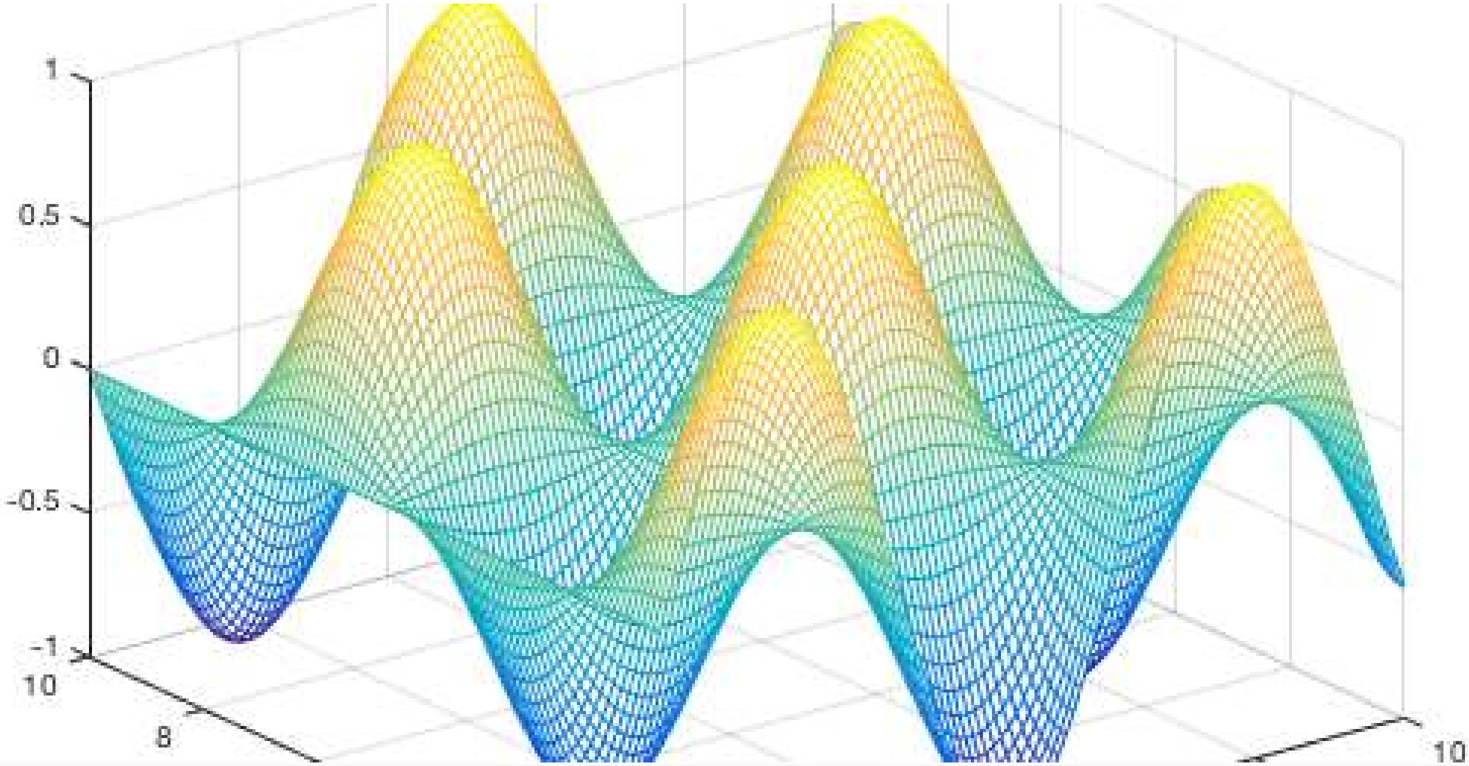
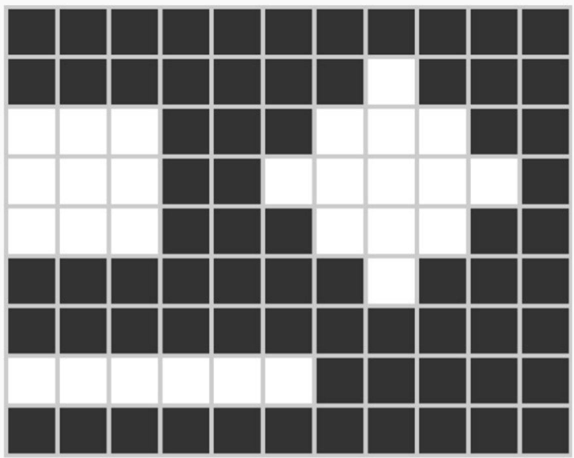
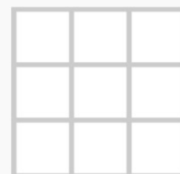


MATLAB Lecture 8: Plots and curve fitting

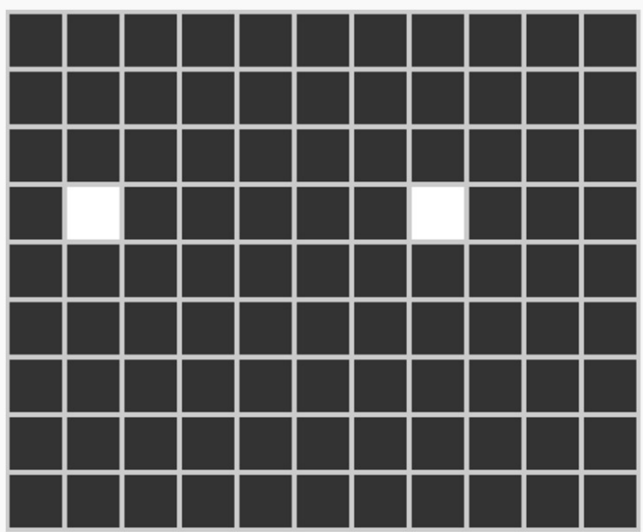
MCDB/BCHM 4312/5312

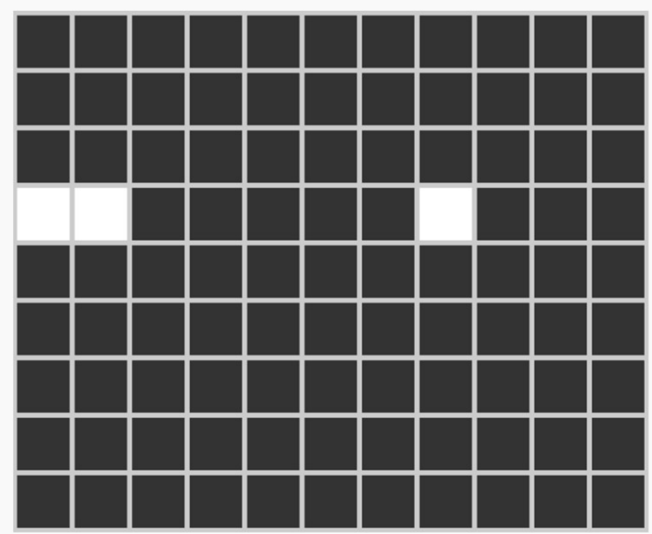
Homework 6: Erosion in MATLAB





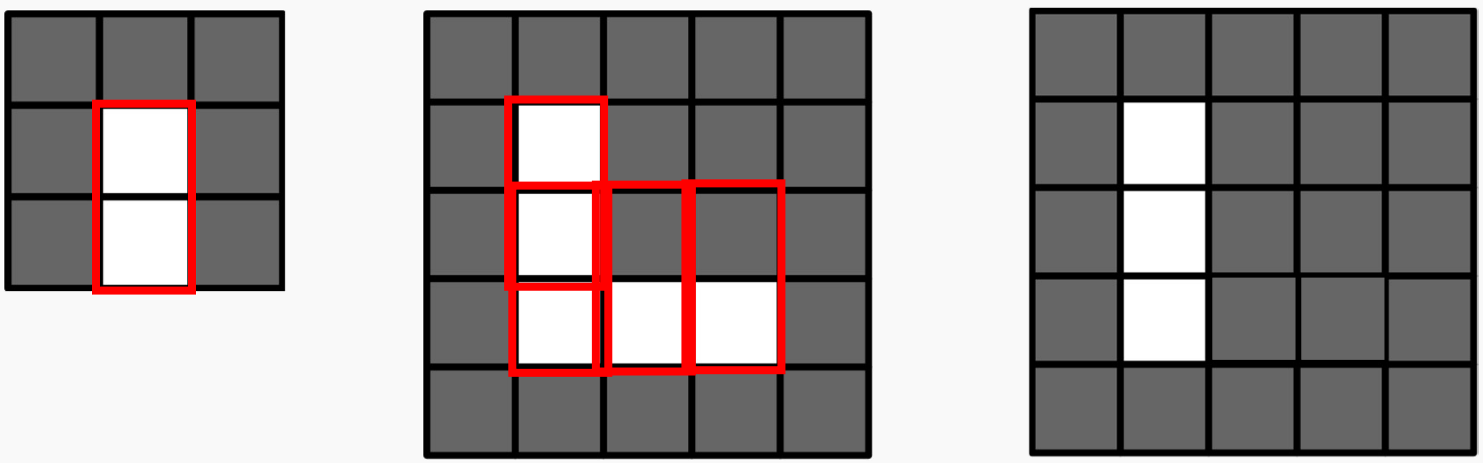
Both these answers are correct





This is how MATLAB treats the edges

Lecture 5: Morphological opening

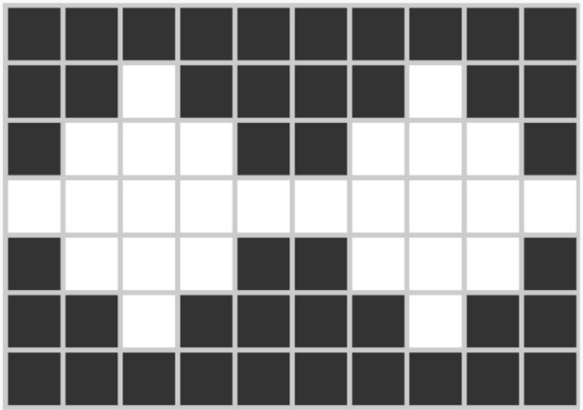


Slide the structuring element over all the foreground pixels

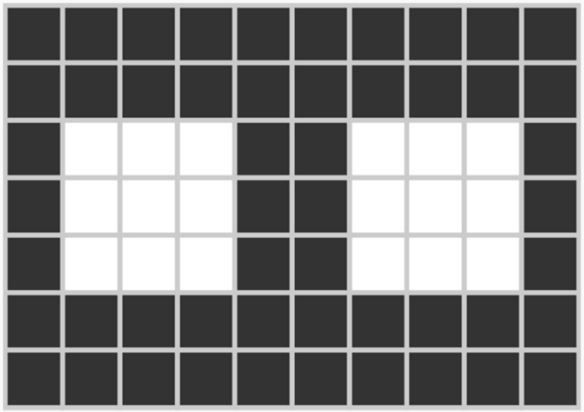
All pixels in the image which fits the structuring element will be kept

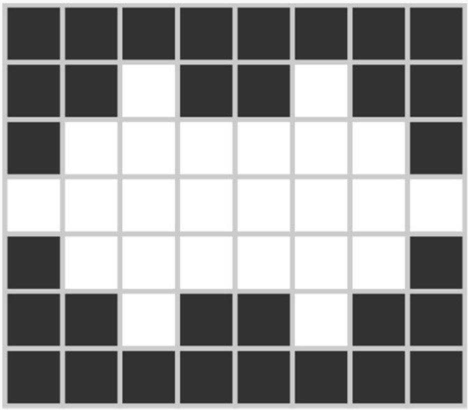
Homework 6: Morphological opening

- Actually quite difficult to separate objects with an opening
  - Only works if objects were minimally connected to being with



You could separate these using a [1 1] structuring element





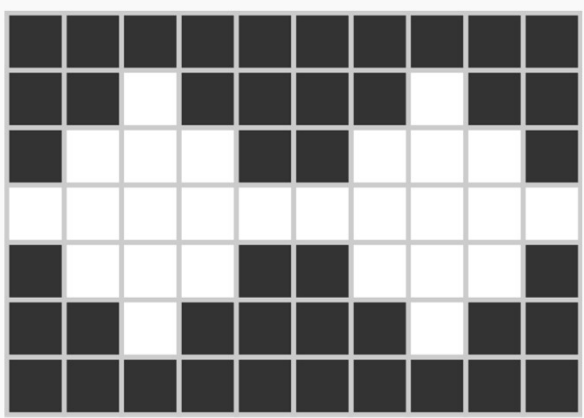
Would be much harder to separate these objects

Watershed is a better approach for separating objects

Example code to test

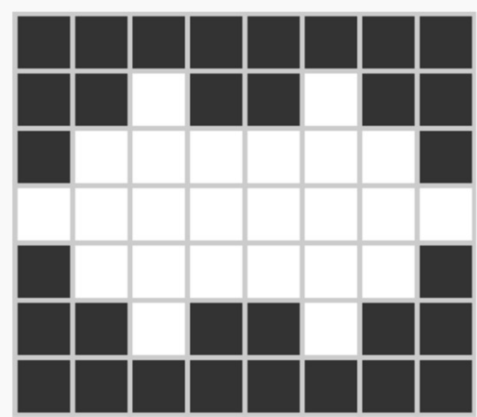
```
M = [0 0 1 0 0 0 0 1 0 0;
     0 1 1 1 0 0 1 1 1 0;
     1 1 1 1 1 1 1 1 1 1;
     0 1 1 1 0 0 1 1 1 0;
     0 0 1 0 0 0 0 1 0 0];
```

```
imopen(M, ones(2))
```



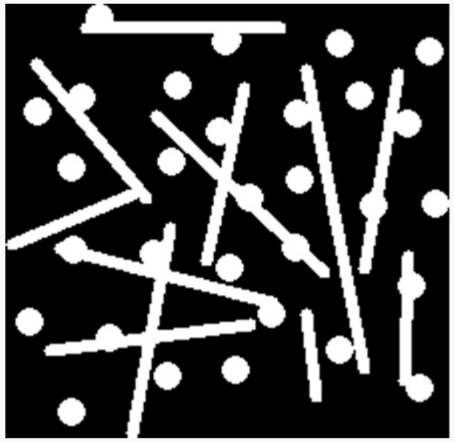
```
M = [0 0 1 0 0 1 0 0;
     0 1 1 1 1 1 1 0;
     1 1 1 1 1 1 1 1;
     0 1 1 1 1 1 1 0;
     0 0 1 0 0 1 0 0];
```

```
imopen(M, ones(2))
```



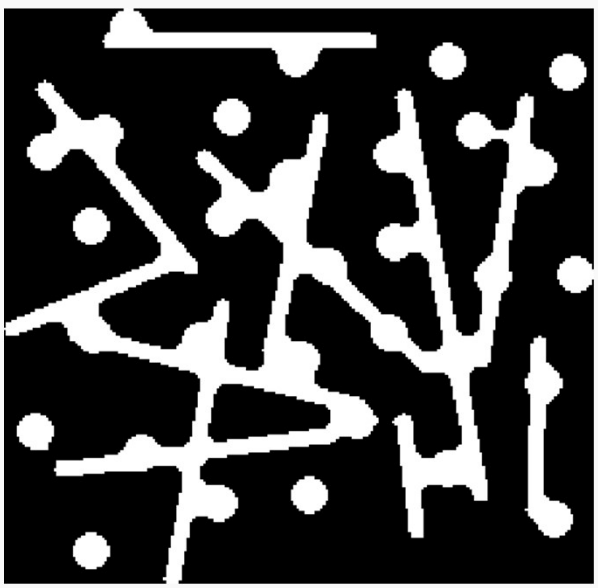
What is the result of the morphological opening operation?

Image

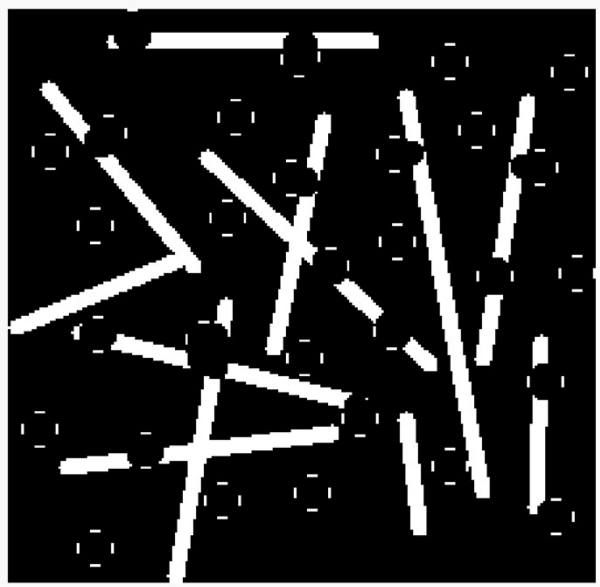


Structuring element

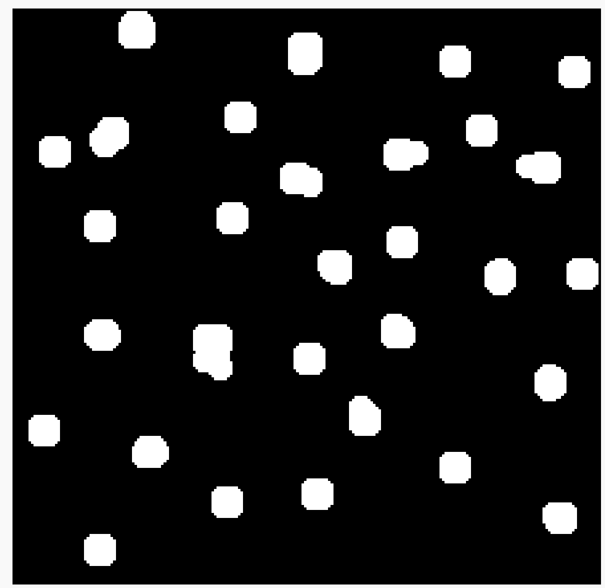
(A)



(B)



(C)



Useful applications of opening

- Removing unwanted shapes
  - Tidying up boundaries of segmented cells
  - Removing contamination/unwanted objects

- Removing small objects – opening removes objects smaller than the structuring element

Learning Goals

- How to plot data in MATLAB
- Curve fitting
- Interpolating data

How to plot your data

Basic line plot syntax:

```
plot(xdata, ydata)
```

xdata and ydata must have the same lengths

Creating a range of values

- Remember the colon operator can provide you with a vector of numbers

```
xdata = 1:10
```

- You can also specify a step size

```
xdata = 1:2:10
```

Note: the colon operator will not include the last point if it does not line up with the step size

```
xdata = [1, 3, 5, 7, 9]
```

Another way of creating a range of values

Syntax:

xdata = linspace(start, end, number of points)

- linspace = **lin**ear **space**

- Unlike the colon operator, `linspace` will include both the start and end values

Example

Write a script to do the following:

1. Create a linearly spaced array ranging from [0, 10] with 100 points called `xdata`

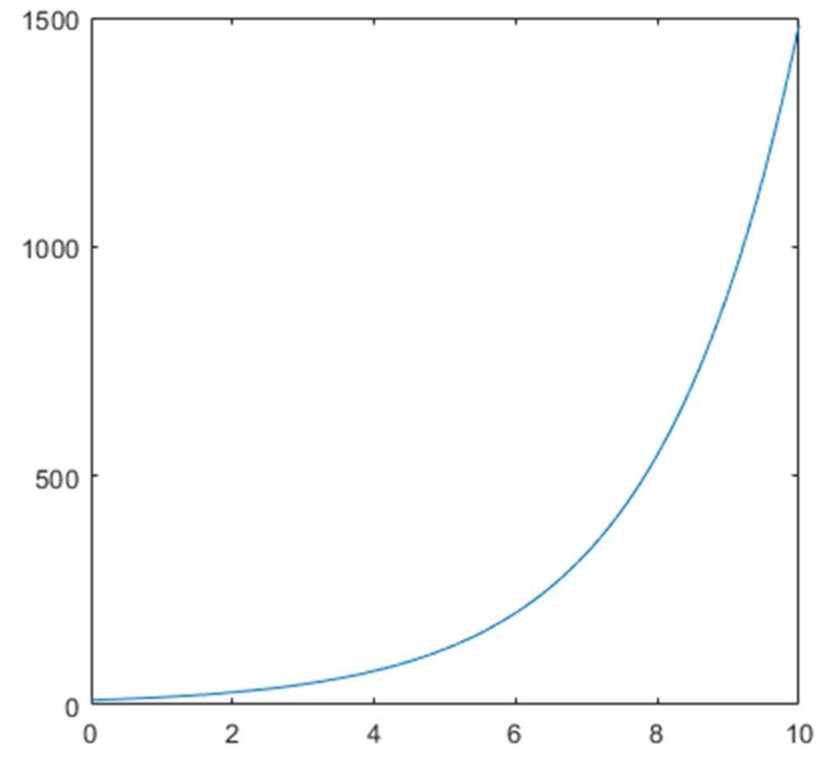
2. Evaluate the following function over `xdata`

$$y = 10e^{0.5x}$$

3. Plot the data using a line plot

```
plot(xdata, ydata)
```

```
xx = linspace(0, 10, 100);
yy = 10 * exp(0.5 * xx);
plot(xx, yy)
```



Labeling your axes

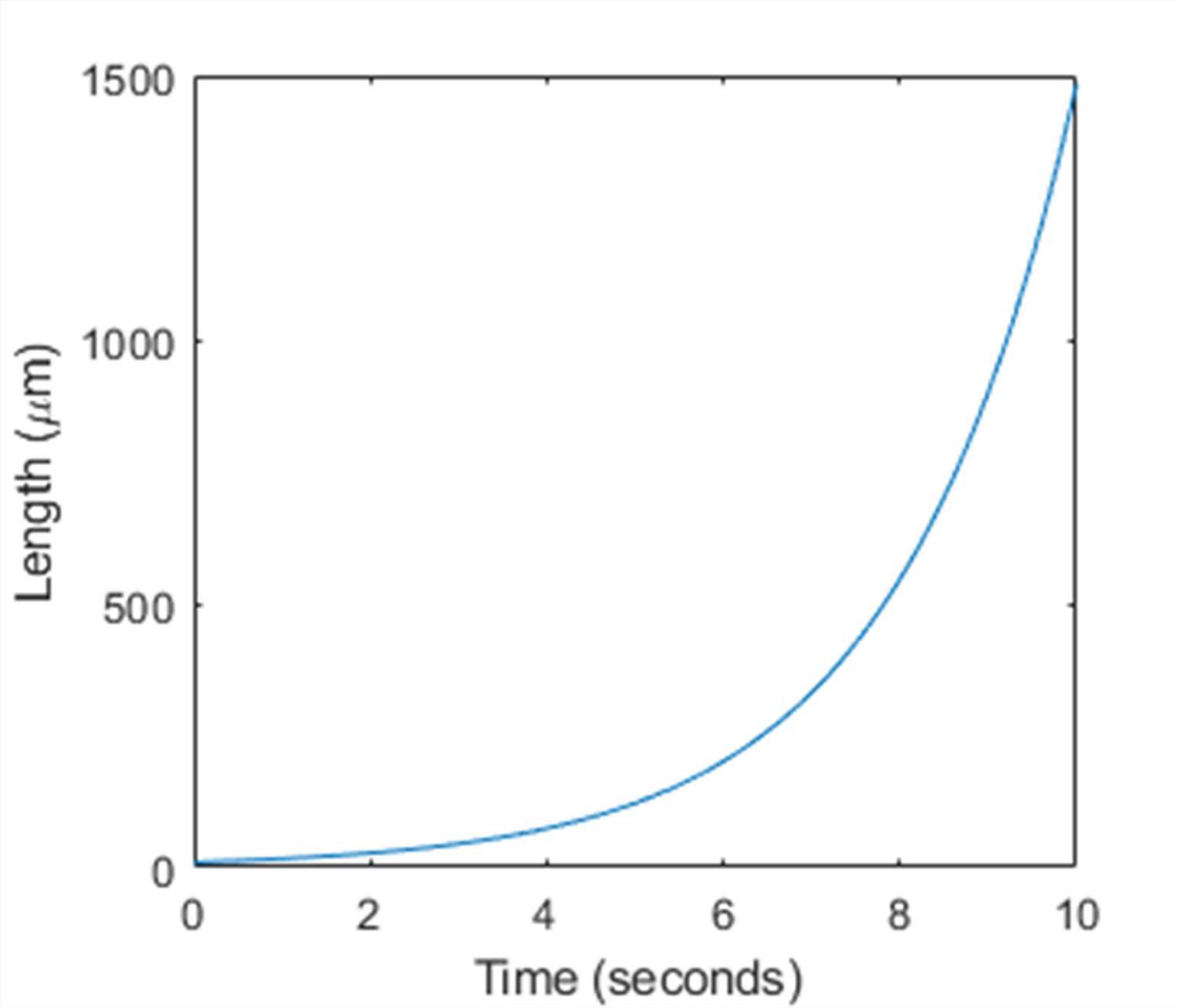
- Hopefully everyone recognized that the equation was for exponential growth
    - E.g. bacteria length, number of cells

- Label the axes as:

```
xlabel('Time (seconds)')

ylabel('Length (\mum)')
```

Labeled axes



Axes and title labels

- By default, MATLAB will interpret TeX/LaTeX style commands

- Examples for Greek letters:

```
\mu
\theta
\Theta
```

- Examples for subscript and superscripts: _ and ^

- Full list: https://www.mathworks.com/help/matlab/creating_plots/greek-letters-and-special-characters-in-graph-text.html

Axes and title labels

- To disable this behavior, specify additional arguments:

```
ylabel('Length \mum', 'Interpreter', 'none')
```

Titles

- You can add titles using the command

```
title('Bacterial growth')
```

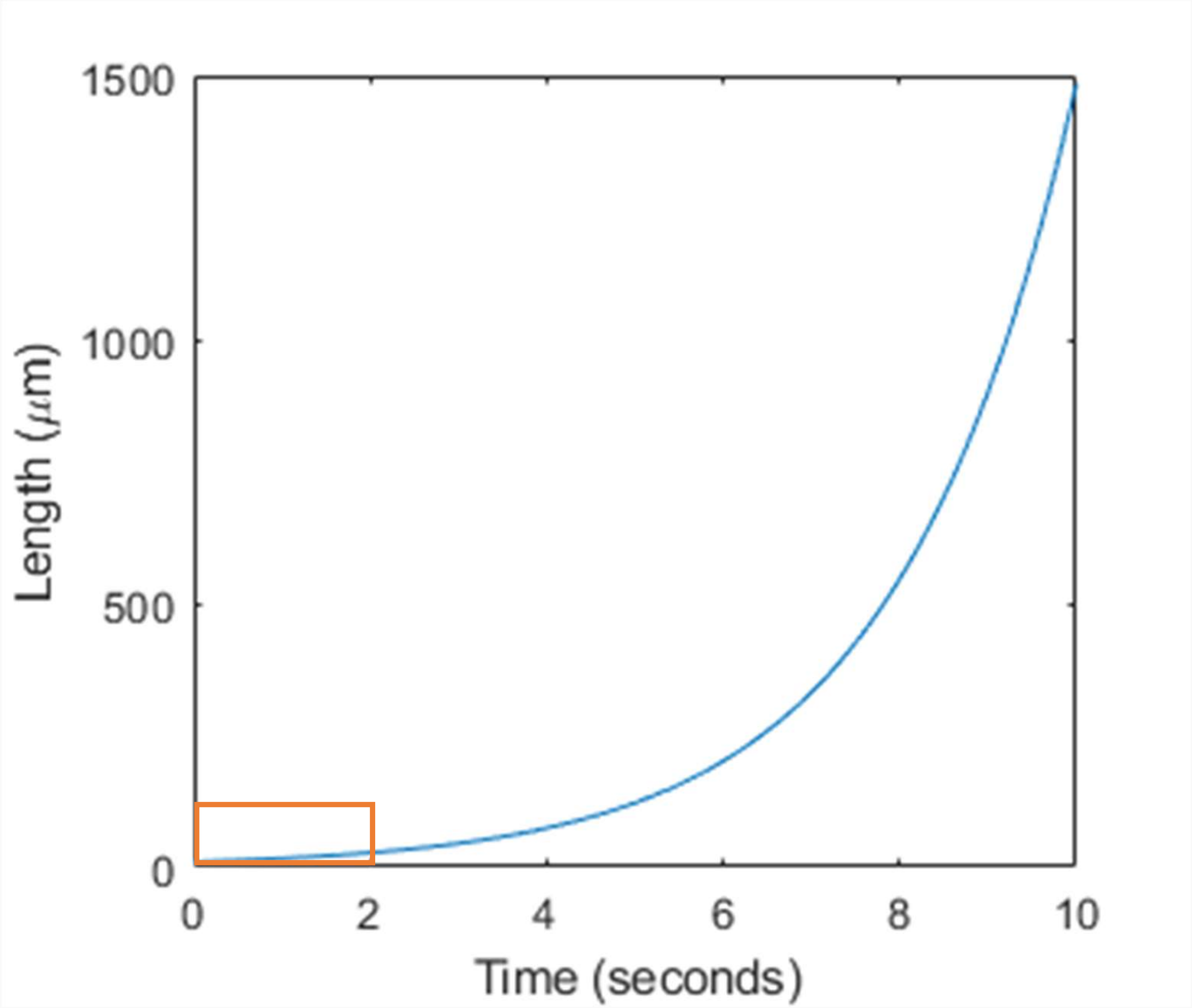
One small detail

- Most figure commands (e.g. `title`, `xlabel`, `plot`) will operate on the last selected figure

- Last selected figure =
  - Last figure that was created, or
  - Last figure window that you selected

- To select a figure programmatically, you can use figure(number)
- E.g. figure(1) selects the window titled Figure 1

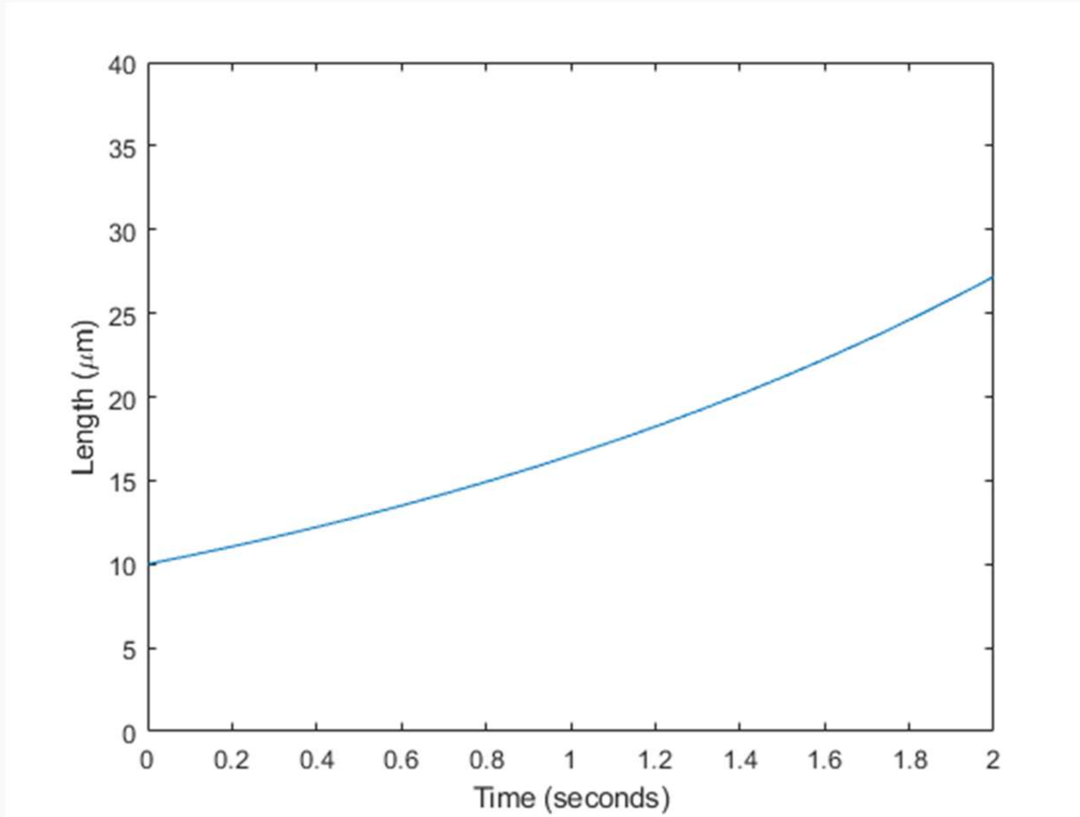
Specifying limits for the axes



Specifying limits for the axes

After plotting:

```
xlim([0, 2])
ylim([0, 40])
```



Example code

```
xx = linspace(0, 10, 100);
yy = 10 * exp(0.5 * xx);

plot(xx, yy)
xlabel('Time (seconds)');
ylabel('Length (\mum)');
xlim([0, 2])
ylim([0, 40])
```

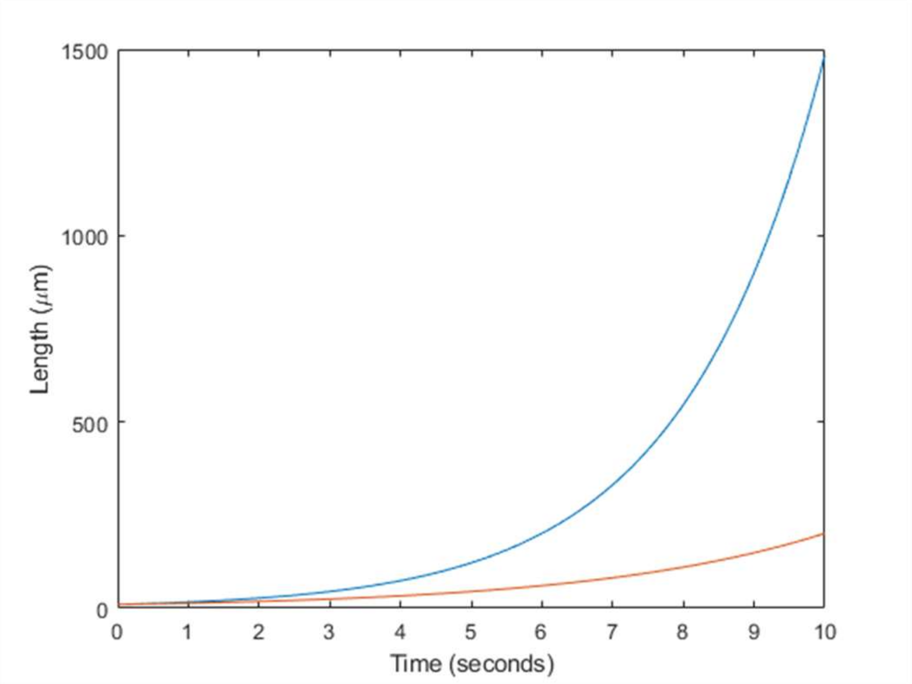
Plotting multiple plots on the same axes

```
plot(xdata1, ydata1, xdata2, ydata2, …
                xdataN, ydataN)
```

Modify your script to plot

$$y = 10e^{0.5x}$$

$$y2 = 10e^{0.3x}$$



Example code

```
xx = linspace(0, 10, 100);
yy = 10 * exp(0.5 * xx);

yy2 = 10 * exp(0.3 * xx);

plot(xx, yy, xx, yy2)
xlabel('Time (seconds)');
ylabel('Length (\mum)');
xlim([0, 2])
ylim([0, 40])
```

Line specifications

- You can add additional options to change line color and line style etc

```
plot(xx, yy, '--r', xx, yy2, 'ok')
```

Line Specification	Type
`'-'`	Solid line (default)
`'--'`	Dashed line
`'.'`	Dotted line
`'.-'`	Dot-dashed line
`'o'`	Circles

Color Specification	Type
`'r'`	Red
`'g'`	Green
`'b'`	Blue
`'y'`	Yellow
`'k'`	Black

* Look up documentation for more examples

Plotting multiple plots on the same axes

- Use 'hold on' and 'hold off'

- Useful when you do not have the data ahead of time

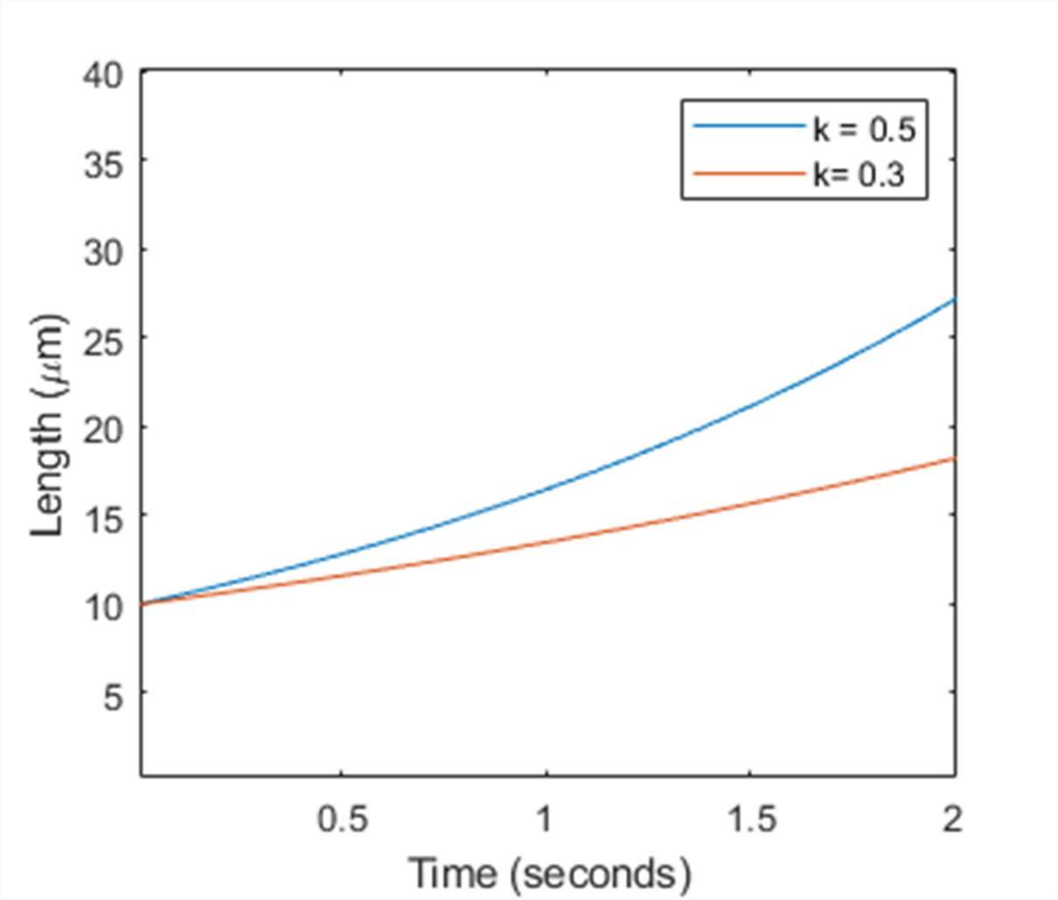
```
figure;
plot(xx, yy)
hold on
plot(xx, yy2)
hold off
```

Remember hold off otherwise MATLAB will keep drawing on the same axes

Inserting legends

Example: `legend('k = 0.5', 'k= 0.3')`

Legend for first plot   Legend for second plot



Semilog plots

- Syntax:

```
semilogx(xdata, ydata)

semilogy(xdata, ydata)
```

Semilog plots only one of the axes using a log scale

Which of the following semilog plots would show the equation below as a straight line?

$$y = 10e^{0.5x}$$

```
(A)  semilogx(xdata, ydata)

(B)  semilogy(xdata, ydata)
```

Example

- Modify your script to make create a new figure window:

```
figure;
```

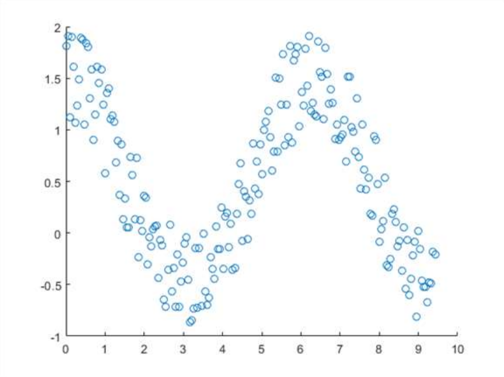
- Plot the data using an appropriate semilog plot so the graph appears as a straight line

```
semilogy(xx, yy)
```

Other plots you might find useful

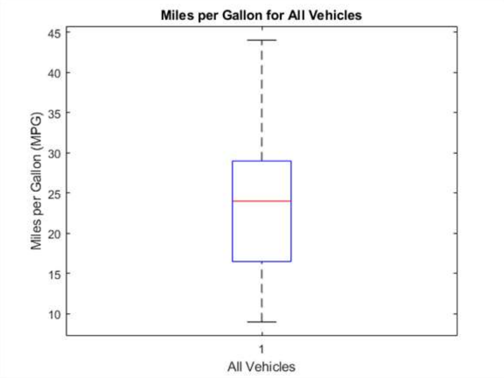
Scatter plot

```
scatter(xdata, ydata)
```



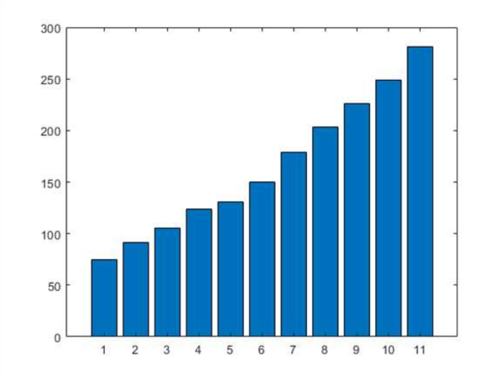
Box-and-whisker plot

```
boxplot(xdata)
```



Bar plot

```
bar(xdata, ydata)
```



Curve fitting

- Curve fitting is a very important tool in data analysis

- It allows you to fit data to a **model** (a mathematical function that describes the data)

Examples of when you might want to do this in cell microscopy

- Measuring the growth rate of cells – exponential growth
- Measuring size of particles/puncta – Gaussian

There will be some homework questions on these

Basic fitting function

- Syntax:

```
F0 = fit(X, Y, FT)
```

Input arguments:

`X, Y` = **<u>COLUMN vectors</u>** containing data to fit to

`FT` = String describing model to fit

Output argument:

`F0` = Fit object

Finding the list of fit types

Easiest way:

```
>> doc fit
```

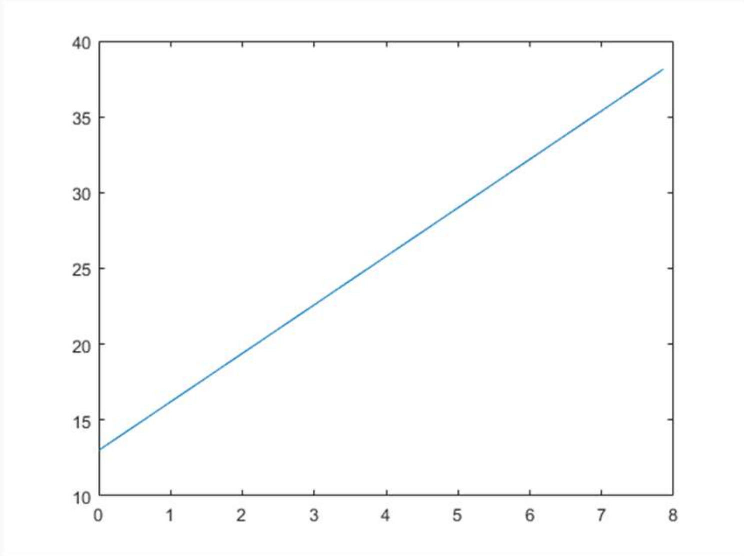
- Scroll to the end, and select

"List of Library Models for Curve and Surface Fitting"

- Link:

https://www.mathworks.com/help/curvefit/list-of-library-models-for-curve-and-surface-fitting.html

What function is most appropriate to fit the following data?



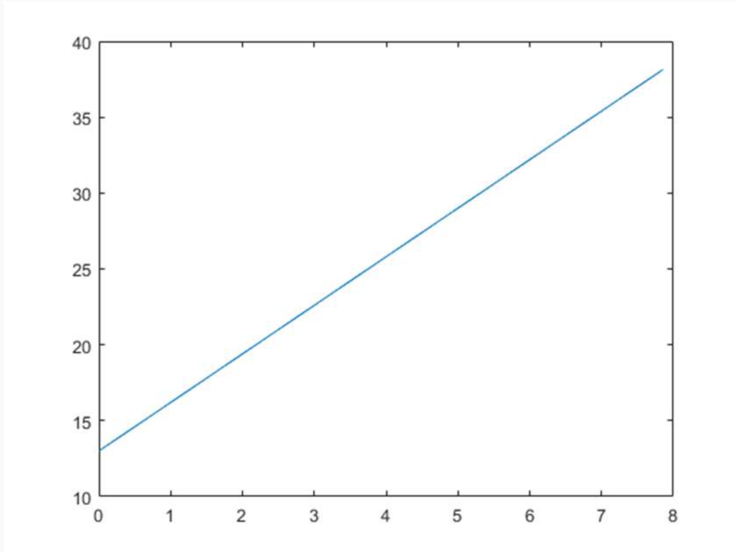
(A) `poly1`

(B) `poly2`

(C) `gauss2`

(D) `sin1`

What function is most appropriate to fit the following data?



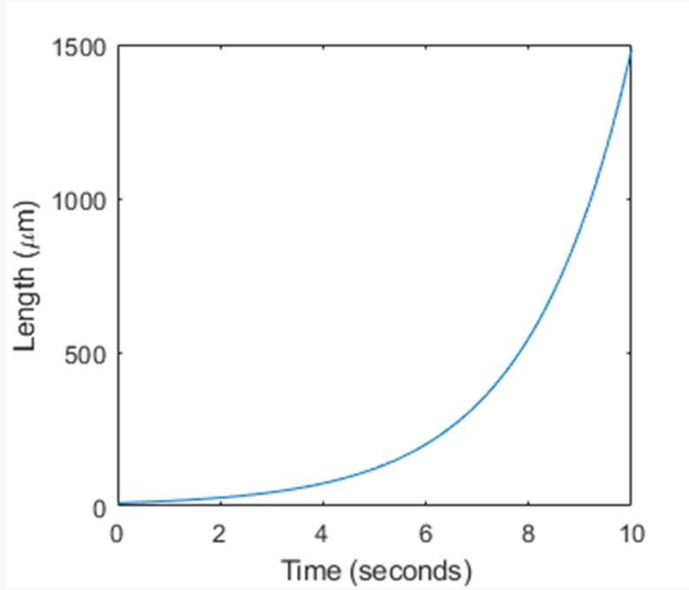
(A) `poly1`

(B) `poly2`

(C) `gauss2`

(D) `sin1`

What model is most appropriate to fit the following data?



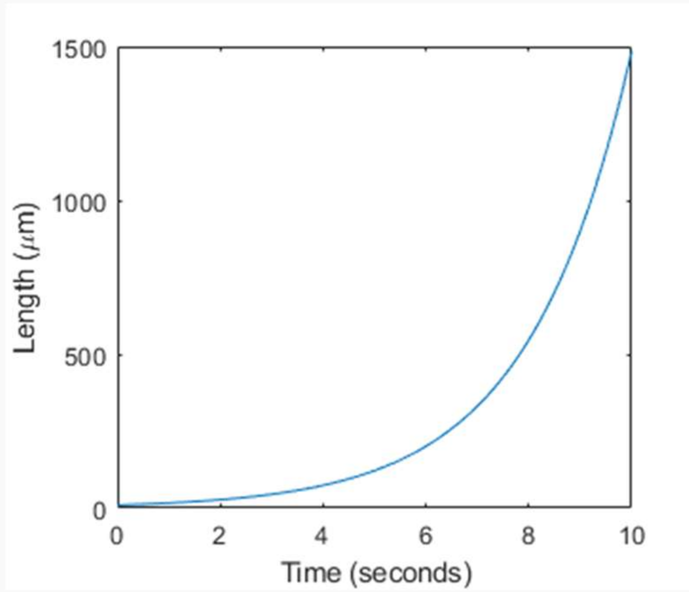
(A) `poly2`

(B) `gauss1`

(C) `exp1`

(D) `exp2`

What model is most appropriate to fit the following data?



(A) `poly2`

(B) `gauss1`

(C) **`exp1`**

(D) `exp2`

Example

- Edit your script to fit the original curve yy

$$F0 = fit(X, Y, FT)$$

Input arguments:

X, Y = COLUMN vectors containing data to fit to

FT = String describing model to fit

Output argument:

F0 = Fit object

Example

- Going back to your script, fit a function to the yy data

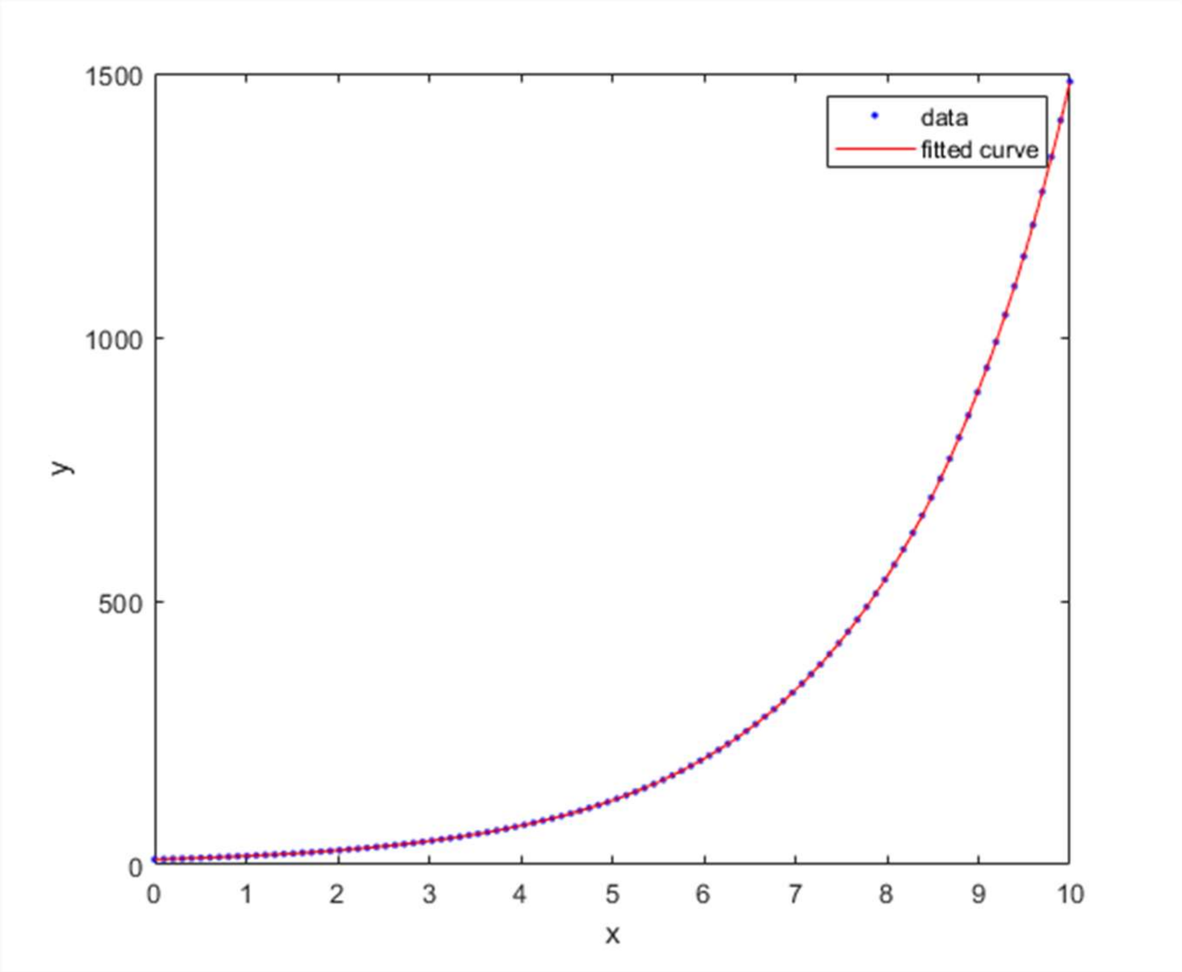
```
fitObj = fit(xx', yy', 'exp1')
```

- Plot the result

```
plot(fitObj, xx, yy)
```

- Pay attention to the order of the arguments
- The first argument is the fit object → this is a special plot

Plotting the fitted data



Be sure to check your fit!

What is the growth rate/growth constant of the curve?

```
>> fitObj

    General model Exp1:
     fitObj(x) = a*exp(b*x)
     Coefficients (with 95% confidence bounds):
       a =          10  (10, 10)
       b =         0.5  (0.5, 0.5)
```

What is the growth rate/growth constant of the curve?

```
>> fitObj

     General model Exp1:
      fitObj(x) = a*exp(b*x)
      Coefficients (with 95% confidence bounds):
        a =           10  (10, 10)
        b =          0.5  (0.5, 0.5)

>> fitObj.b

   ans =
       0.5000
```

Goodness-of-fit

- Values that statistically describe how well the data fits to the model

```
[fitObj, gof] = fit(xx', yy', 'exp1')
```

- `gof` is a struct that has five different fields

- The statistic we will use for this course is

  `rsquare` – coefficient of determination

Details about the others:
https://www.mathworks.com/help/curvefit/evaluating-goodness-of-fit.html

R-squared or the coefficient of determination

- Simple explanation of R^2 is for a measure of the difference of the data from the model

- R^2 typically takes values between 0 and 1
  - $R^2 = 0$ – no data lies on the line described by the model ("bad fit")
  - $R^2 = 1$ – all the data lies on the line described by the model ("perfect fit")

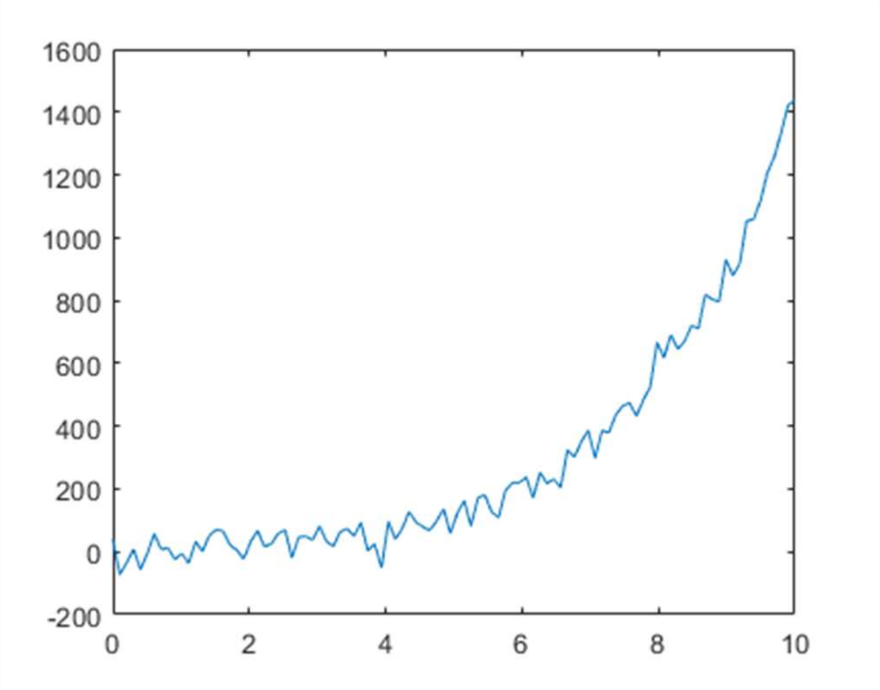
- Typically want values – 0.98 and above

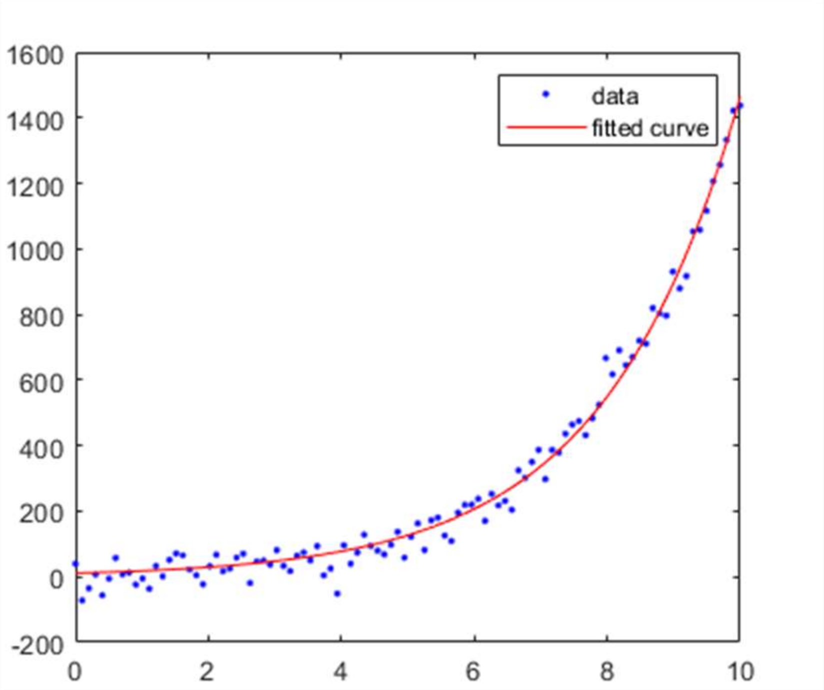
- Beyond the scope of this class, but just be aware that R^2 might not always be the best indicator of goodness-of-fit

Issues that could affect fitting results

- Noisy images/noisy signals – try median/Gaussian filtering

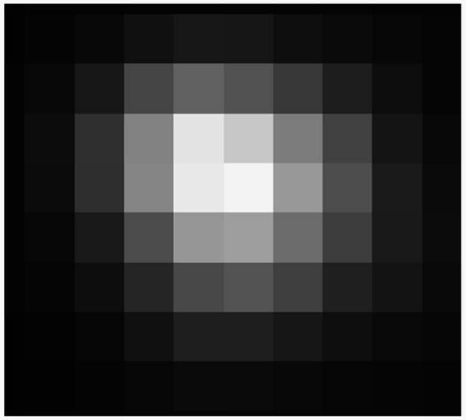
- Small variations during segmentation – try smoothing (function `smooth`)

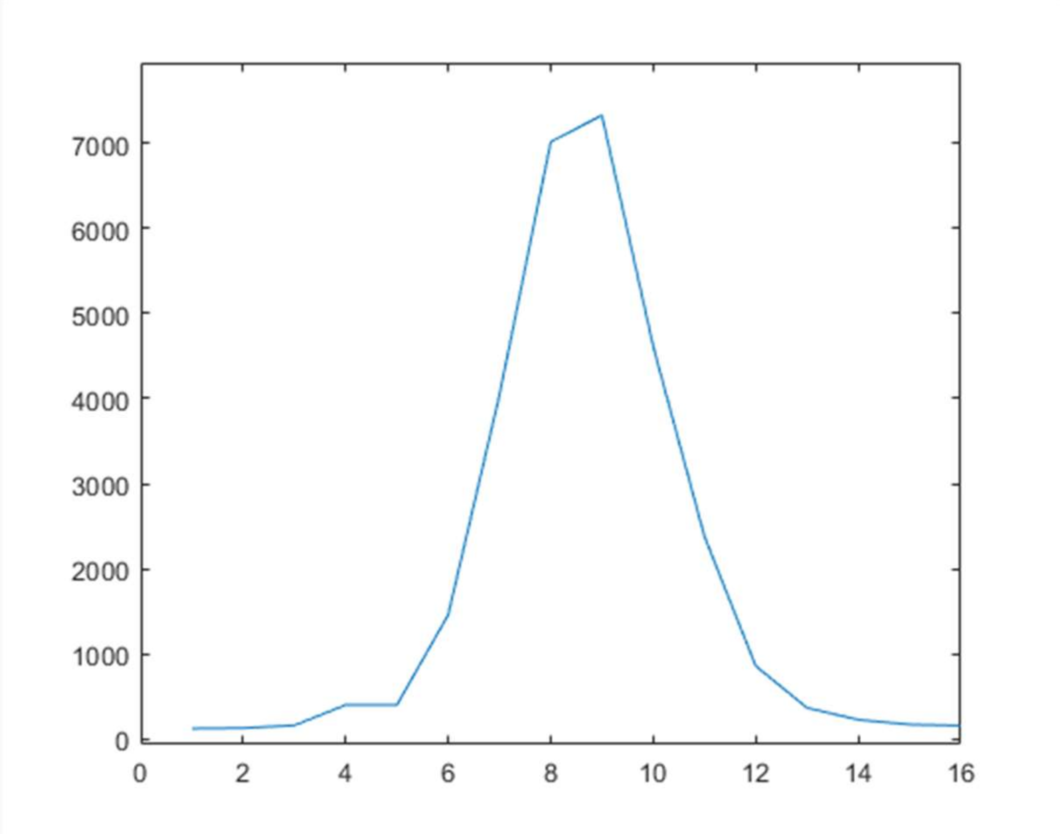




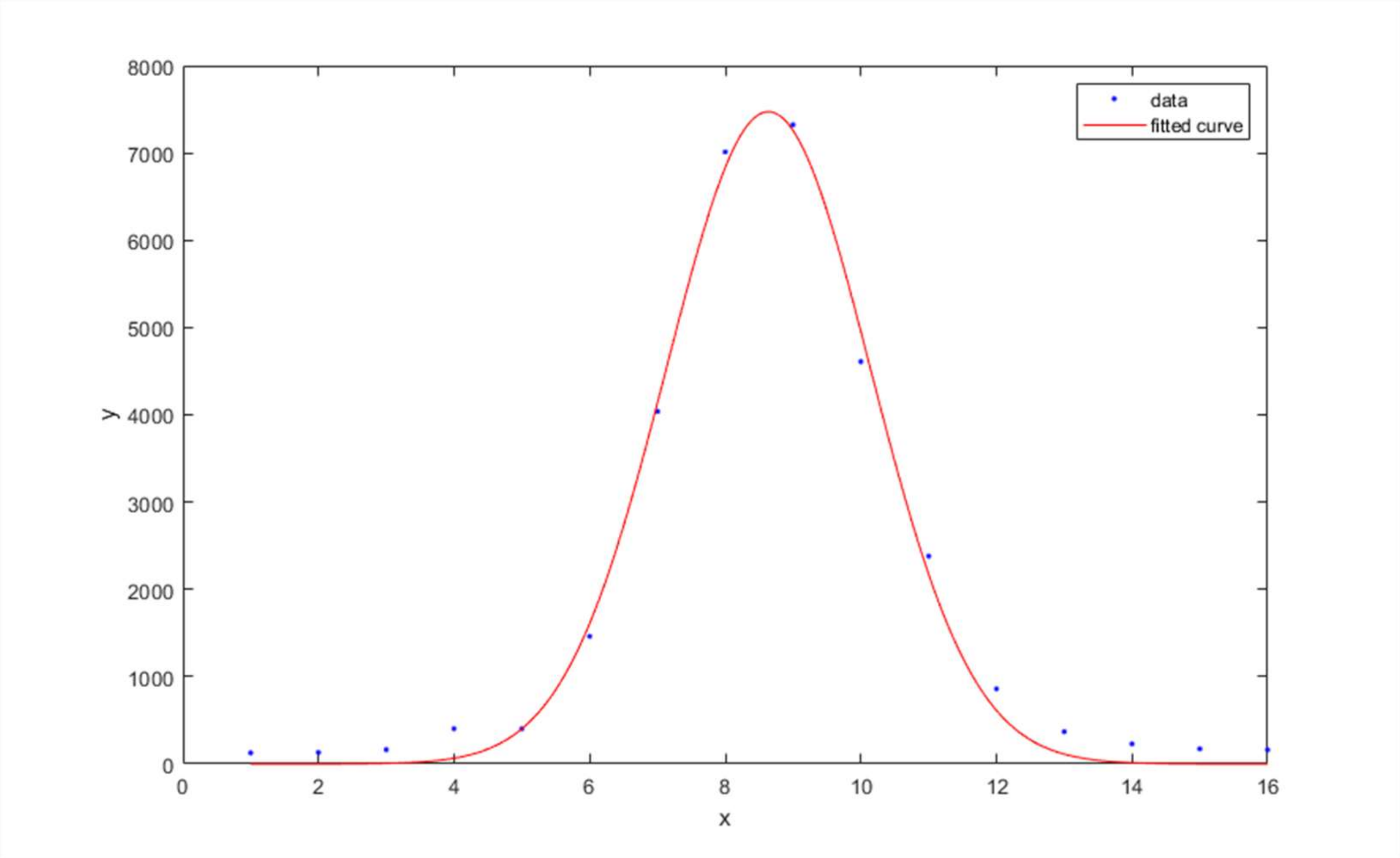
Issues that could affect fitting results

- Constant background illumination/camera dark noise – imaging issue





Inaccurate fitting due to offset



The model 'gauss1' does not have a term for a constant offset

Have to subtract the offset to get a proper fit

Final tips

- Read the documentation/Check the model equations very carefully
- Some equations are a little bit different – there are small variations to general equations depending on its use
    - There is an example of this in your homework
- If unsure what the function looks like, plot it out