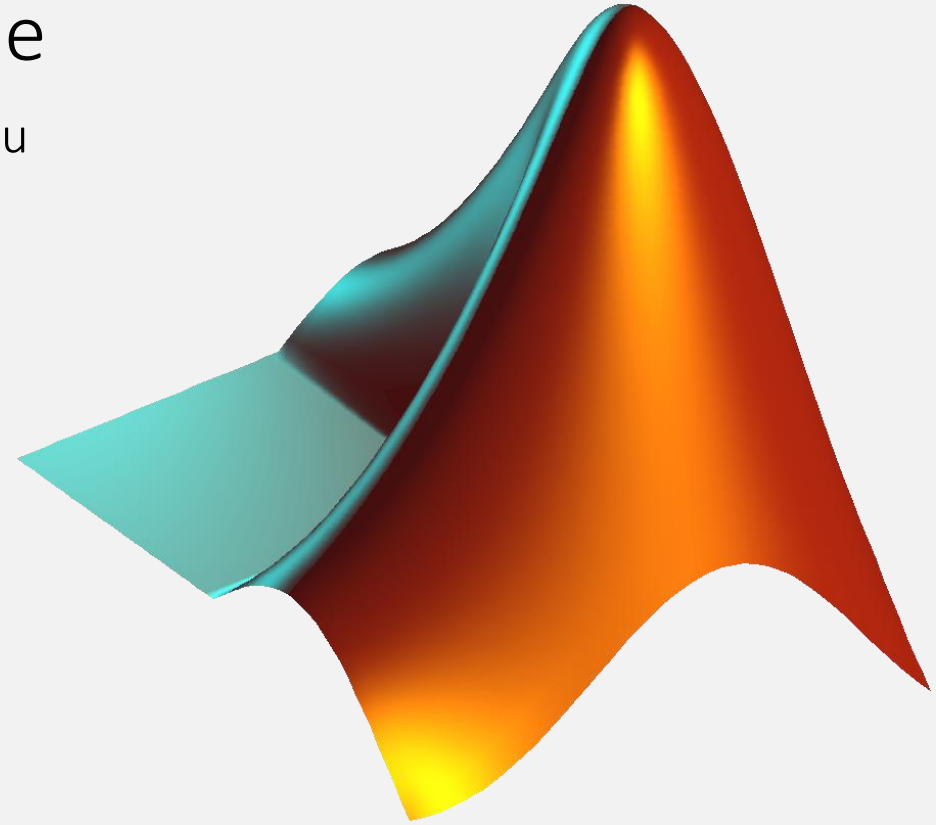
Winter 2022

Intro to MATLAB

Part 3/3

Amandine Gamble

amandinegamble@ucla.edu



UCLA

QCBio Collaboratory

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

Reminders

Reminders

`if` statements

```
exp1 = 400;
exp2 = 500;
% conditional testing
if (exp1 > exp2)
    max = exp1
elseif (exp1 == exp2)
    max = 'both are equal'
else
    max = exp2
end
```

`while` loops

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

- Declare your variables
- Set up the incremenation

`for` loops

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

Functions

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

```
describeSeq('TATCCGCG')
```

- Start the script with the function definition
- Name the file with the function name

Reminders

Read files

```
data = importdata(inputFile, delimiter, headerLines);
```

```
data = importdata('inputFile.txt', '\t', 1);
```

Write files

```
fid = fopen('outputFile.txt', 'w');
fprintf(fid, 'Value I want to save %d\n', exp1Genes{ii}, exp1Results(ii));
fclose(fid);
```

```
fid = fopen(outputFile, 'w');
fprintf(fid, 'Value I want to save %d\n', exp1Genes{ii}, exp1Results(ii));
fclose(fid);
```

Practice

Transform this script into a set of functions we can apply to the larger gene expression dataset, and saving data in .csv

```
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
```

Files you need

- Mitchell_gene_data.txt dataset

Models

- printGenesAboveThreshold.m script
- F_printGeneExpression function
- F_printGeneExpressionFromFile function

1. Import file
2. Extract data of interest
3. Compare each gene expression level to the threshold
4. If it is above, save its name and expression level on a file

Practice

F_printGenesAboveThreshold

```
function out = F_printGenesAboveThreshold(exp1Genes, exp1Results, threshold,
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 (exp1Results(ii) > threshold)
            fprintf(fid, '%s, %d\n', exp1Genes{ii}, exp1Results(ii));
            out = exp1Results(ii);
        end
    end
    fclose(fid);
```

Practice

F_printGenesAboveThresholdFromFile

```
function out = F_printGenesAboveThresholdFromFile(inputFile, outputFile, threshold)
    % 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 = F_printGenesAboveThreshold(exp1Genes, exp1Results, threshold,
outputFile);
```

```
F_printGenesAboveThresholdFromFile('Mitchell_gene_data.txt',
'output_GenesAboveThreshold.txt', 30);
```

Plots

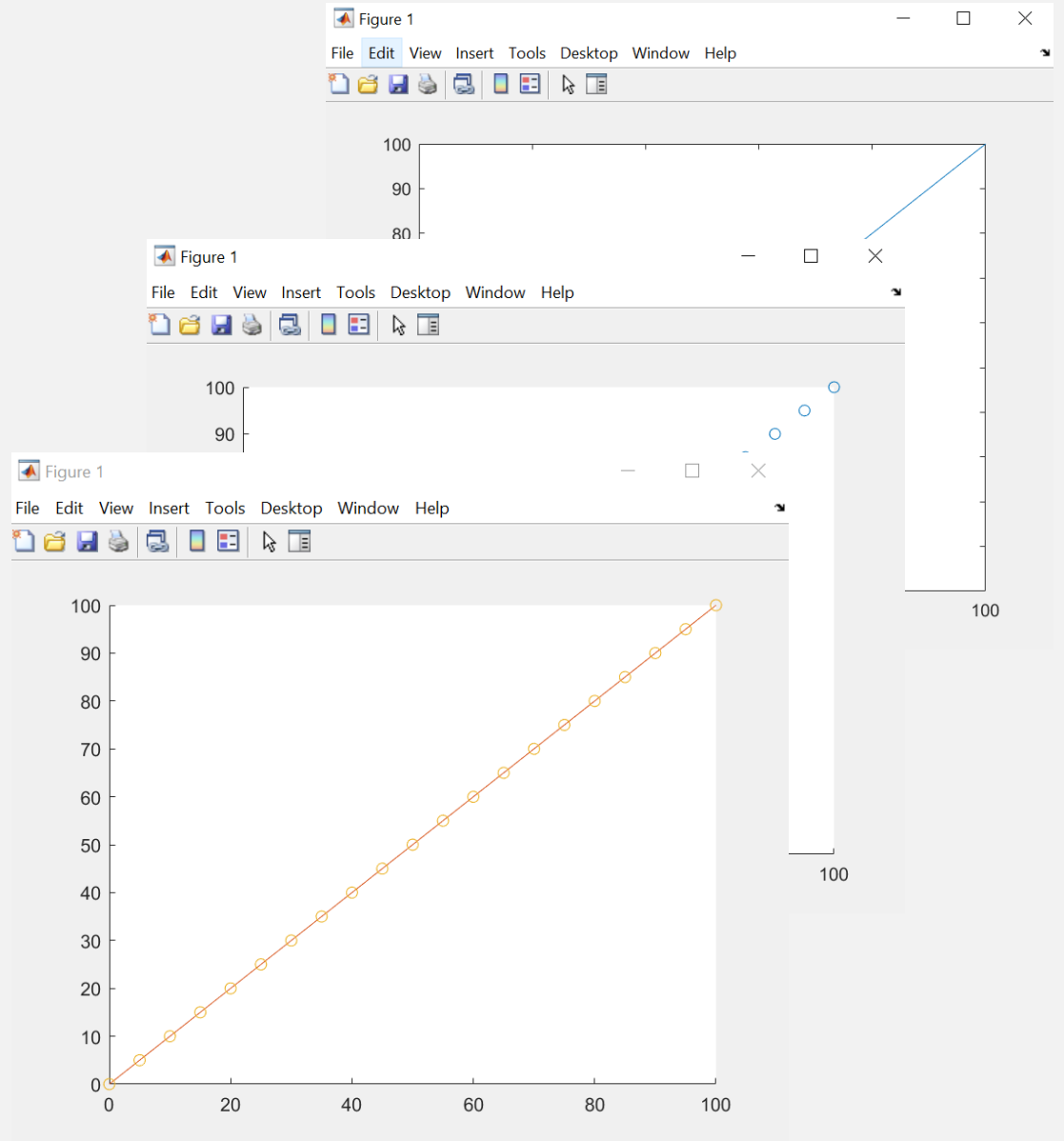
Our first plots

- Create a plot with the function `plot`
- And others…

```
x = [0:5:100];
y = x;
plot(x, y);
```

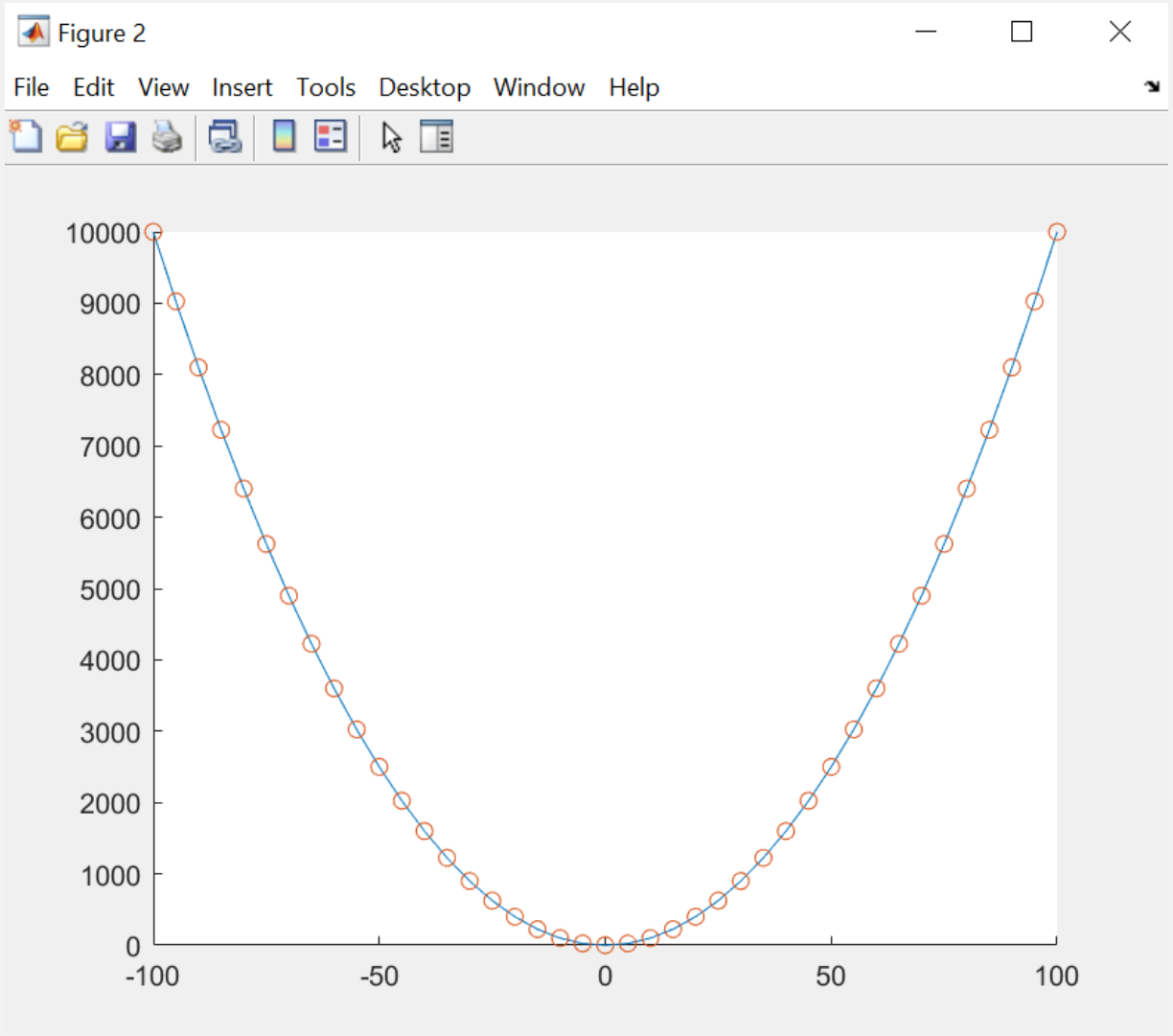
```
x = [0:5:100];
y = x;
plot(x, y);
scatter(x, y);
```

```
x = [0:5:100];
y = x;
hold on;
plot(x, y);
scatter(x, y);
```



Our first plots

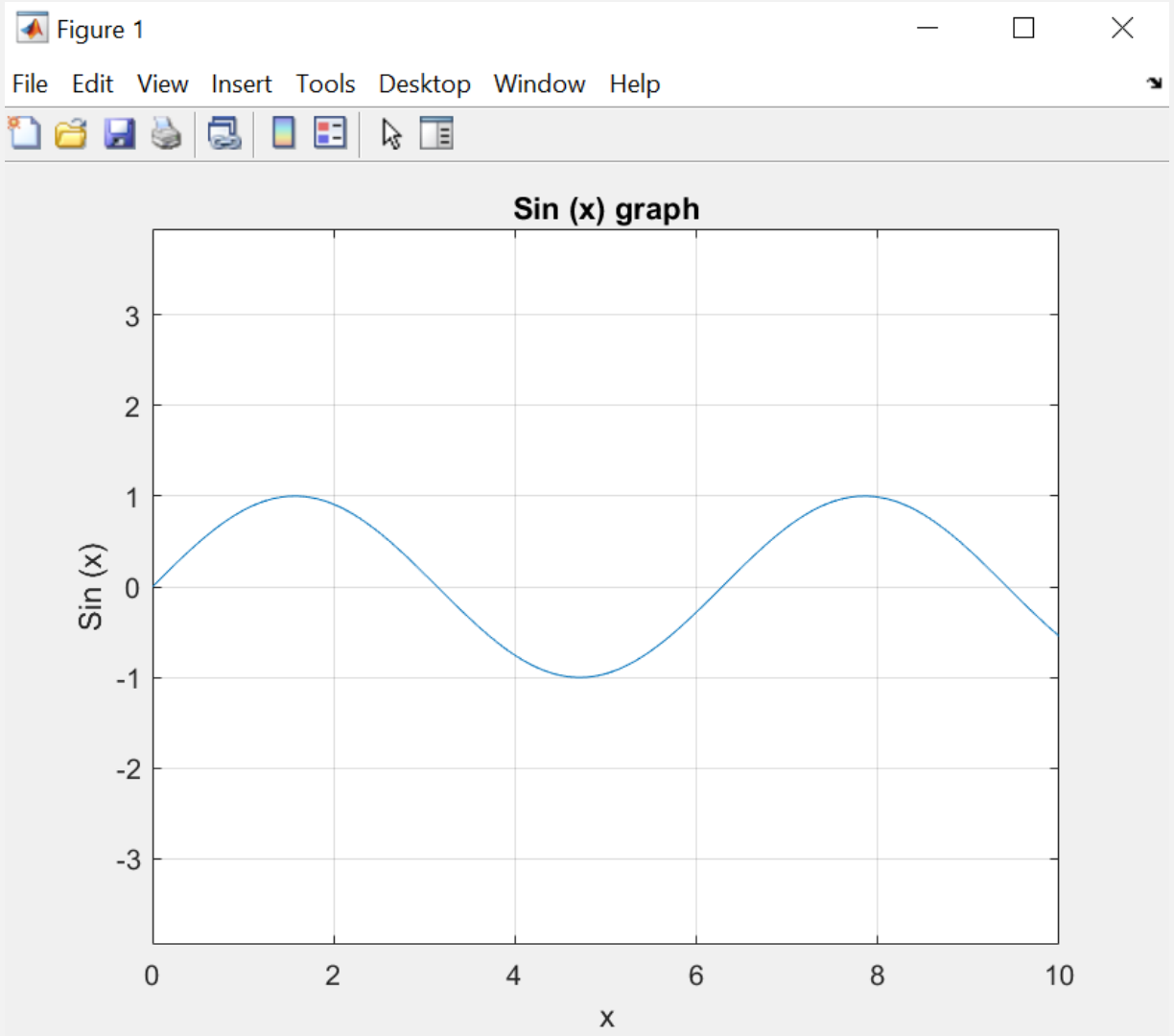
```
figure;
x = [-100:5:100];
y = x.^2;
hold on;
plot(x, y);
scatter(x, y);
```



Our first plots

- Customization

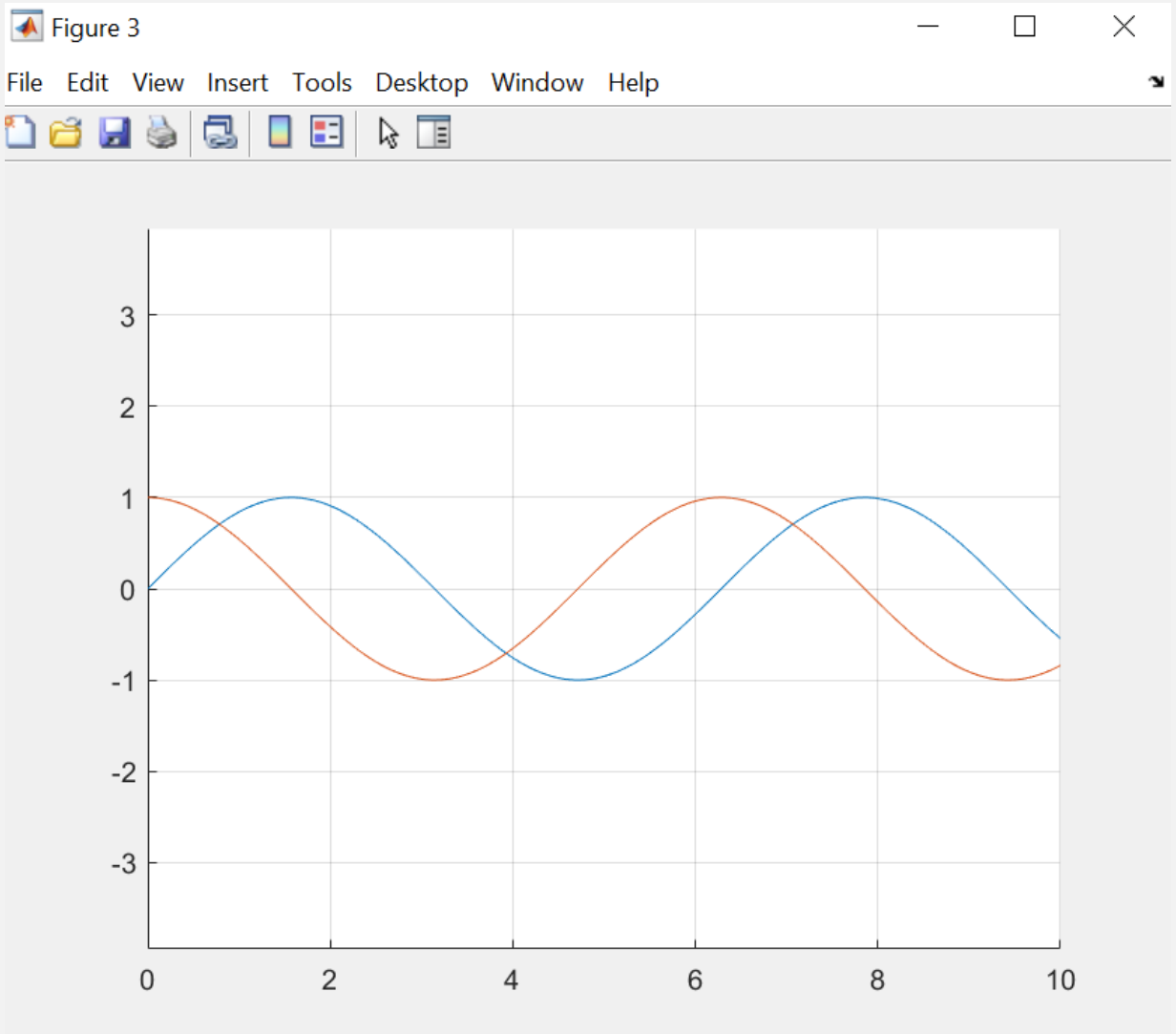
```
figure;
x = [0:0.01:10];
y = sin(x);
plot(x, y);
xlabel('x');
ylabel('Sin (x)');
title('Sin (x) graph');
grid on, axis equal;
```



Our first plots

- Customization

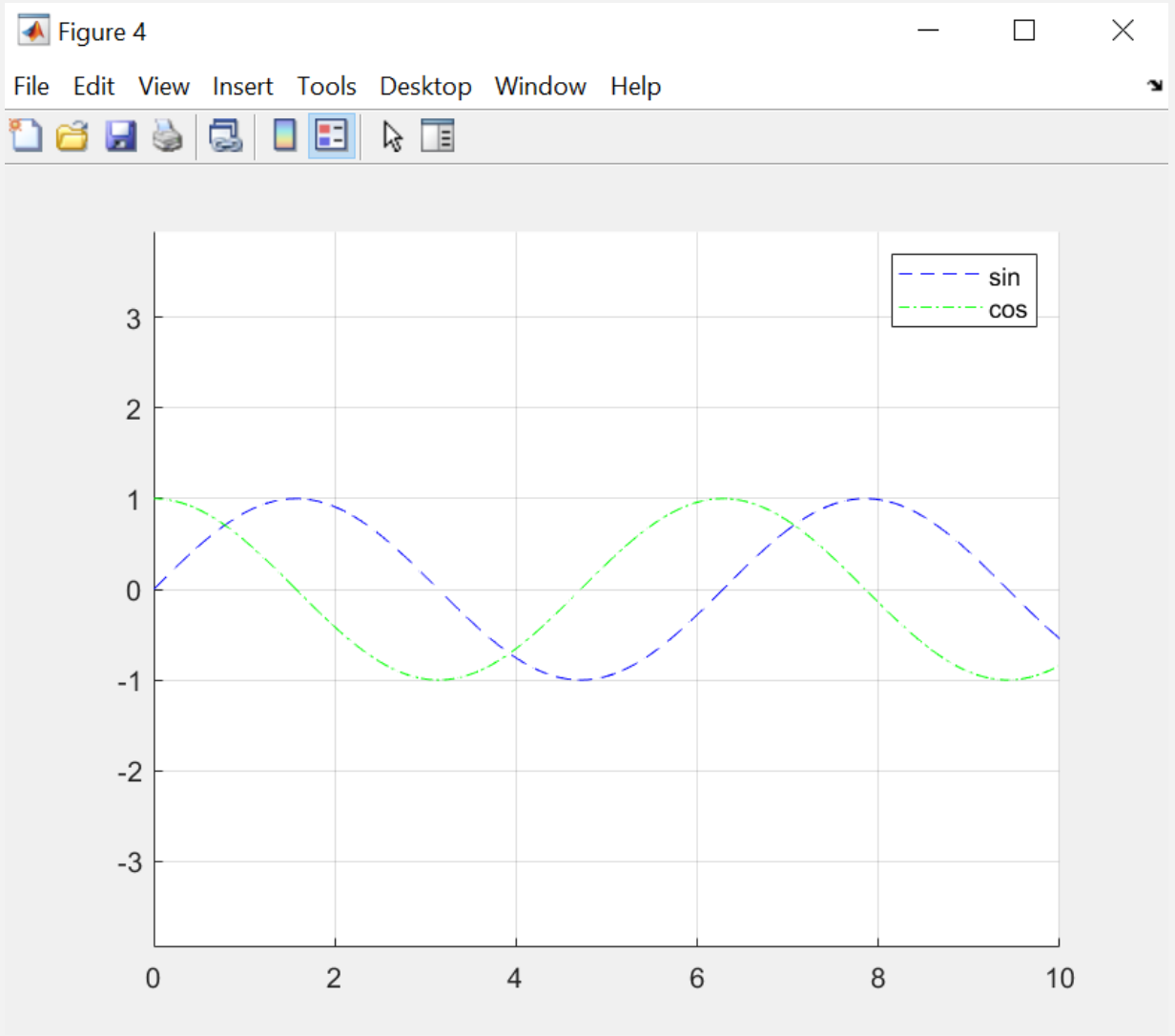
```
figure;
x = [0:0.01:10];
y1 = sin(x);
y2 = cos(x);
hold on;
plot(x, y1);
plot(x, y2);
grid on, axis equal;
```



Our first plots

- Customization

```
figure;
x = [0:0.01:10];
y1 = sin(x);
y2 = cos(x);
hold on;
plot(x, y1, '--b');
plot(x, y2, '-.g');
legend({'sin', 'cos'});
ylim([-3, 3]);
grid on, axis equal;
```

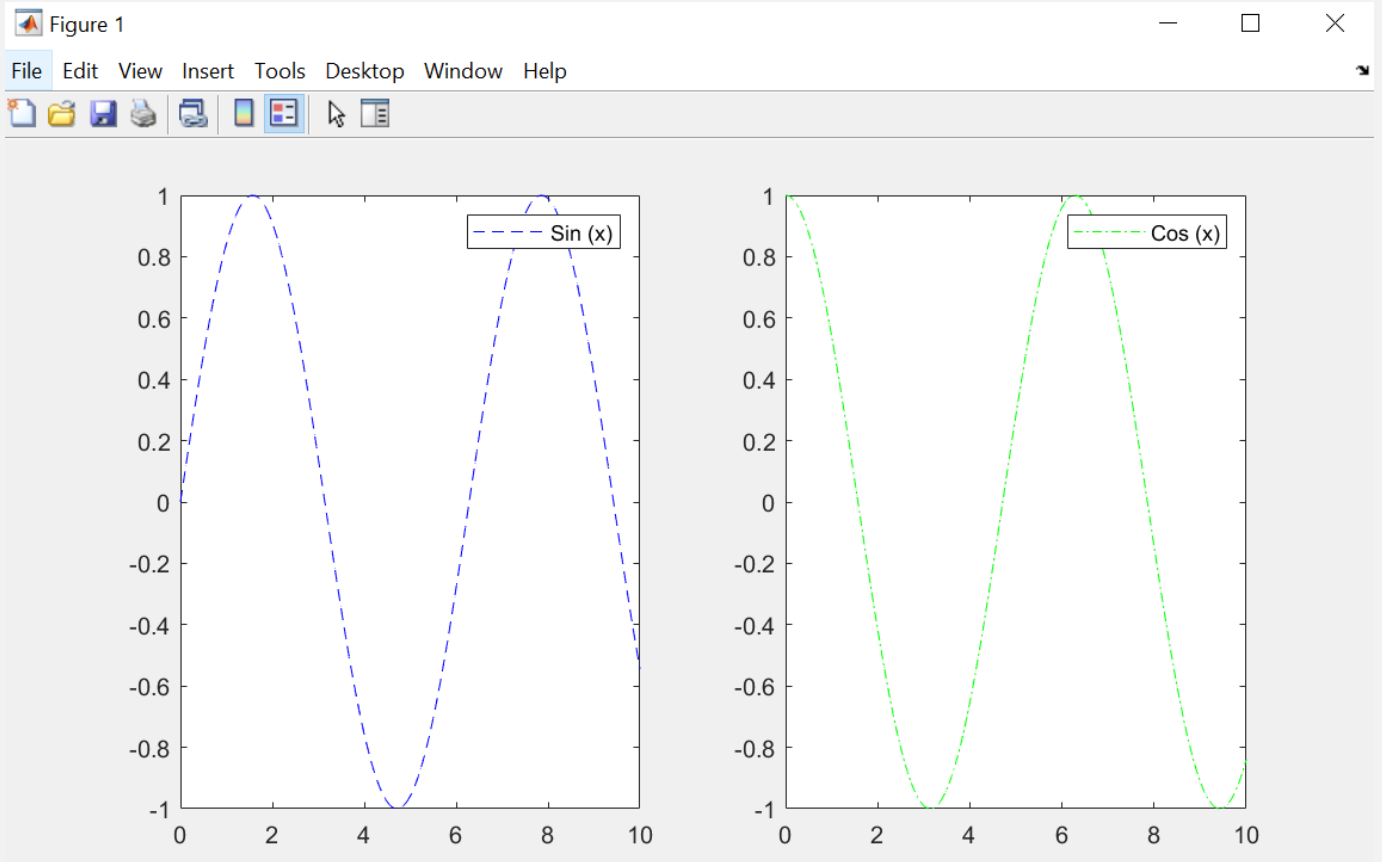


Our first plots

- Subplots

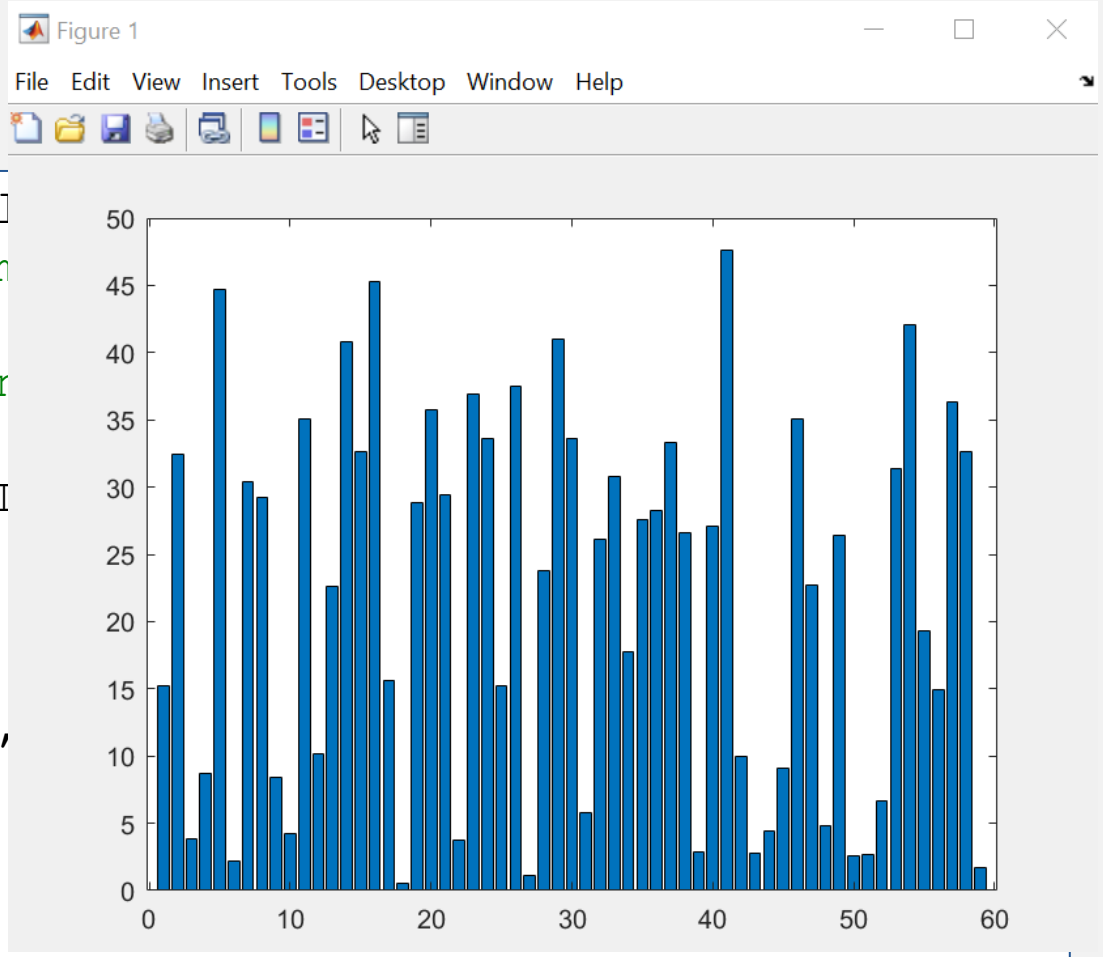
```
figure;
x = [0:0.01:10];
y1 = sin(x);
y2 = cos(x);
subplot(1, 2, 1);
plot(x, y1, '--b');
legend({'Sin (x)'});
subplot(1, 2, 2);
plot(x, y2, '-.g');
legend({'Cos (x)'});
```

```
>> close all
```



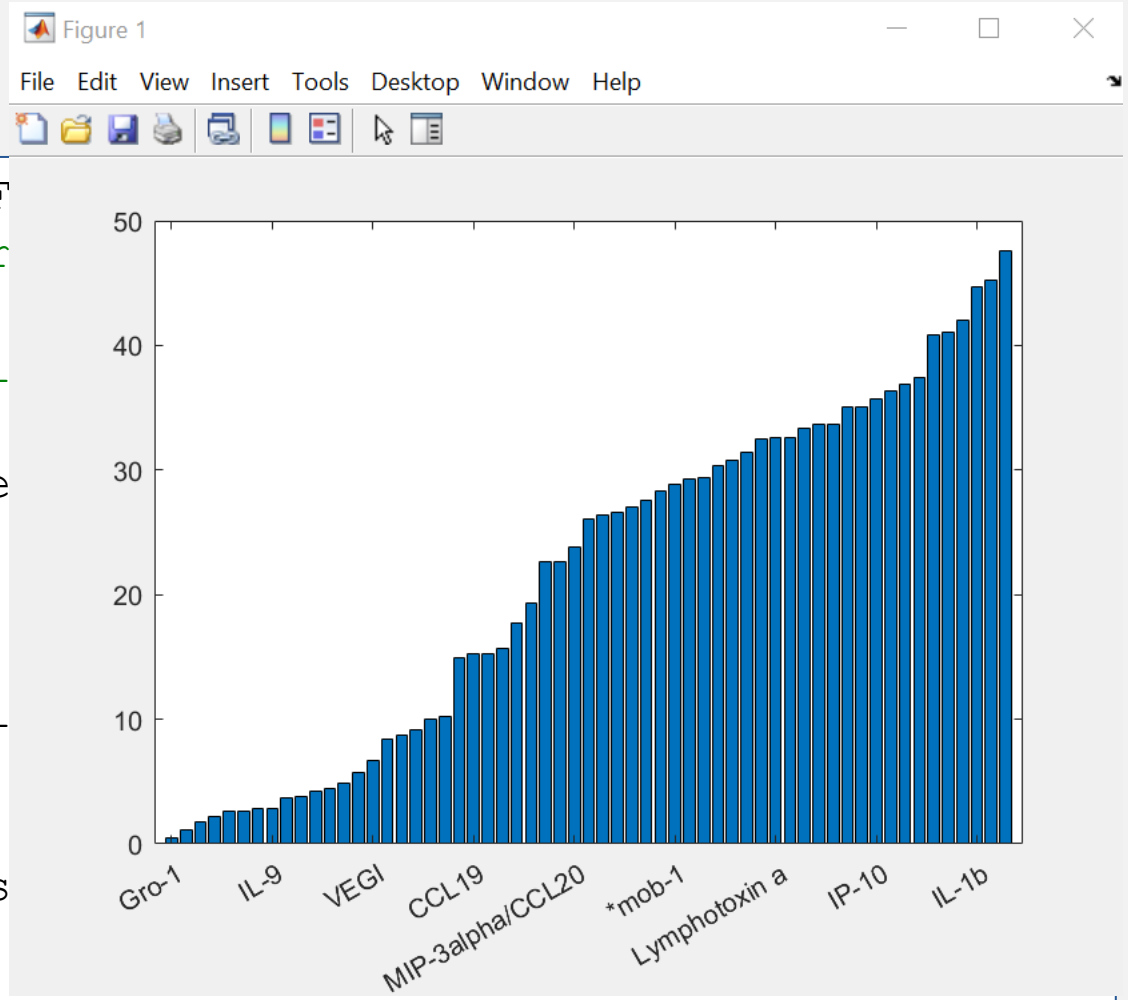
Bar plots

```
function out = F_printGenesAboveThresholdFromFil
    % Function returns the expression value from
    delimiter = '\t'; % tab delimited
    headerLines = 1; % column headers are on lir

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

    fprintf('% s loaded. %d genes\n', inputFile,

    % call function to print gene information
    out = F_printGenesAboveThreshold(exp1Genes,
outputFile);

    % plot data
    bar(exp1Results);
```



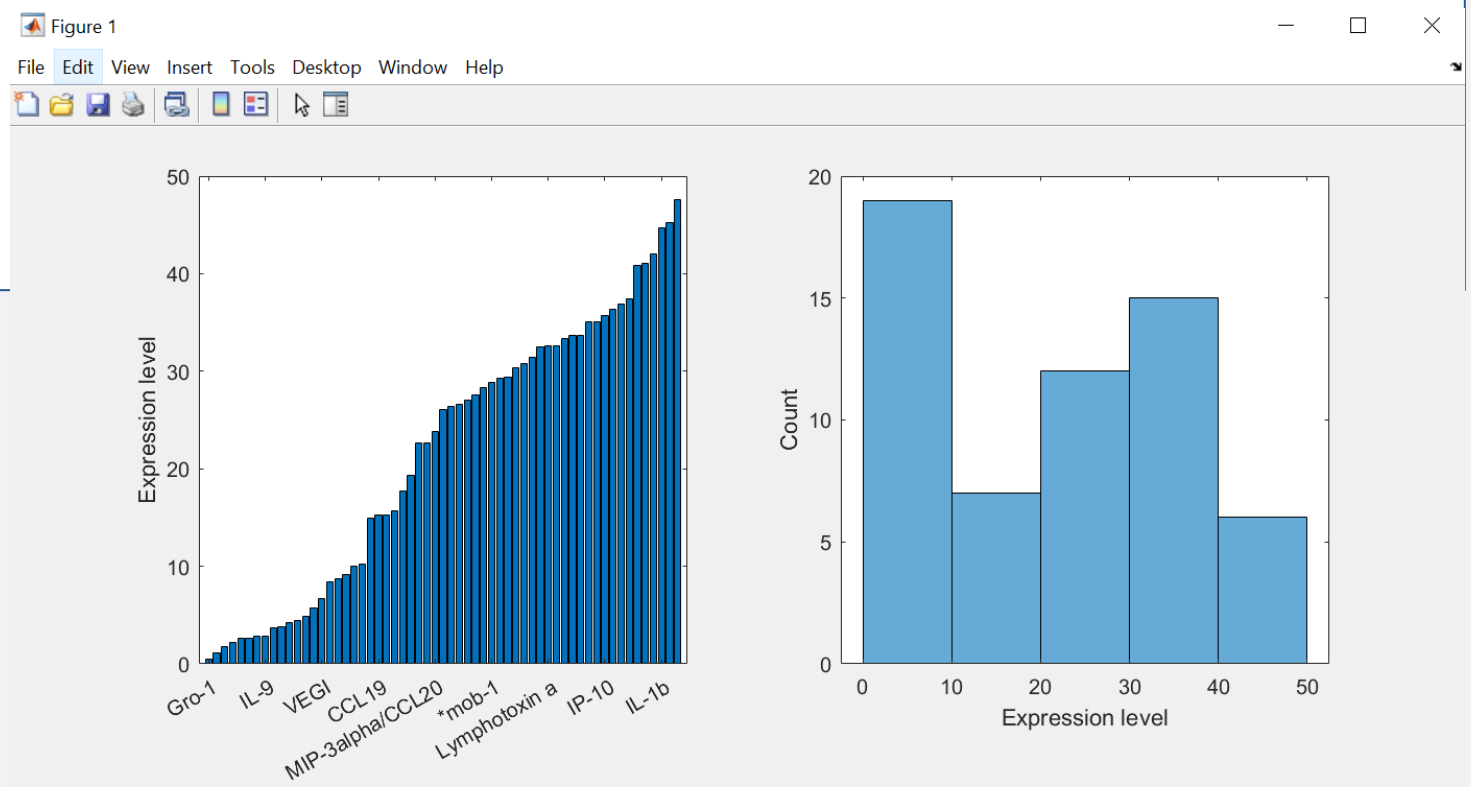
Bar plots

```
function out = F_printGenesAboveThresholdFromF
    % Function returns the expression value fr
    delimiter = '\t'; % tab delimited
    headerLines = 1; % column headers are on l

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

    fprintf('% s loaded. %d genes\n', inputFil

    % call function to print gene information
    out = F_printGenesAboveThreshold(exp1Genes
outputFile);

    % plot data
    [exp1ResultsSorted, kk] = sort(exp1Results);
    namesSorted = exp1Genes(kk);
    bar(exp1ResultsSorted);
    set(gca, 'XTick', 1:7:length(exp1Genes), 'XTickLabel', namesSorted);
```



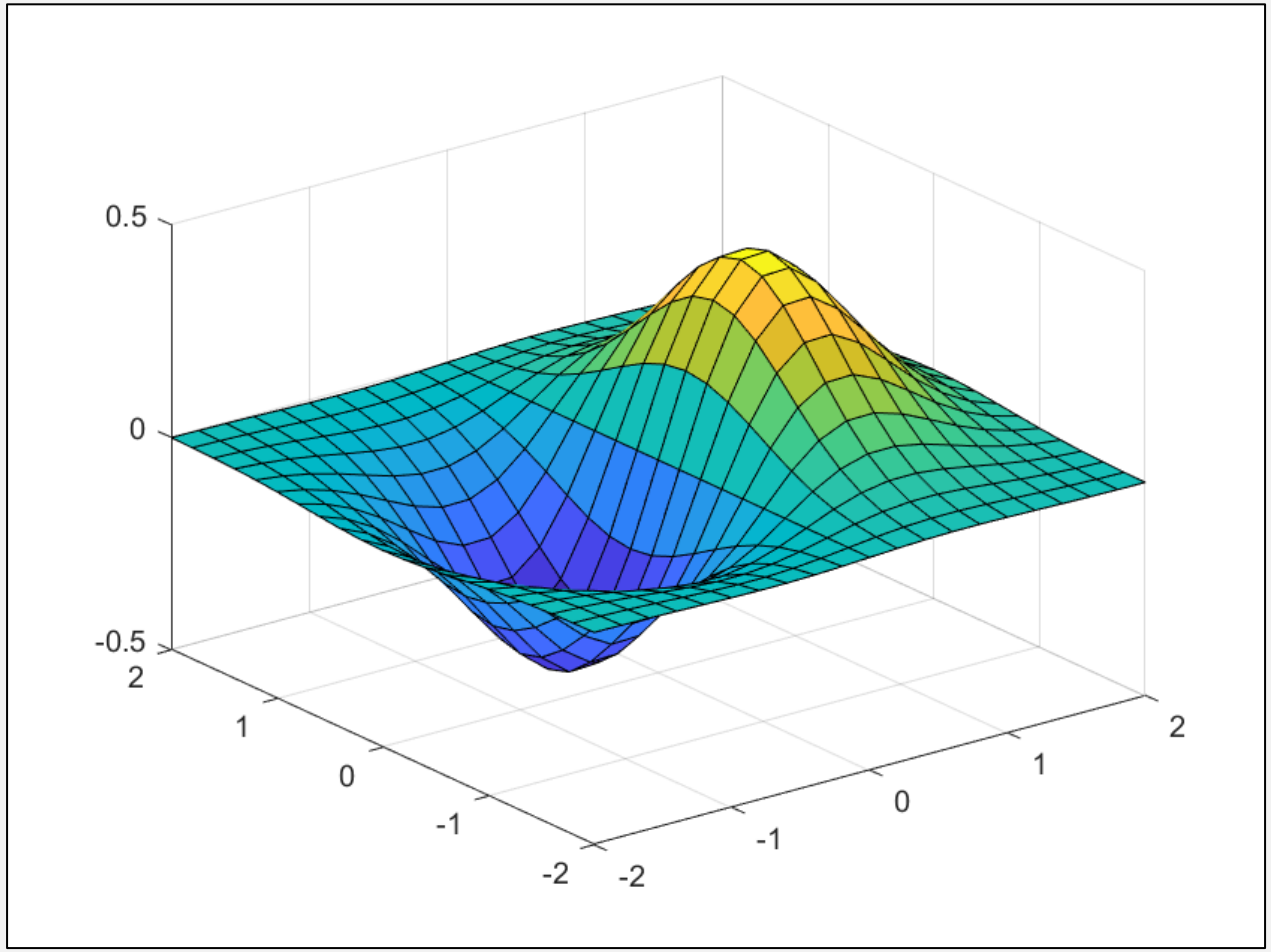
Bar plots

```
% plot data
    [exp1ResultsSorted, kk] = sort(exp1Results);
    namesSorted = exp1Genes(kk);
    subplot(1,2,1);
    bar(exp1ResultsSorted);
    set(gca, 'XTick', 1:7:length(exp1Genes), 'XTickLabel', namesSorted);
    ylabel('Expression level');
    subplot(1,2,2);
    histogram(exp1Results);
    xlabel('Expression level');
    ylabel('Count');
```



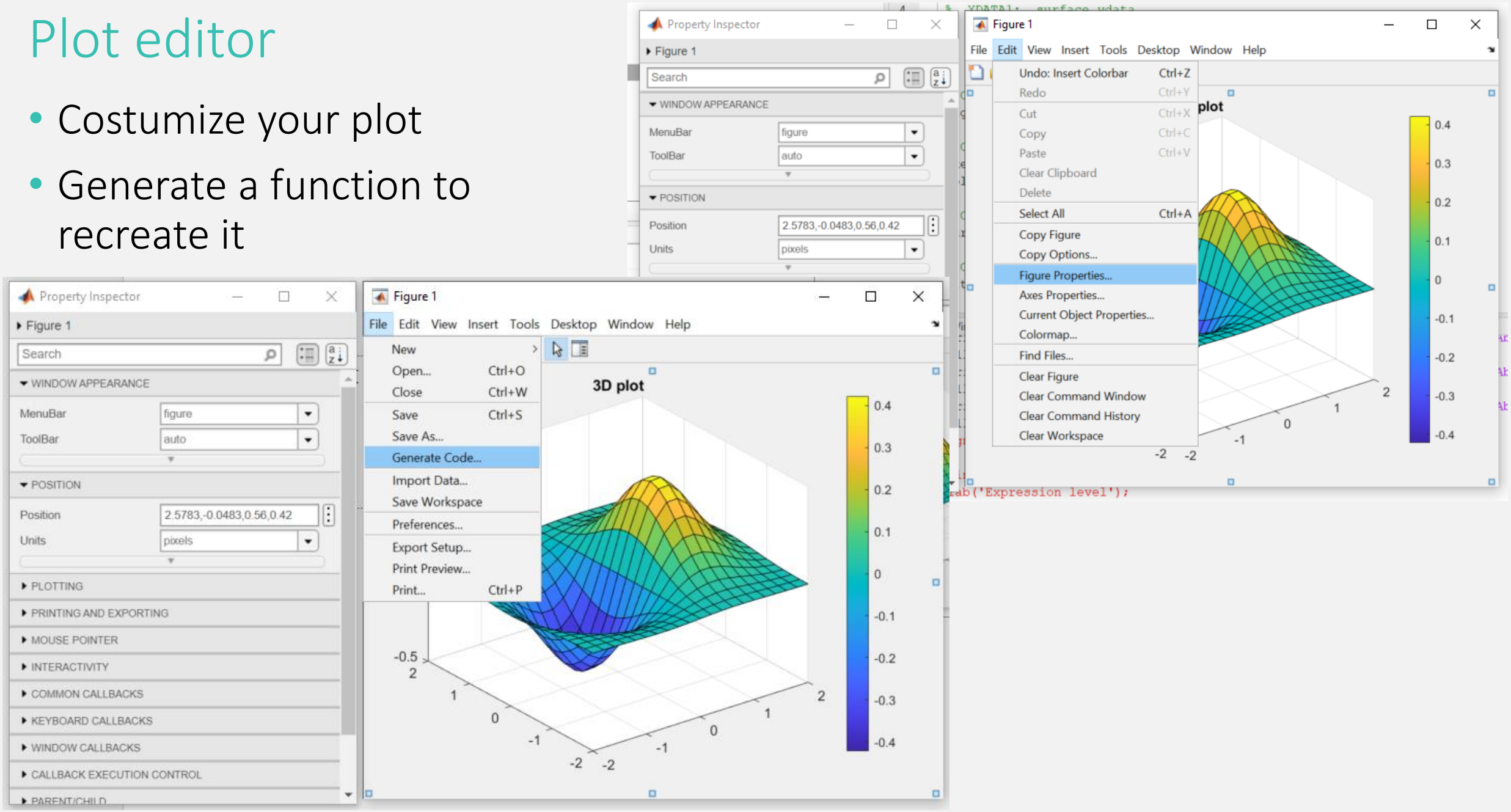
3D plots

```
% generate x and y coordinates
[x, y] = meshgrid(-2:.2:2);

% set z as a function of x and y
z = x.*exp(-x.^2 - y.^2);

% plot
surf(x, y, z);
```



Plot editor

- Costumize your plot
- Generate a function to recreate it



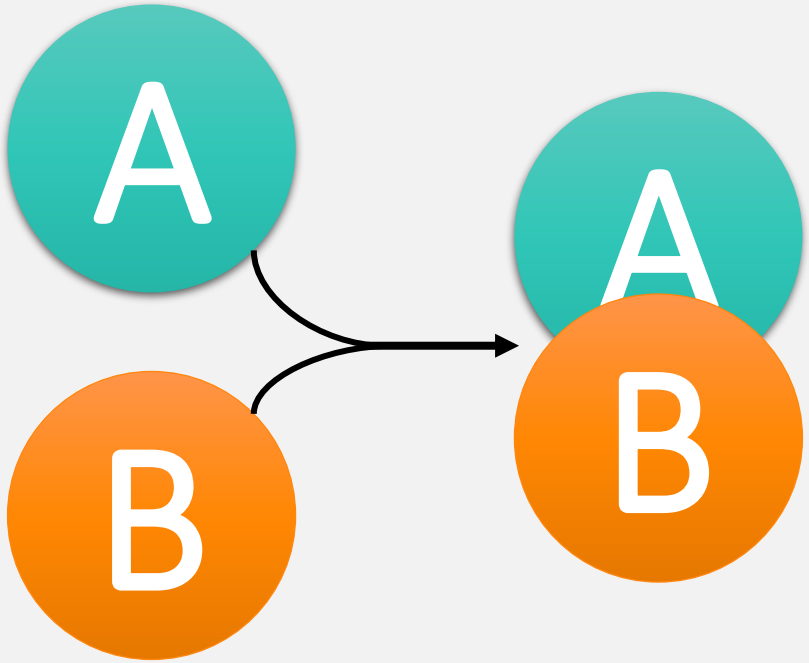
Introduction to dynamical systems: ODEs

Modeling dynamical systems

- Modeling biological processes with equations
- Ordinary differential equations for temporal dynamics
  - Test or generate hypotheses
  - Mechanistic explanation of the results
  - Visualization
  - …

Modeling dynamical systems

- Case study: 2 metabolites (A & B) forming an (hetero)polymer (AB)



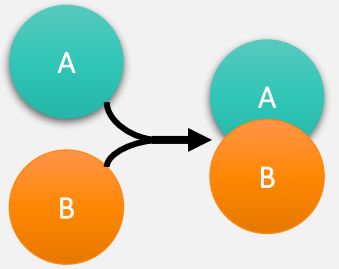
How do the concentrations [A], [B], and [AB] vary over time?

$$\frac{d[A]}{dt} \propto [A][B] \longrightarrow \frac{d[A]}{dt} = -k[A][B]$$

$$\frac{d[A]}{dt} \propto [A][B] \longrightarrow \frac{d[B]}{dt} = -k[A][B]$$

$$\frac{d[AB]}{dt} \propto [A][B] \longrightarrow \frac{d[AB]}{dt} = k[A][B]$$

Modeling dynamical systems



- Case study: 2 metabolites (A & B) forming an (hetero)polymer (AB)

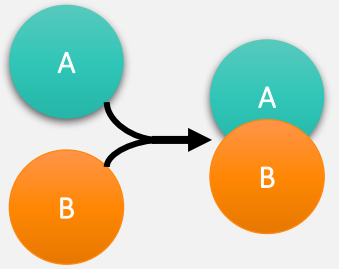
ODE function, defining the rates of changes

```
function delta = F_odeFile(t, x)
% ODE function for a system of differential equations
% t = vector of time points
% x = vector of metabolites

% parameters
k = 7;

% rate = k[A][B]
bindingRate = k * x(1) * x(2);

% store the rate of change for each metabolite
delta = zeros(length(x), 1);

% d[A]/dt = - rate
delta(1) = -bindingRate;
% d[B]/dt = - rate
delta(2) = -bindingRate;
% d[AB]/dt = - rate
delta(3) = bindingRate;

end
```

System of differential equations

$$\begin{cases} \frac{d[A]}{dt} = -k[A][B] \\ \frac{d[B]}{dt} = -k[A][B] \\ \frac{d[AB]}{dt} = k[A][B] \end{cases}$$

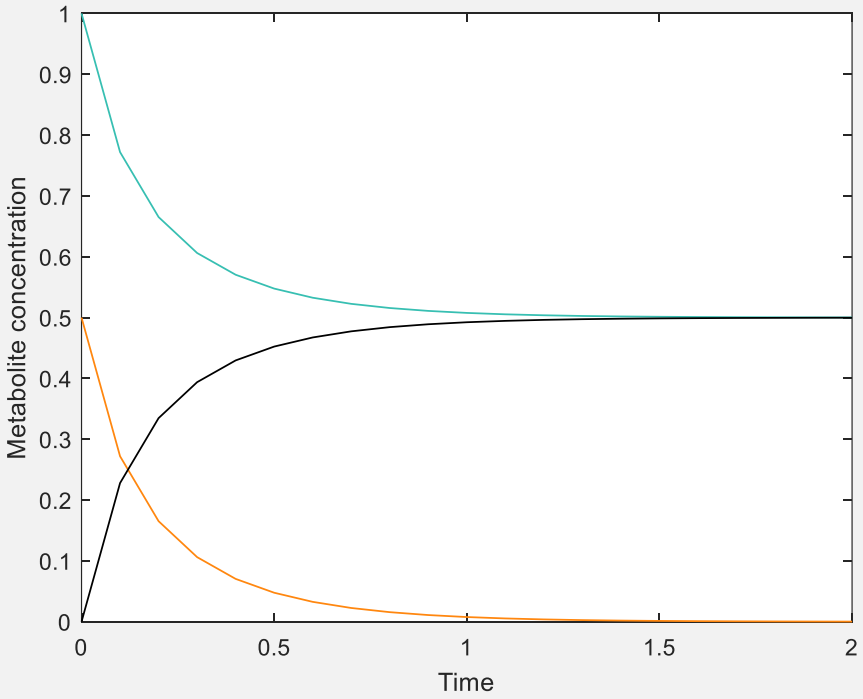
Modeling dynamical systems



- Case study: 2 metabolites (A & B) forming an (hetero)polymer (AB)

ODE solver

```
% ODE solver: ode15s(ODE function, time span, initial conditions)
[t, y] = ode15s(@F_odeFile, [0:0.1:2], [1,0.5,0]);
plot(t, y);
```



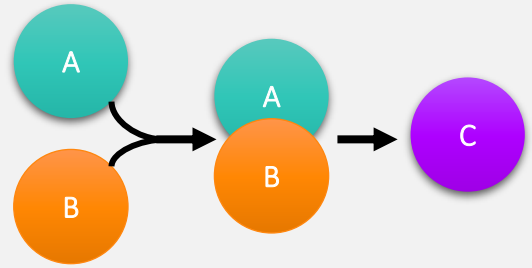
System of differential equations

$$\frac{d[A]}{dt} = -k[A][B]$$

$$\frac{d[B]}{dt} = -k[A][B]$$

$$\frac{d[AB]}{dt} = k[A][B]$$

Modeling dynamical systems



- Case study: fancier version

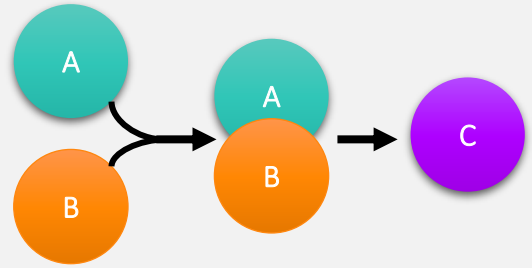
ODE function, defining the rates of changes

```
function delta = F_odeFile2(t, x)
% ODE function for a system of differential equations
% t = vector of time points
% x = vector of metabolites

% parameters
k = 7;
k2 = 0.7;

% rate = k[A][B]
bindingRate = k * x(1) * x(2);
conversionRate = k2 * x(3);

% store the rate of change for each metabolite
delta = zeros(length(x), 1);

% d[A]/dt = - rate
delta(1) = -bindingRate;
% d[B]/dt = - rate
delta(2) = -bindingRate;
% d[AB]/dt = - rate
delta(3) = bindingRate - conversionRate;
% d[C]/dt = - rate
delta(4) = conversionRate;

end
```

System of differential equations

$$\begin{cases} \frac{d[A]}{dt} = -k[A][B] \\ \frac{d[B]}{dt} = -k[A][B] \\ \frac{d[AB]}{dt} = k[A][B] - k_2[AB] \\ \frac{d[C]}{dt} = k_2[AB] \end{cases}$$

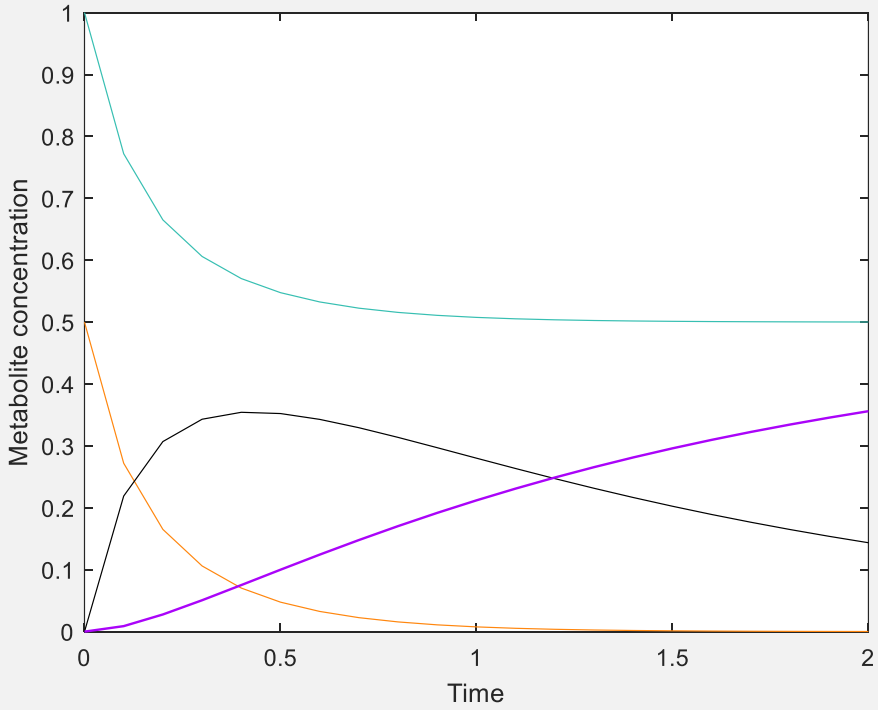
Modeling dynamical systems



- Case study: fancier version

ODE solver

```
% ODE solver: ode15s(ODE function, time span, initial conditions)
[t, y] = ode15s(@F_odeFile2, [0:0.1:2], [1,0.5,0,0]);
plot(t, y);
```



System of differential equations

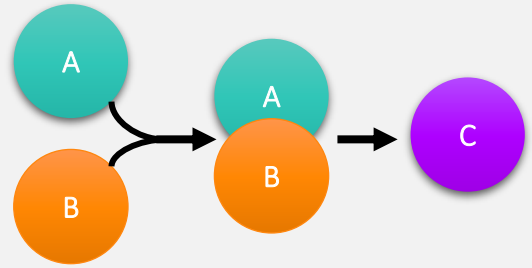
$$\frac{d[A]}{dt} = -k[A][B]$$

$$\frac{d[B]}{dt} = -k[A][B]$$

$$\frac{d[AB]}{dt} = k[A][B] - k_2[AB]$$

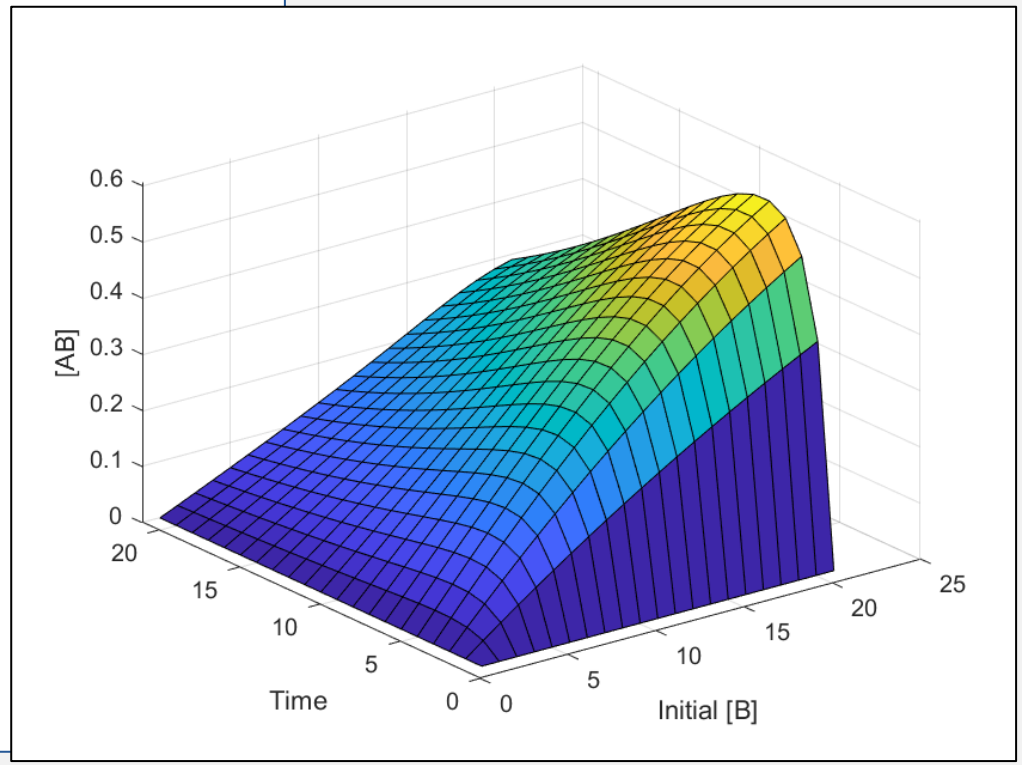
$$\frac{d[C]}{dt} = k_2[AB]$$

Modeling dynamical systems



- Exploring the parameter space

```
matrixOfTimeCourses = zeros(length([0:0.1:2]),
length([0:0.05:1]));
currentIndex = 1;
for (ii = [0:0.05:1])
    [t, y] = ode15s(@F_odeFile2, [0:0.1:2], [1,ii,0,0]);

    matrixOfTimeCourses(:,currentIndex) = y(:,3);
    currentIndex = currentIndex + 1;

    %figure;
    %plot(t, y);
end

figure;
surf(matrixOfTimeCourses);
xlabel('Initial [B]')
ylabel('Time');
zlabel('[AB]');
```



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