Introduction to Matlab & Numerical Methods for solving Ordinary Differential Equation (ODE)

Tianyun (Jason) Lin

Lab 1, C&SB 150, Spring 2019

April 5th, 2019

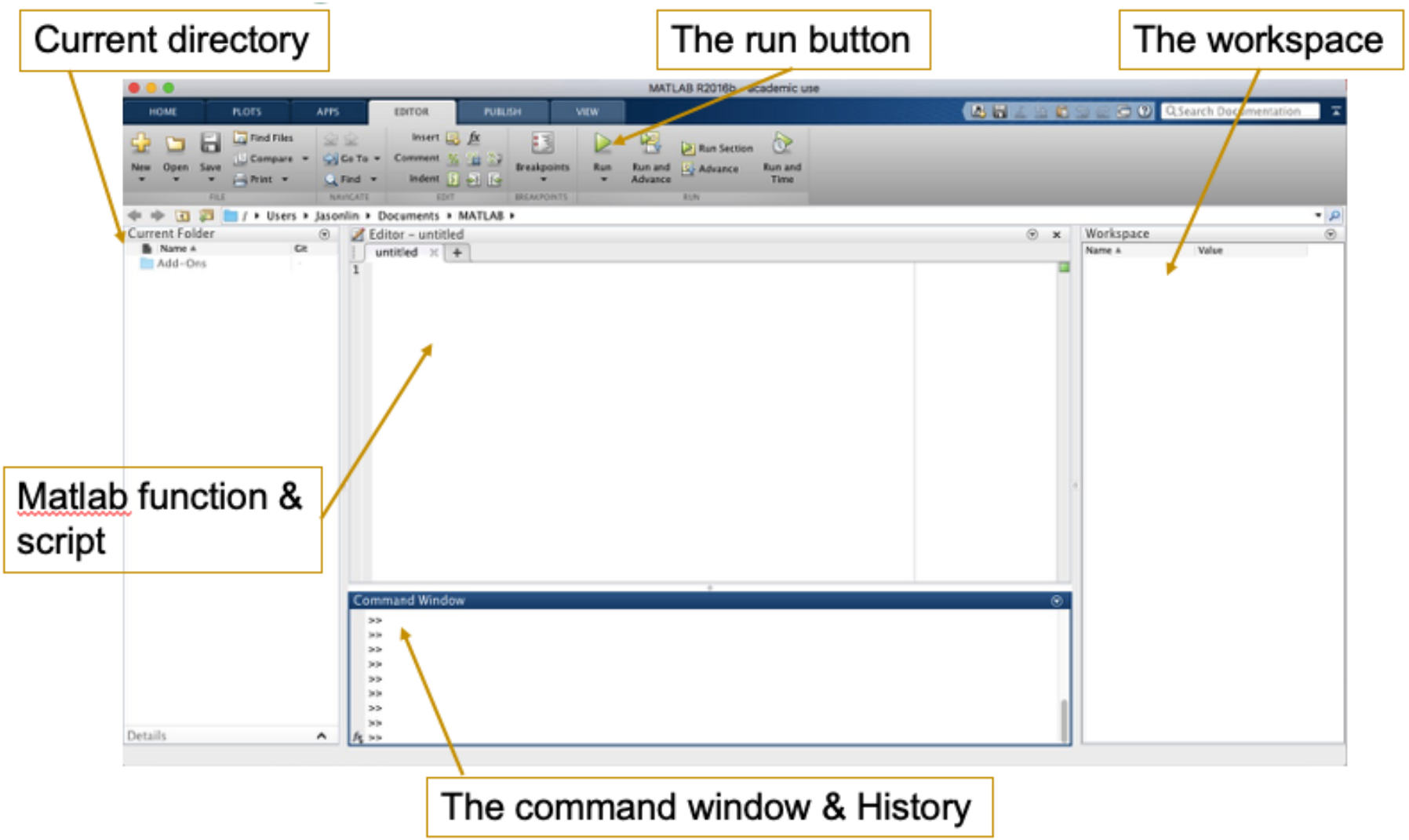
Introduction to Matlab

What is Matlab?

- "**Mat**rix **Lab**oratory"
- A numerical computing environment.
- Also a programming language.
- We'll use Matlab to model and simulate various biological systems.
- If you already know or prefer another programming language, use it!
  - I know most programming language except C++ and Mathematica. So if you prefer those two languages, please try to use Matlab for this course. It is going to be hard for me to grade. Plus if you know C++ or Mathematica, Matlab should be straightforward.

Introduction to Matlab

Getting started



Introduction to Matlab

Getting started

- You can now open up Matlab and follow along the examples...

Introduction to Matlab

Using Matlab as a calculator

```
>> 5 + 6
ans =
11
```

Or

```
>> log10(10)
ans =
1
```

- If the output variable is not specified, Matlab will store the answer for your expression in 'ans', which stands for 'answer'
- Can store the answer for your output in a variable of your choice.
- Use commands 'who' or 'whos' in the command window to see the variables.

Introduction to Matlab

Mathematical functions

- Lots of predefined functions (use tab to see more similar functions):

Table 2.1: Elementary functions

`cos(x)`	Cosine	`abs(x)`	Absolute value
`sin(x)`	Sine	`sign(x)`	Signum function
`tan(x)`	Tangent	`max(x)`	Maximum value
`acos(x)`	Arc cosine	`min(x)`	Minimum value
`asin(x)`	Arc sine	`ceil(x)`	Round towards $+\infty$
`atan(x)`	Arc tangent	`floor(x)`	Round towards $-\infty$
`exp(x)`	Exponential	`round(x)`	Round to nearest integer
`sqrt(x)`	Square root	`rem(x)`	Remainder after division
`log(x)`	Natural logarithm	`angle(x)`	Phase angle
`log10(x)`	Common logarithm	`conj(x)`	Complex conjugate

- Also predefined constants (Avoid using i or j for looping):

Table 2.2: Predefined constant values

`pi`	The π number, $\pi = 3.14159\ldots$
`i,j`	The imaginary unit i, $\sqrt{-1}$
`Inf`	The infinity, ∞
`NaN`	Not a number

- Use `help` to learn more about the function.

Introduction to Matlab

Using Matlab as a programming language

- Store values in memory.

```
>> x = 5
x =
5
```

- "=" vs. "=="
  - "=": Assign the value on the right side of the equal sign to the left.
  - "==": Check if two sides are the same (output 1 if the same and 0 otherwise).

Introduction to Matlab

Arrays

- Can be vectors or matrices.
- One can define vectors with colon operator.
- The colon operator allows one to set up a vector of equally spaced entries.

For example:

```
>> x = 1:4 % generate a vector 1 2 3 4
>> x = 1:2:6 % generate a vector 1 3 5
```

- Good when the matrix or vector is too large to enter one element at a time.
- Also used in for loop.
- One can also use `linspace(a,b,n)`

Introduction to Matlab

Arrays

- Basic math functions also work for arrays.

For example:

```
>> x = [1 2 3 4 5]; % generate a vector 1 2 3 4 5
>> y = exp(x)
y =
2.7183    7.3891   20.0855   54.5982  148.4132
```

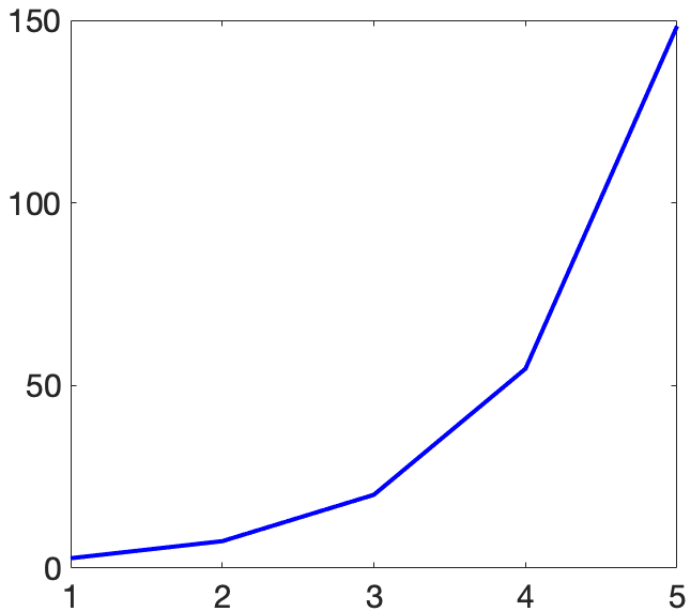
Introduction to Matlab

Arrays

We can then easily plot y against x in Matlab:

```
>> figure % create a blank figure to store your plot
>> plot(x,y)
```

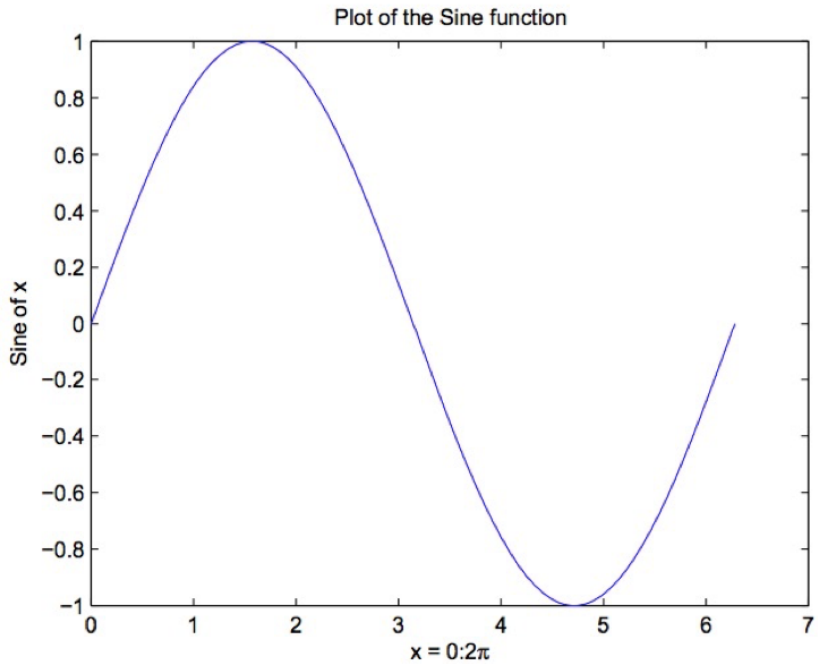
which gives:



Introduction to Matlab

Basic plotting

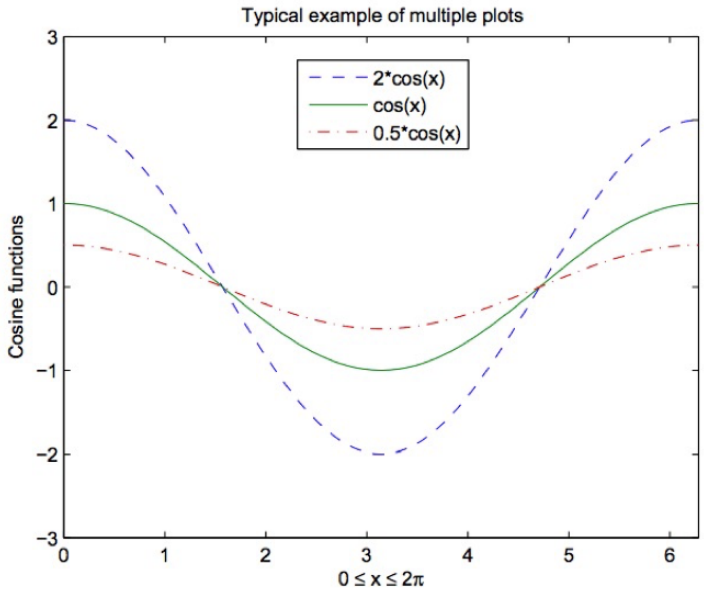
```
>> x = 0:0.1:2*pi;
>> y = sin(x);
>> plot(x,y)
>> xlabel('x = 0:2\pi')
>> ylabel('Sine of x')
>> title('Plot of the sine function')
```



Introduction to Matlab

Multiple data sets in one plot

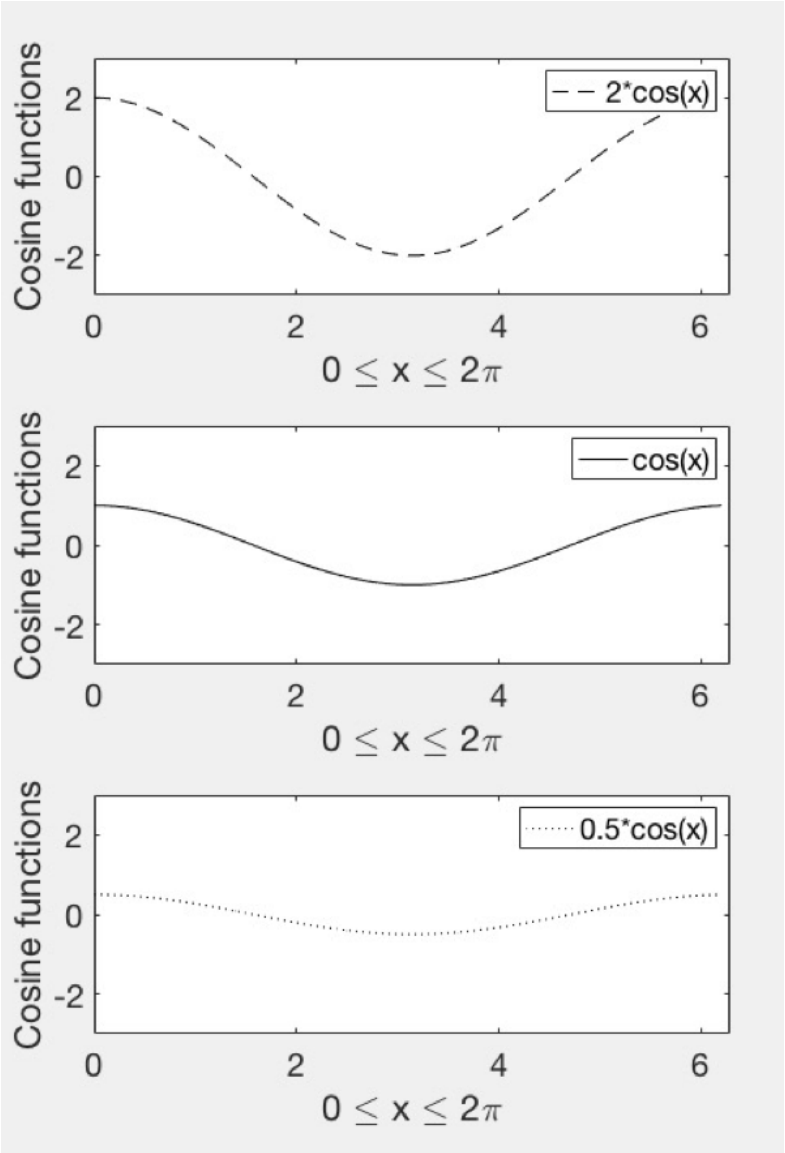
```
>> x = 0:0.1:2*pi;
>> y1 = 2*cos(x);
>> y2 = cos(x);
>> y3 = 0.5*cos(x);
>> plot(x,y1,'--',x,y2,'-',x,y3,':')
>> xlabel('0 \leq x \leq 2\pi')
>> ylabel('Cosine functions')
>> legend('2*cos(x)','cos(x)','0.5*cos(x)')
>> title('Typical example of multiple plots')
>> axis([0 2*pi -3 3])
```



Introduction to Matlab

Or split your plot into a bunch of smaller plots

```
>> x = 0:0.1:2*pi;
>> y1 = 2*cos(x);
>> y2 = cos(x);
>> y3 = 0.5*cos(x);
>> subplot(3,1,1)
>> plot(x,y1,'k--')
>> xlabel('0 \leq x \leq 2\pi')
>> ylabel('Cosine functions')
>> legend('2*cos(x)')
>> axis([0 2*pi -3 3])
>> subplot(3,1,2)
>> plot(x,y2,'k-')
>> xlabel('0 \leq x \leq 2\pi')
>> ylabel('Cosine functions')
>> legend('cos(x)')
>> axis([0 2*pi -3 3])
>> subplot(3,1,3)
>> plot(x,y3,'k:')
>> xlabel('0 \leq x \leq 2\pi')
>> ylabel('Cosine functions')
>> legend('0.5*cos(x)')
>> axis([0 2*pi -3 3])
```



Introduction to Matlab

Specify line styles and colors

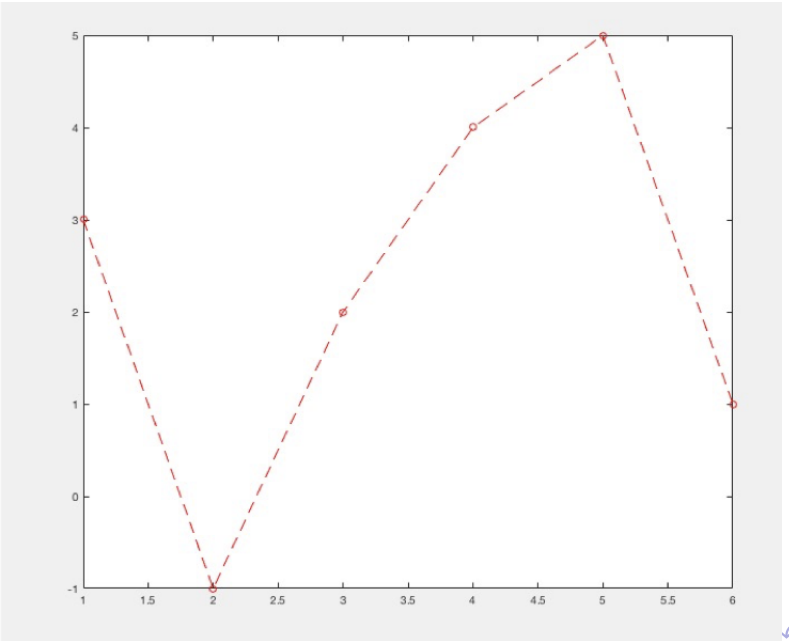
```
>> plot(x,y,'style_color_marker')
```

Table 2.3: Attributes for `plot`

Symbol	Color	Symbol	Line Style	Symbol	Marker
k	Black	–	Solid	+	Plus sign
r	Red	−−	Dashed	o	Circle
b	Blue	:	Dotted	*	Asterisk
g	Green	−.	Dash-dot	.	Point
c	Cyan	none	No line	×	Cross
m	Magenta			*s*	Square
y	Yellow			*d*	Diamond

For example:

```
>> x = [1 2 3 4 5 6];
>> y = [3 -1 2 4 5 1];
>> plot(x,y,'r--o')
```



Introduction to Matlab

Save plots you generated

Right after your plot, type:

```
>> print('filename','-dpng')
```

Or save as .fig file:

```
>> savefig('filename.fig')
```

Introduction to Matlab

Writing Matlab scripts & functions

- A script is just a list of commands to be run in some order.
- Placing these commands in a file that ends in .m alow you to run the scripts by typing the name of the file in the Matlab command window.
- Undesirable effect:
  - variables already exist in workspace may be overwritten.
  - the output of the script can be effect by the existing variables.

Introduction to Matlab

Writing Matlab scripts & functions

- A function can take in particular variables (called arguments) and doing something specific and return some particular result.
- function return-values = functioname(arguments)

Introduction to Matlab

Loops: for loop

```
>> for x = 1:100
       a(x) = x^2;
   end
```

Use for iterative methods or looping through a matrix or vector.

Introduction to Matlab

Loops: while loop

```
a = 10
n = 1
>> while 2^n < a
        n = n + 1;
   end
```

Output the smallest integer n such that $2^n \geq n$.

Introduction to Matlab

Logical statements

- If statement:

```
>> If some-logical-relation
       statements;
   end
```

- The statements will be executed only if the relation is true.

Introduction to Matlab

Other logical statements

- Relations operators in Matlab:

$<$	less than
$>$	greater than
$<=$	less than or equal
$>=$	greater than or equal
$==$	equal
$\sim=$	not equal

- Multiple relations can be connected through:

&	and
$\vert$	or
$\sim$	not

Introduction to Matlab

"Vectorize"

- for loops are SLOW in Matlab.
- Matlab "thinks" in vectors and matrices, and it is most efficient if user treat all the variables as a vector or a matrix.
- Try examples in section 3.3 in the assignment.

Introduction to Matlab

Find values in a array

```
>> find(X)
```

- The function above outputs the indices of all nonzero elements of X. And

```
>> find(X>2)
```

- returns the indices of entries that are greater than 2.
- Note: X could also be a matrix. In this case, the function returns the linear indices as if the matrix was stored as a single column of elements.
- find out more using `help` or Matlab documentation online.

Introduction to Matlab

Save and load data

```
>> save filename
```

Or

```
>> load filename
```

- find out more using `help` or Matlab documentation online.

Introduction to Matlab

Solve differential equation numerically in Matlab

- Matlab has 2 functions ode23, ode45, which are capable of numerically solving differential equations, both of them use Runge-Kutta, but different order of approximation.

```
>> [T, Y] = ode45('yprime', t0, te, y0);
```

- 'yprime' is the name of the function that describes your system of differential equations.
- Ouputs a vector of time, T, and a matrix Y that has the values of each variable in your system.
- The hard part is defining 'yprime'

Introduction to Matlab

Solve differential equation numerically in Matlab

- Take logistic growth model for example:

$$\frac{dN}{dt} = rN(1 - \frac{N}{K})$$

Where r is growth rate and K is carrying capacity. And we can define this ODE (given that we know r and K) in matlab as:

```
function dNdt = logGrowth(t,N)
r = 0.5;
K = 100;
dNdt = r * N .* (1 - N/K);
```

- We can then solve it using ode45 in Matlab (`intLog.m`).

Introduction to Matlab

Solve differential equation numerically in Matlab

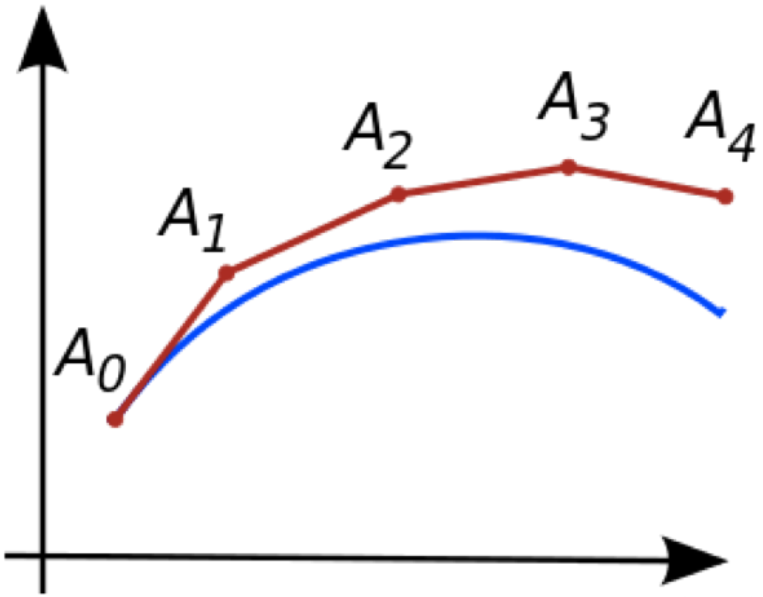
```
% Numerical solution of the logistic equation
t0 =0; % Initial time
tf =50; %Final time
N0 =1; %Initial population size
[T, vNint] = ode45(@logGrowth, [t0 tf], N0);
```

- Hint for challenge problem: You CAN pass extra parameters to ODE functions. Check ode45 documentations.

Numerical Methods for solving ODE

Euler's method

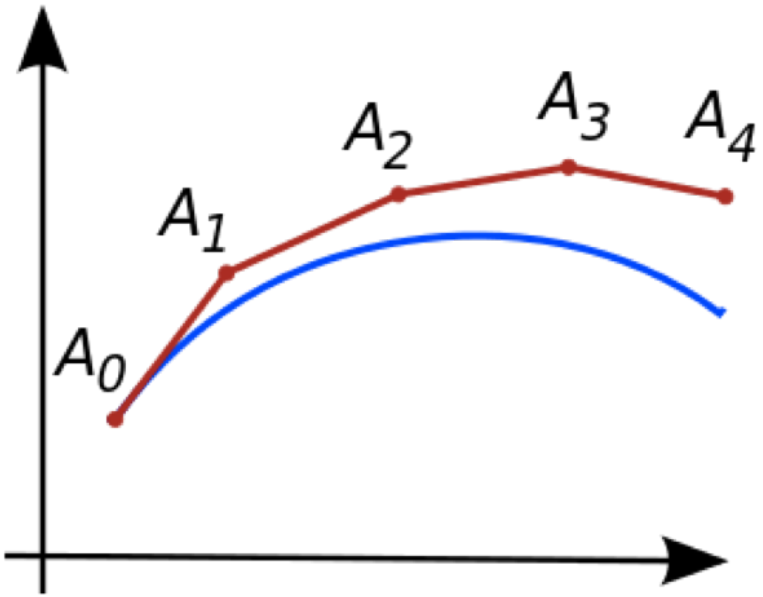
- A.k.a forward Euler's method.
- First order numerical method for solving ordinary differential equations.
- Given $\frac{dy}{dt} = f(t, y)$; $y(0) = c$
- Starting from initial condition
- $y_{n+1} = y_n + \triangle t f(t_n, y_n)$



Numerical Methods for solving ODE

Euler's method

- Another intuition:
- Think about Taylor expansion:
- $y(t_{n+1}) = y(t_n + \triangle t) = y(t_n) + \triangle t y'(t_n) + \frac{1}{2}\triangle t^2 y''(t_n) + ...$



Numerical Methods for solving ODE

Runge-Kutta methods (RK4)

$$y_{n+1} = y_n + \Delta t\, T_4(t_n, y_n, \Delta t)$$

$$T_4(t_n, y_n, \Delta t) = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$k_1 = f(t_n, y_n)$$

$$k_2 = f\left(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta t}{2}\, k_1\right)$$

$$k_3 = f\left(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta t}{2}\, k_2\right)$$

$$k_4 = f(t_n + \Delta t, y_n + \Delta t k_3)$$

Other helpful resources for mastering Matlab:

- Introduction to Matlab for Engineering Students, David Houcque
- Matlab basics and a little beyond: a short introduction to Matlab, David Eyre
- Matlab Primer, Kermit Sigmon
- Day 1, Introduction to using Matlab, MIT
- Day 2, Introduction to using Matlab, MIT