elements is different between A and b, the error message pops up

SIMPLEST OPERATORS:

A correct method of using "+" operator will be for:

```
c = [5, 6, 7, 8];
d = [9, 10, 11, 12];
>> c + d
>> ans =
    14    16    18    20
```

SIMPLEST OPERATORS:

Or for:

```
A = [1, 2, 3, 4; 5, 6, 7, 8; 9, 10, 11, 12];
E = [10, 11, 16, 12; 3, 8, 6, 9; 2, 1, 3, 0];

>> A + E
```

SIMPLEST OPERATORS:

```
a = [1:5]; %[1, 2, 3, 4, 5]
a>3; %[0, 0, 0, 1, 1] %could also be <, >=, <=, ==
a(a>3); %[4, 5]
a.*a;  %[1, 4, 9, 16, 25]
```

Matrix computations in MATLAB

Excersise #3:

You buy some quantity of apples, bananas, mango, and pineapples three months in a row. The price of these fruits was constant during these three months. Could we calculate the amount of money you spent on fruits during that period?

5 minutes to do

HINT:

Use matrix computations and “sum” function (check in the manual )

Matrix computations in MATLAB

Excersise #4 (optional):

Using matrix (or) vector operators, perform the task of Exercise #1 and #2 (optional).

10 minutes to do

HINT: no hints to experts!

MY ANSWER (Exercise #3):

```
cost = [3.5 1.5 5 4.5];

quatity_mon1 = [10 12 5 2];
quatity_mon2 = [12 8 4 1];
quatity_mon3 = [11 7 3 5];

Q_array = [quatity_mon1; quatity_mon2; quatity_mon3];
Spent_money = sum(Q_array*(cost'));
fprintf('The amount of money spent on fruits for the last three months
is %.2f\n', Spent_money);
```

MY ANSWER (Exercise #4a):

In principle, with using MATLAB in a better way the whole code for the task could be looking like this:

```
users_age = [18, 65, 34, 40, 25, 24, 45, 21, 34, 27, 22, 16, 21];
count = sum((users_age >= 18).*(users_age <= 24));
fprintf('The number of users between 18 and 24 years old is = %i \n', count);
```

Basics of programming in MATLAB

MY ANSWER (Exercise #4b):

```
users_age = [18, 65, 34, 40, 25, 24, 45, 21, 34, 27, 22, 16, 21];
n = length(users_age);
pos = [1:n];

indexes = ((users_age >= 18).*(users_age <= 24));
picked_users = users_age(indexes>0);
postions = pos(indexes>0);

picked_users = picked_users([1:3]);
postions = postions([1:3]);

disp('Picked users:');
disp(picked_users);
disp('Their indexes:');
disp(postions);
```

The content of the class

- MATLAB GUI (Graphical User Interface)
- Basics of programming in MATLAB
- Matrix computations
- **Data analysis: upload data to your computer**
- Fitting of the model parameters to the data
- Plotting the results

Upload data to your computer

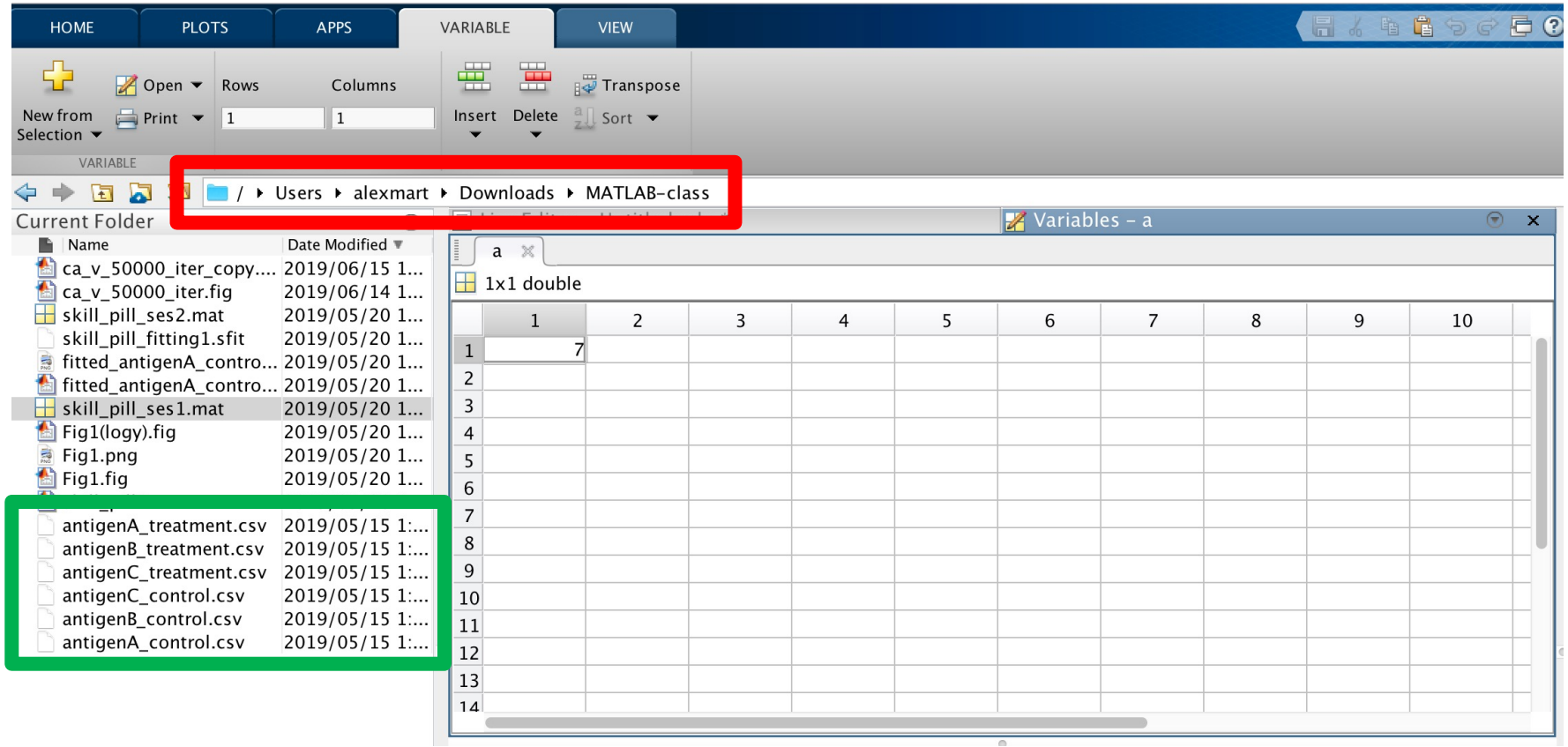
- Please access .csv data files through the DropBox

- After download, you should have six files on your computer in a folder:

1) antigenA_control.csv 2) antigenB_control.csv
3) antigenC_control.csv 4) antigenA_treatment.csv
5) antigenB_treatment.csv 6) antigenC_treatment.csv

Upload data to your computer

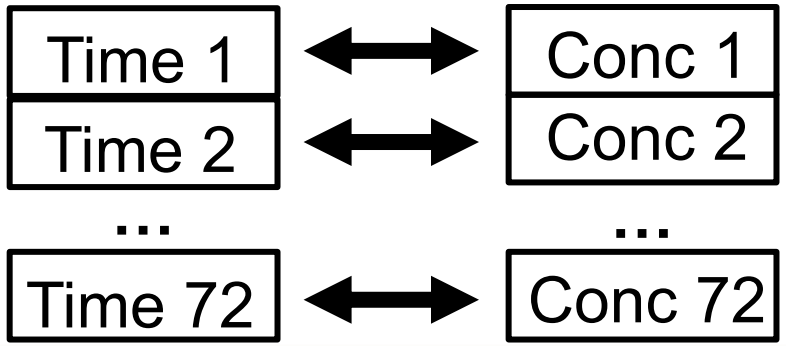
- Go to the folder with the downloaded files in the MATLAB interface, until you get the files listed in the left panel:



Data interpretation

- Each of the six files is a time series of a specific antigen (A, B or C) measured in hosts of a group (control or treatment)

- In other words, each csv file can be interpreted as a table of two columns, the first column is the time when an antigen was measured and the second column is the concentration of the antigen)



We have 72 elements in each row of the data

Uploading the data to the workspace

- Upload data to MATLAB (type or copy-paste to the Command Window):

```
antigenA_control = csvread('antigenA_control.csv');

antigenB_control = csvread('antigenB_control.csv');

antigenC_control = csvread('antigenC_control.csv');


antigenA_treatment = csvread('antigenA_treatment.csv');

antigenB_treatment = csvread('antigenB_treatment.csv');

antigenC_treatment = csvread('antigenC_treatment.csv');
```

Organizing the data

- If you want to merge all data to one big table (use console):

```
resTable = [antigenA_control(:,1), antigenA_control(:,2),
antigenB_control(:,2), antigenC_control(:,2), antigenA_treatment(:,2),
antigenB_treatment(:,2), antigenC_treatment(:,2)];
```

Time column

Concentration column

IMPORTANT: because column 2 in our csv data corresponds to the time column 1, therefore, the merging as shown above is correct only when the first column is the same across all tables to be merged (fortunately, this is our case)

- Use "," for merging columns or " "
- Use ";" for merging rows into one row

Organizing data in MATLAB

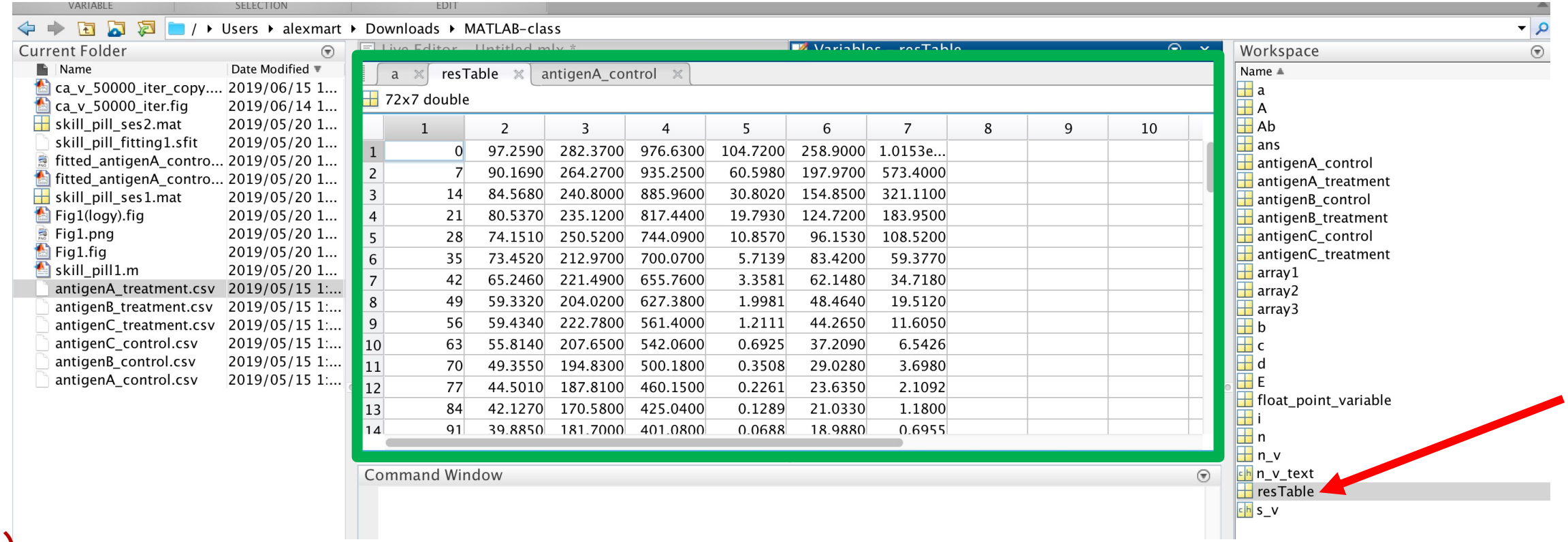
In the resulting table we will have the first column for the measurement times, and other six columns for the antigen concentrations:

Time	A_ctrl	B_ctrl	C_ctrl	A_treat	B_treat	C_treat
Day t1	#	#	#	#	#	#
Day t2	#	#	#	#	#	#
…						

Double click on the `resTable` in the Workspace to open (~~and edit~~) the data.

Organizing data in MATLAB

Double click on the resTable in the Workspace to open (~~and edit~~) the data.



A correct mathematical model of the data

- We can plot our data…
- In the Command Window, type this:

```
plot(resTable(:,1), resTable(:,2)); %plot the data
```

Or:

```
plot(resTable(:,1), resTable(:,2), resTable(:,1), resTable(:,3));
```

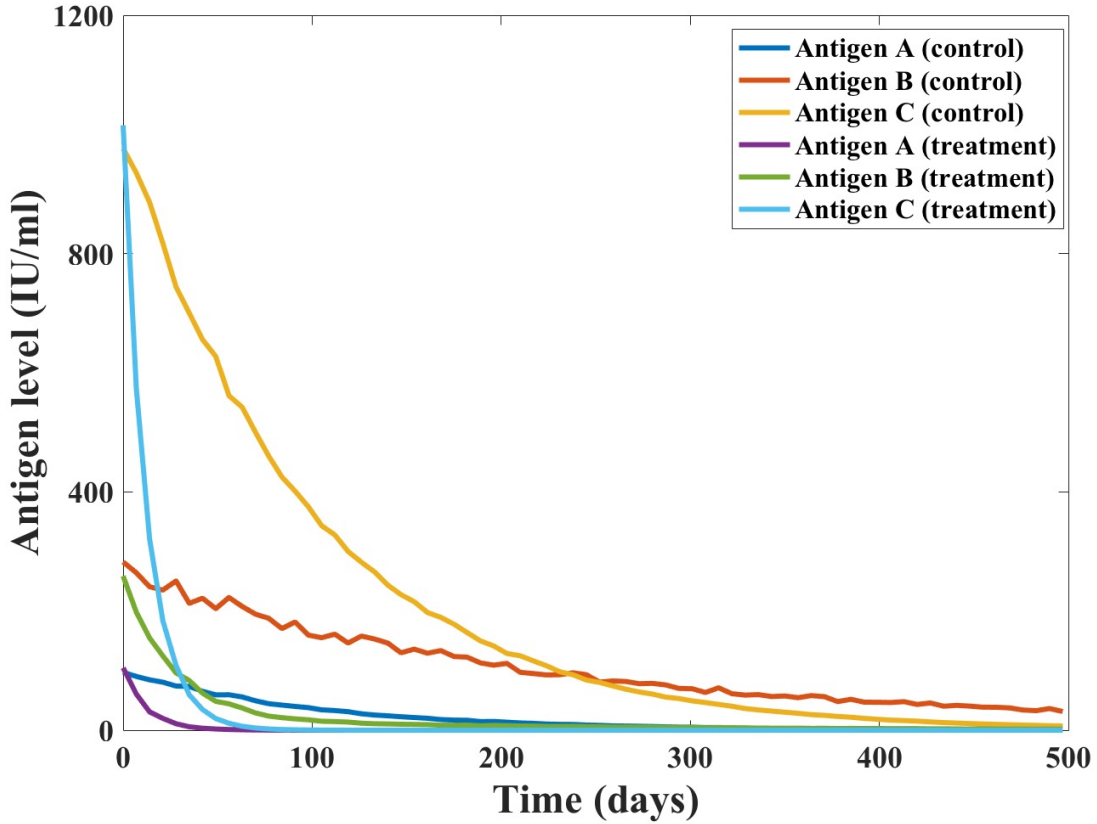
- Also, you can use different plotting axis scales instead of linear "plot" you can use "semilogy" (linear x-axis and logarithmic y-axis), "semilogx" (logarithmic x-axis and linear y-axis), loglog(both axis are logarithmic)

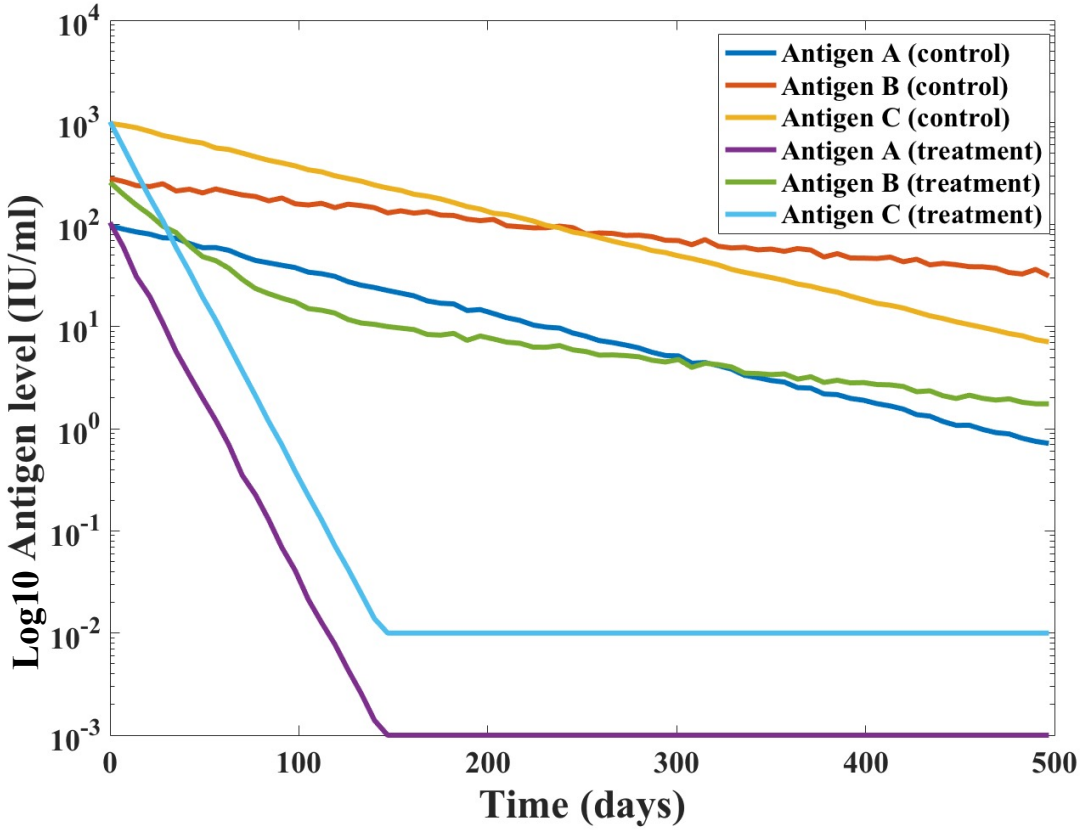
A correct mathematical model of the data

- Plot all together by using `plot(t, y1, t, y2, t, y3, t, y4, t, y5, t, y6);`
- Note: you must use a time column (t) every time before the data column in the input arguments of the plot functions
- Or, `plot(resTable(:,1), resTable(:,[2:7]));`

A correct mathematical model of the data

- plot(resTable(:,1), resTable(:,[2:7]))





Exponential decay models

- Try to search in "Exponential models" in MATLAB help
- Let's consider two types of the model, single-phase:

  Antigen concentration (t) = $A*e^{k1*t}$ + *plateau1*

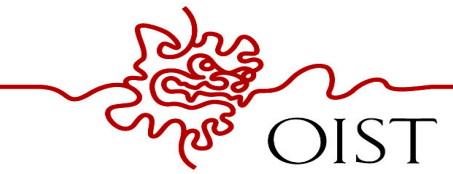
- and double-phase decay:

  Antigen concentration (t) = $B*e^{k2*t}$ + $C*e^{k3*t}$ + *plateau2*

Note: in the models A > 0, B > 0, C > 0 and k1 < 0, k2 < 0, k3 < 0 (because the antigen concentration decays, for a growth k should be positive). Also, *plateau1 > 0* and *plateau2 > 0*. The plateaus represent detection limits of the assays.

The content of the class

- MATLAB GUI (Graphical User Interface)
- Basics of programming in MATLAB
- Matrix computations
- Upload data to your computer
- **Fitting of the model parameters to the data**
- Plotting the results

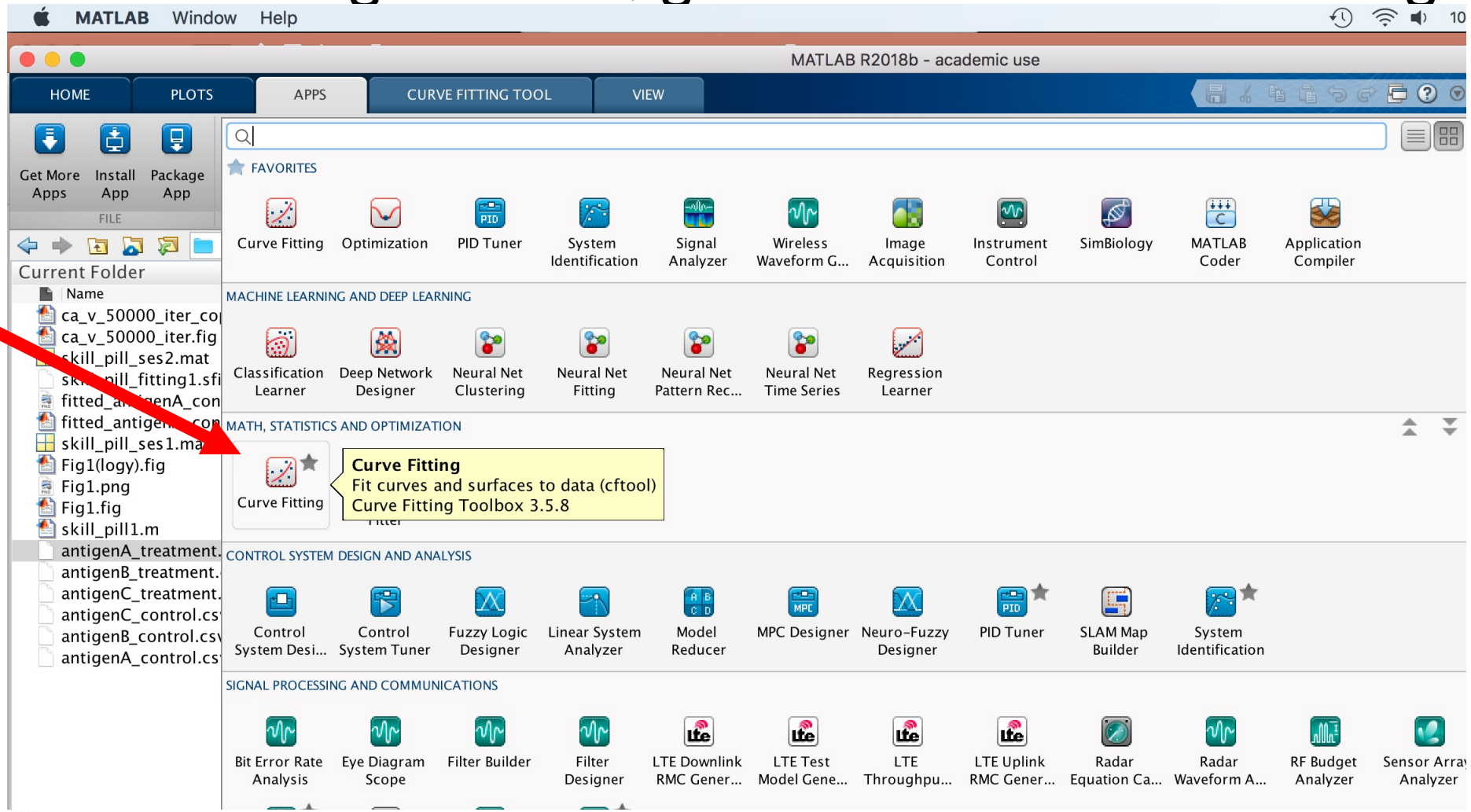


Fitting of the model parameters

- Use Curve Fitting Toolbox, go to APPS - > Curve Fitting
- Setting up the toolbox
- Results

Fitting of the model parameters

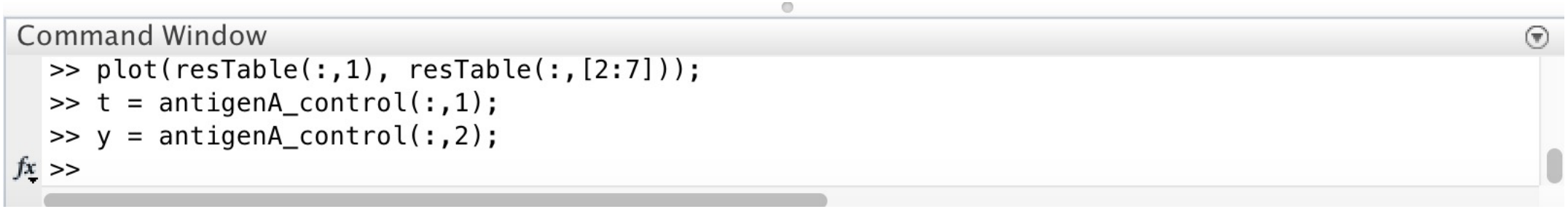
- Use Curve Fitting Toolbox, go to APPS - > Curve Fitting



Fitting of the model parameters

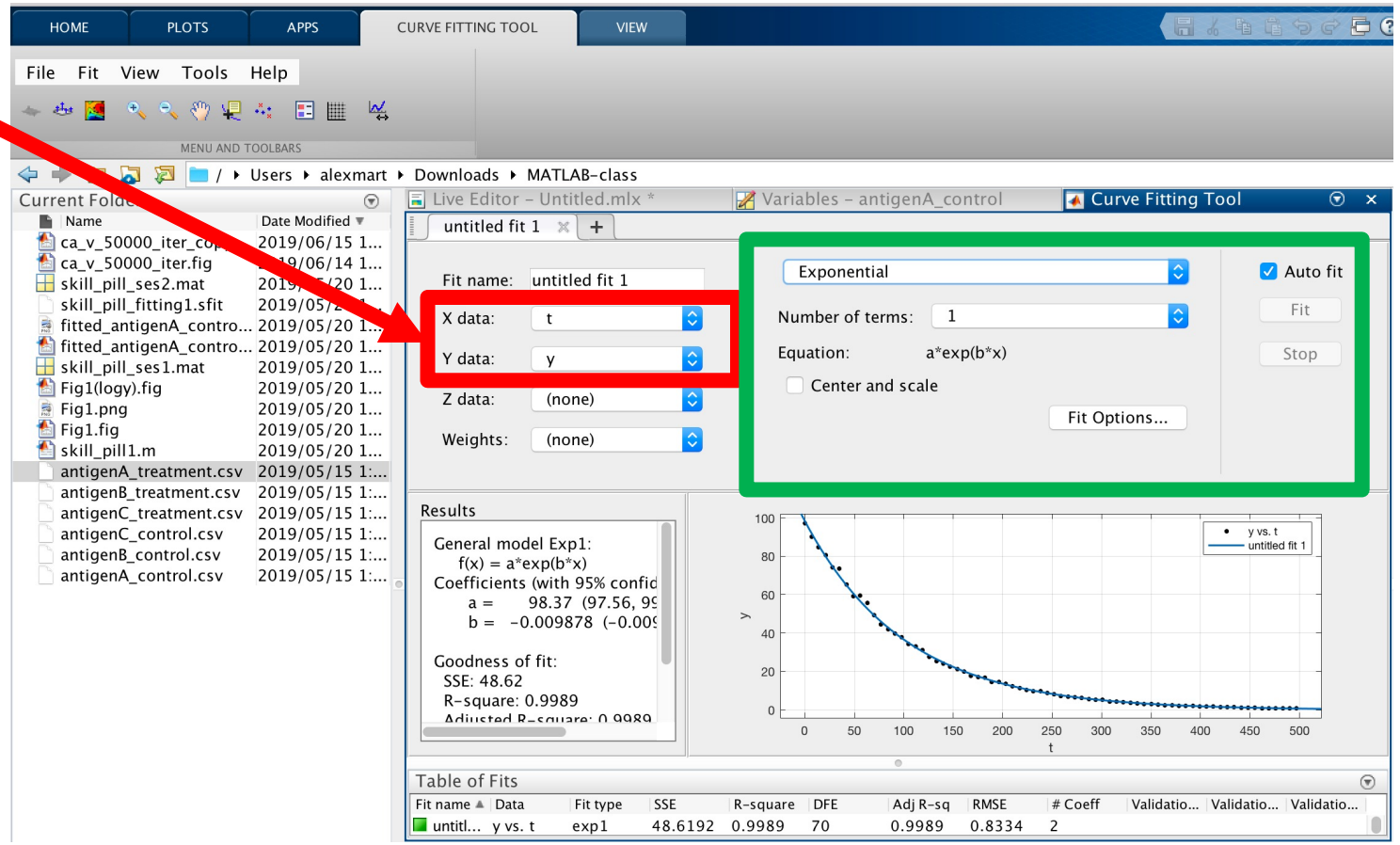
- Use Curve Fitting Toolbox, go to APPS - > Curve Fitting
- Setting up the toolbox

```
>> t = antigenA_control(:,1);
>> y = antigenA_control(:,2);
```



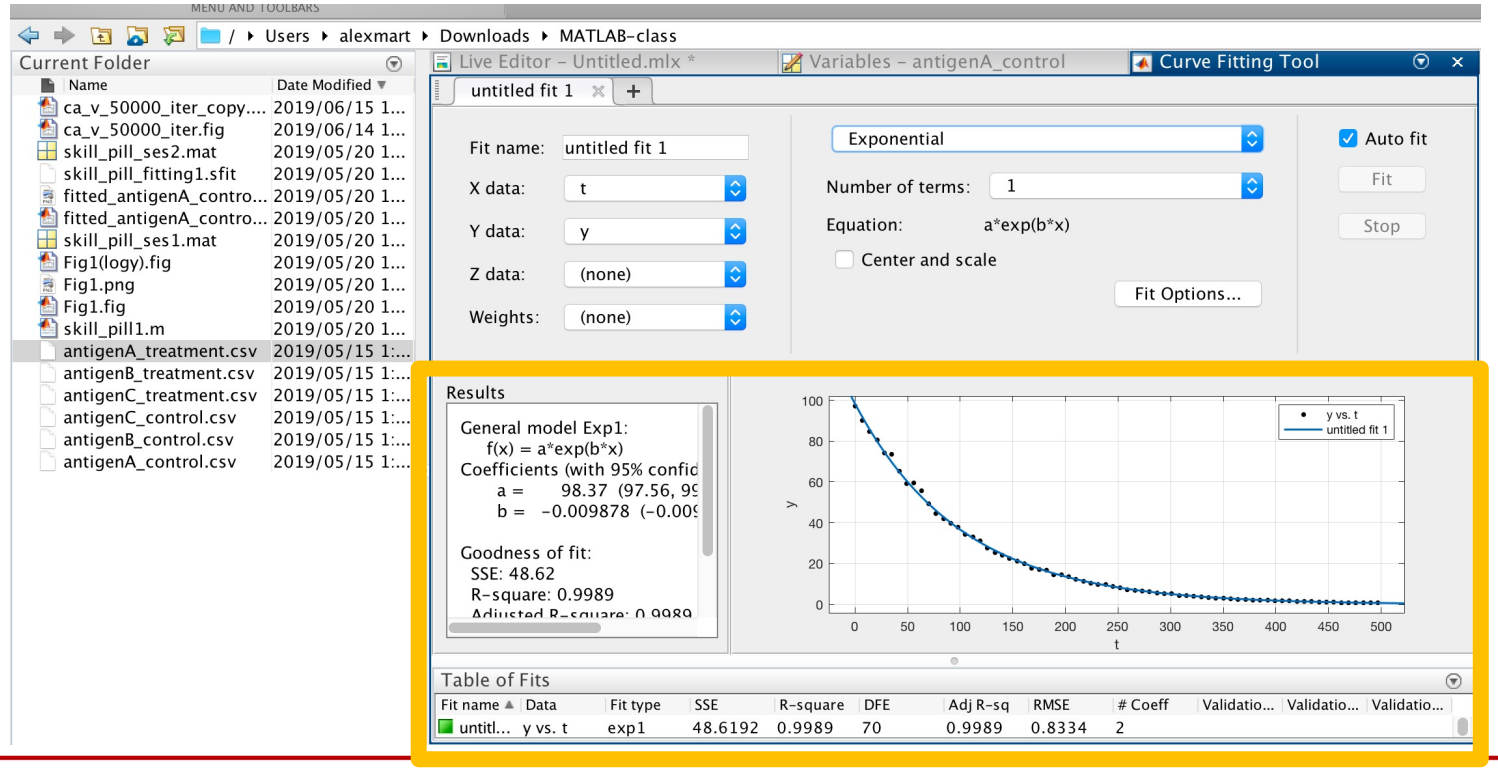
Fitting of the model parameters

- Use Curve Fitting Toolbox, go to APPS - > Curve Fitting
- Setting up the toolbox



Fitting of the model parameters

- Use Curve Fitting Toolbox, go to APPS - > Curve Fitting
- Setting up the toolbox
- Results



Fitting of the model parameters

Excersise #5:

Do the same as for antigenA_control, but for:

antigenB_control, antigenC_control, antigenA_treatment, antigenB_treatment, antigenC_treatment

5 minutes to do

Fitting of the model parameters

- Group comparison tests to compare decay rates of antigens between the groups (control and treatment)
- Correlation tests (Spearman, Pearson)
- All of them are available as functions in MATLAB, check in the manual

Thanks

Thank you all for visiting our class today!