Intro to MATLAB
Part 2/3
Amandine Gamble
amandinegamble@ucla.edu
Outline of the workshop

Day 1
- Interface
- Command lines and basic syntax
- Variables and operations
- Scripts
- if statements

Day 2
- for and while loops
- More matrices
- Functions
- Files

Day 3
- Plotting
- Introduction to dynamical systems: ODEs
if statements
(reminder)
if statements

exp1 = 400;
exp2 = 500;
if (exp1 < exp2)
    min = exp1
end
**if statements**

```
exp1 = 400;
exp2 = 500;
if (exp1 >= exp2)
    max = exp1
else
    max = exp2
end
```
if statements

```python
exp1 = 400;
exp2 = 500;
if (exp1 > exp2):
    max = exp1
elseif (exp1 == exp2):
    max = 'both are equal'
else:
    max = exp2
end
```
for and while loops
for and while loops

Loop \rightarrow Check condition \rightarrow True \rightarrow Run while code \rightarrow Stop

False

Useful for simulation of dynamic processes over time

```matlab
aa = 10;
% while loop execution
while (aa < 20)
    disp(aa)
    aa = aa + 1;
end
```

```
% for loop execution
for (bb = 10:19)
    disp(bb)
end
```

Need to declare the index

Incrementation

Index declaration and incrementation happen in for

1. Make it print 20
2. Make it display a sentence indicating the value of a (e.g., ‘value of aa: ’)
for and while loops

• A longer algorithm:

```matlab
fprintf('New simulation\n');
aa = 10;
bb = 10;
% while loop execution
while (aa <= 20)
    % e.g., aa is a time step
    fprintf('value of aa: %d\n', aa);
aa = aa + 1;
    % e.g., bb is a variable that doubles at every time step
    fprintf('value of bb: %d\n', bb);
    bb = bb*2;
end
```
for and while loops

- break statements

```matlab
fprintf('New simulation\n');
aa = 10;
bb = 10;
% while loop execution
while (aa <= 20)
    % e.g., aa is a time step
    fprintf('value of aa: %d\n', aa);
    aa = aa + 1;
    % e.g., bb is a variable that doubles at every time step
    fprintf('value of bb: %d\n', bb);
    bb = bb*2;
    if (bb > 5000)
        % terminate the loop
        break;
    end
end
```

Print aa and bb in the same command
for and while loops

- break statements

Loop

Check while condition

Check if condition

Run while code

Run if code (break)

Stop
for and while loops

• Nested loops

```matlab
fprintf('New simulation\n');
% for loop execution
% e.g., define a grid
for (aa = 1:10)
  for (bb = 1:10)
    fprintf('aa is %d and bb is %d\n' , aa, bb);
  end
end
```
More matrices
Calling a $i^{\text{th}}$ position in a matrix

• With a vector

```matlab
>> myVector = [4, 5, 6, 7, 9];
>> myVector(1)
ans =
   4
>> myVector(4)
ans =
   7
>> myVector(2:4)
ans =
   5   6   7
>> myVector(2:length(myVector))
ans =
   5   6   7   9
```
Calling a $i^{th}$ position in a matrix

- With a matrix

```matlab
>> myMatrix = [1, 2, 3; 4, 5, 6; 7, 8, 9]

myMatrix =

    1     2     3
    4     5     6
    7     8     9

>> myMatrix(1, 2)

ans =

    2

>> myMatrix(2, 1)

ans =

    4
```
Calling a $i^{th}$ position in a matrix

• With a matrix

```matlab
>> myMatrix(1, :)
ans =
    1     2     3

>> myMatrix(:, 1)
ans =
    1
    4
    7
```
Creating special matrices

• Matrices of 0s and 1s with the functions \texttt{zeros} and \texttt{ones}

\begin{verbatim}
>> zeros(3)
ans =
    0     0     0
    0     0     0
    0     0     0
>> ones(1, 4)
ans =
    1     1     1     1
    1     1     1     1
    1     1     1     1
    1     1     1     1
>> ones(4, 5)
ans =
    1     1     1     1     1
    1     1     1     1     1
    1     1     1     1     1
    1     1     1     1     1
\end{verbatim}
High-dimensional matrices

```matlab
>> ones(4, 5, 2)

ans(:,:,1) =
    1     1     1     1     1
    1     1     1     1     1
    1     1     1     1     1
    1     1     1     1     1

ans(:,:,2) =
    1     1     1     1     1
    1     1     1     1     1
    1     1     1     1     1
    1     1     1     1     1
```

 Rows

 Columns

 Pages
Edit elements in a matrix

```matlab
>> myMatrix = ones(4, 5, 2);

>> myMatrix(1, 1, 1)
ans =
     1

>> myMatrix(1, 1, 1) = 2

myMatrix(:,:,1) =
     2     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1

myMatrix(:,:,2) =
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
```
Sorting

- Sort elements with the `sort` function

```matlab
>> myVector = [5, 12, 8, 0];
>> sort(myVector)
ans =
    0  5  8  12
```

```matlab
>> myMatrix = [12, 8, 5; 1, 22, 4];
>> sort(myMatrix, 1)
ans =
    1  8  4
    12 22  5

>> sort(myMatrix, 2)
ans =
    5  8  12
    1  4  22
```
Cell arrays

- Like matrices but different
  - Defined with {} instead of [ ]
  - Handle strings better
    Avoid issues with string dimensions

```matlab
>> geneNames = {'1L1A', 'NFKBIA', 'BCL2'}
geneNames =
    1×3 cell array
    {'1L1A'}    {'NFKBIA'}    {'BCL2'}
```

```matlab
>> geneNames = ['1L1A', 'NFKBIA', 'BCL2']
geneNames =
    '1L1ANFKBIABCL2'
```

```matlab
>> geneNames = ['1L1A'; 'NFKBIA'; 'BCL2']
Error using vertcat
Dimensions of arrays being concatenated are not consistent.
```
Cell arrays

• Generating cell arrays with the function `cell`

```matlab
>> myCells = cell(2, 3);
>> myCells(1, :) = {'1L1A', 'NFKBIA', 'BCL2'}
myCells =

2×3 cell array

    {'1L1A'   }    {'NFKBIA'  }    {'BCL2'    }
    {0×0 double}    {0×0 double}    {0×0 double}

>> myCells(2, :) = {120, 446, 653}
myCells =

2×3 cell array

    {'1L1A'}    {'NFKBIA'}    {'BCL2'}
    {[ 120]}    {[   446]}    {[ 653]}
```
Cell arrays

• ( ) refers to a set of cells
• { } refers to the data within the cells

```
>> class(myCells(2, 1))
an =
    'cell'
>> class(myCells{2, 1})
an =
    'double'
```

```
>> myCells(1, :)
an =
    1×3 cell array
    {'1L1A'}    {'NFKBIA'}    {'BCL2'}

>> myCells{1, :}
an =
    '1L1A'
    ans =
    'NFKBIA'
    ans =
    'BCL2'
```
Cell arrays, for loops and if statements

- Example: I have a set of genes and their levels of expression; I want to know the names and levels of expression of the genes expressed above a set threshold

1. Look at each gene one by one
2. Compare its level of expression to the threshold
3. Display the ones that have higher levels of expression

```matlab
exp1Genes = {'1L1A', 'NFKBIA', 'BCL2'};
exp1Results = [120, 446, 653];
threshold = 200;
for (ii = 1:length(exp1Results))
    if (exp1Results(ii) > threshold)
        fprintf('Genes %s expression %d\n', exp1Genes{ii}, exp1Results(ii))
    end
end
```
String comparisons

• Compare strings with the `strcomp` function

```
exp1Genes = {'1L1A', 'NFKBIA', 'BCL2'};
exp1Results = [120, 446, 653];
geneOfInterest = 'BCL2';
for (ii = 1:length(exp1Results))
    if (strcmp(exp1Genes{ii}, geneOfInterest))
        fprintf('Gene %s expression %d\n', exp1Genes{ii}, exp1Results(ii))
    end
end
```

```
>> 'BCL2' == 'BCL2'
ans =
1×4 logical array
1 1 1 1
```
String comparisons

• Compare strings with the `strcomp` function

```matlab
exp1Genes = {'1L1A', 'NFKBIA', 'BCL2'};
exp1Results = [120, 446, 653];
geneOfInterest = 'BCL2';
for (ii = 1:length(exp1Results))
    if (strcmp(exp1Genes{ii}, geneOfInterest) == 1)
        fprintf('Gene %s expression %d\n', exp1Genes{ii}, exp1Results(ii))
    end
end
```

Keep those scripts saved or open, we will reuse them
Functions
Function syntax

function [outputs] = functionName (inputs)

• In MATLAB, functions have to be in their own script file functionName.m
Our first function

```matlab
function infoSeq = describeSeq(seq)
    seqLength = length(seq);
    infoSeq = sprintf('The sequence %s has %d nucleotides', seq, seqLength);
end
```

2. Save in the working directory with `fileName = functionName`

1. Write the function in a script

3. Run

```matlab
mySeq = 'CTAGGCTGAT';
describeSeq(mySeq)
ans =
    'The sequence CTAGGCTGAT has 10 nucleotides'
```
A more complicated function

Add function definition: name, output, input

```matlab
function out = printGeneExpression(exp1Genes, exp1Results, geneOfInterest)

exp1Genes = {'1L1A', 'NFKBIA', 'BCL2'};
exp1Results = [120, 446, 653];
geneOfInterest = 'BCL2';
for (ii = 1:length(exp1Results))
    if (strcmp(exp1Genes{ii}, geneOfInterest))
        fprintf('Gene %s expression %d\n', exp1Genes{ii}, exp1Results(ii));
    end
end
out = exp1Results(ii);
```

Transform this into a function

Remove input definition (happens outside of the function)

Attribute a value to the output
A more complicated function

```matlab
function out = printGeneExpression(exp1Genes, exp1Results, geneOfInterest)
    for (ii = 1:length(exp1Results))
        if (strcmp(exp1Genes{ii}, geneOfInterest))
            fprintf('Gene %s expression %d\n', exp1Genes{ii}, exp1Results(ii));
            out = exp1Results(ii);
        end
    end
end
```

```matlab
>> exp1Genes = {'1L1A', 'NFKBIA', 'BCL2'};
exp1Results = [120, 446, 653];
geneOfInterest = 'BCL2'

>> printGeneExpression(exp1Genes, exp1Results, geneOfInterest)
Gene BCL2 expression 653

ans =
    653
```

1. Save
2. Define your input variables
3. Execute the function
A more complicated function

- Always expect the unexpected

```matlab
function out = printGeneExpression(exp1Genes, exp1Results, geneOfInterest)
    out = 'No gene found';
    for (ii = 1:length(exp1Results))
        if (strcmp(exp1Genes{ii}, geneOfInterest))
            fprintf('Gene %s expression %d\n', exp1Genes{ii}, exp1Results(ii));
            out = exp1Results(ii);
        end
    end
end
```

```matlab
>> printGeneExpression({'1L1A', 'NFKBIA', 'BCL2'}, [120, 446, 653], 'STAT5A')
```
A more complicated function

• Comment!

```matlab
function out = printGeneExpression(exp1Genes, exp1Results, geneOfInterest)
    % Function returns the expression value from a gene of interest
    out = 'No gene found';
    for (ii = 1:length(exp1Results))
        if (strcmp(exp1Genes{ii}, geneOfInterest))
            fprintf('Gene %s expression %d
', exp1Genes{ii}, exp1Results(ii));
            out = exp1Results(ii);
        end
    end
end
```

>> help printGeneExpression
Function returns the expression value from a gene of interest

>> help max
Working with files

• Read a file

```matlab
>> url = 'https://tinyurl.com/yyd24pm8';
>> imageFile = 'example.jpg';
>> urlwrite(url, imageFile);
>> A = imread(imageFile);
>> image(A);
```
function out = printGeneExpressionFromFile(inputFile, geneOfInterest)
    % Function returns the expression value from a gene of interest
    delimiter = '\t'; % tab delimited
    headerLines = 1; % column headers are on line 1

    A = importdata(inputFile, delimiter, headerLines);
    exp1Genes = A.textdata(2:end, 1);
    exp1Results = A.data;

    fprintf('%s loaded. %d genes\n', inputFile, length(A.data));

    % call function to print gene information
    out = printGeneExpression(exp1Genes, exp1Results, geneOfInterest);
Working with files

- Write a file
  - fieldIdentifier = fopen('fileName', 'w');
  - fprintf(fieldIdentifier, 'someText');
  - fclose(fieldIdentifier);

```matlab
function out = printGeneExpression(exp1Genes, exp1Results, geneOfInterest, outputFile)
    % Function returns the expression value from a gene of interest
    fid = fopen(outputFile, 'w');
    out = 'No gene found';
    for (ii = 1:length(exp1Results))
        if (strcmp(exp1Genes{ii}, geneOfInterest))
            fprintf(fid, 'Gene %s expression %d\n', exp1Genes{ii}, exp1Results(ii));
            out = exp1Results(ii);
        end
    end
    fclose(fid);
end
```
function out = printGeneExpressionFromFile(inputFile, outputFile, geneOfInterest)
% Function returns the expression value from a gene of interest
delimiter = '\t'; % tab delimited
headerLines = 1; % column headers are on line 1

A = importdata(inputFile, delimiter, headerLines);
exp1Genes = A.textdata(2:end, 1);
exp1Results = A.data;

fprintf('%s loaded. %d genes\n', inputFile, length(A.data));

% call function to print gene information
out = printGeneExpression(exp1Genes, exp1Results, geneOfInterest, outputFile);

• Write a file
Take home, questions?
Outline of the workshop

Day 1
• Interface
• Command lines and basic syntax
• Variables and operations
• Scripts
• if statements

Day 2
• for and while loops
• More matrices
• Functions
• Files

Day 3
• Plotting
• Introduction to dynamical systems: ODEs