CircuitLab and Matlab

EE 206 Circuits I

Jake Glower - Lecture #0

Please visit Bison Academy for corresponding
lecture notes, homework sets, and solutions

Circuit design and analysis is usually a three step process:

- First, you design your circuit on paper.
- Next, you simulate your design to make sure it works.
- Then you build your circuit in lab

The tools we'll be using in this class to do this are:

- Matlab: a tool that makes circuit design and analysis much easier. Think of Matlab as a calculator which can solve 50 equations for 50 unknowns and graph the results.
- CircuitLab: a very friendly circuit simulator which allows you to build a circuit using drag and drop. Once built, you can find the voltages or run a time simulation to see how the voltages change over time (useful for AC analysis).
- Breadboards, oscilloscopes, and multimeters: used in lab to allow you to connect components together to build a circuit.

CircuitLab (www.CircuitLab.com)

- CircuitLab is a circuit simulator
- Very friendly to use
- Fairly intuitive (can jump right in)
- Multiple tutorials on YouTube are available

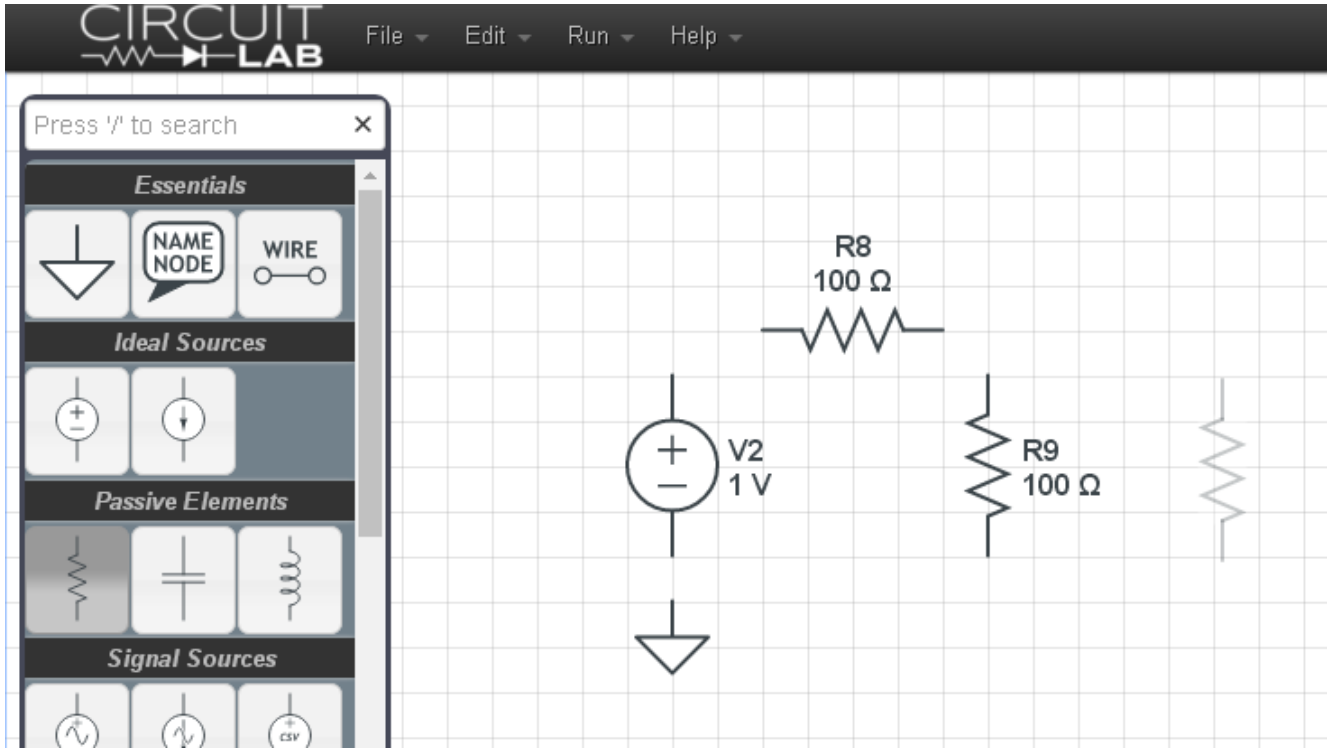
Starting Out:

- Create an account
- Use your NDSU email address and it's free
- NDSU ECE pays of university license
- $34/year for personal license (2020 prices)

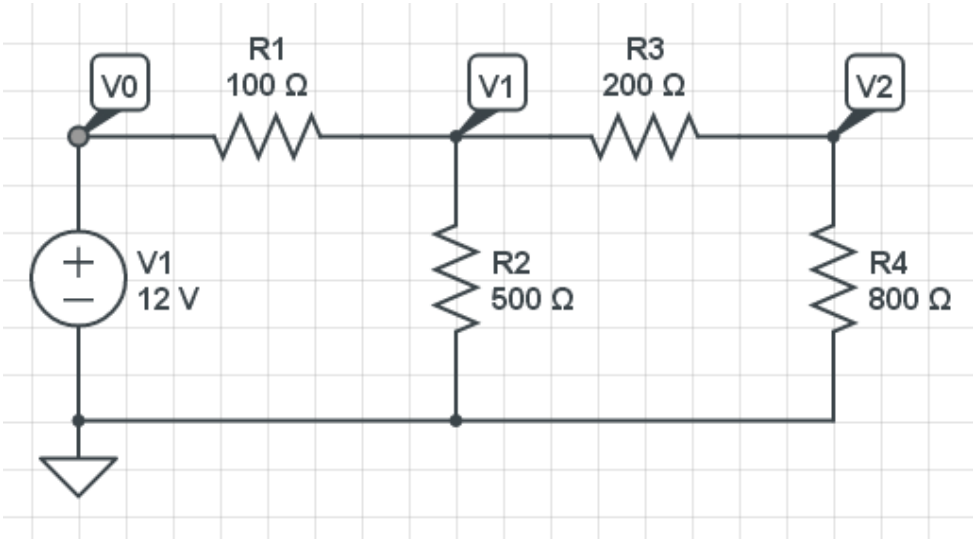
Starting CircuitLab

To build a circuit, drag and drop components from the left (we'll talk about what each one is later in this course)

- R rotates the component
- Double click to change the value

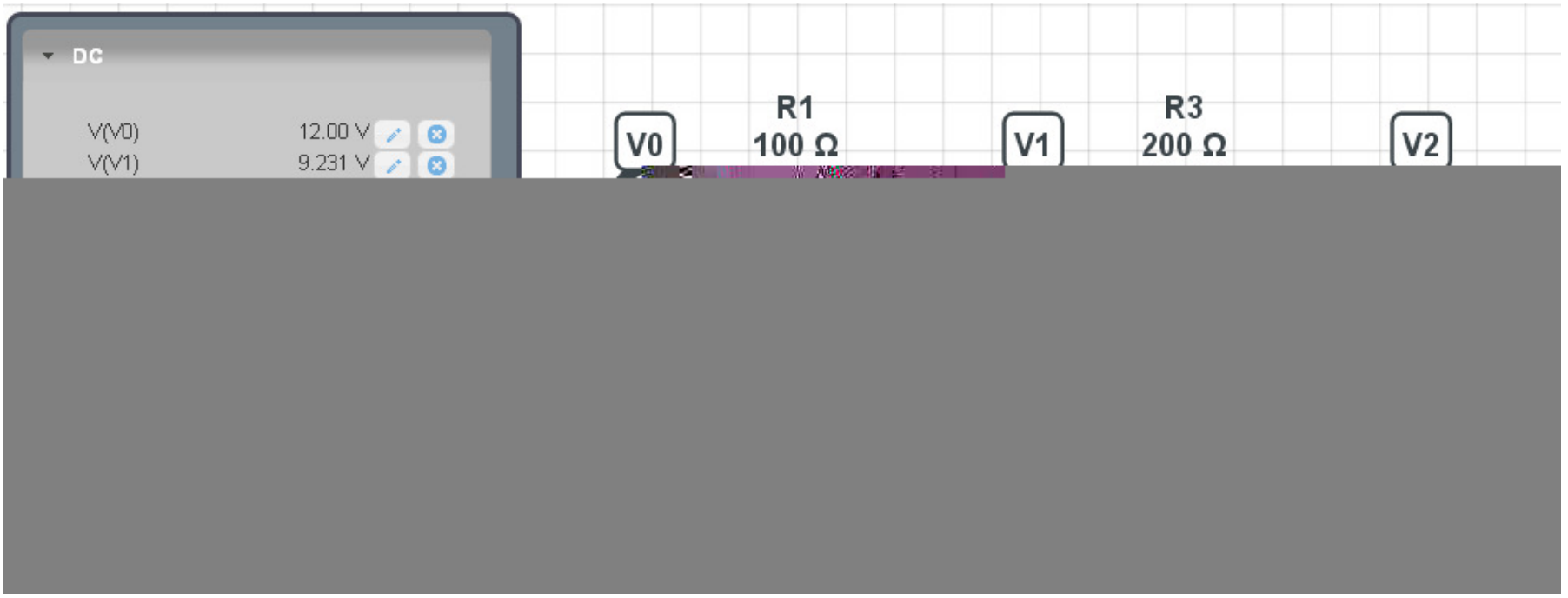


Click and drag to connect the wires together. Adding labels allow you to look at different voltages

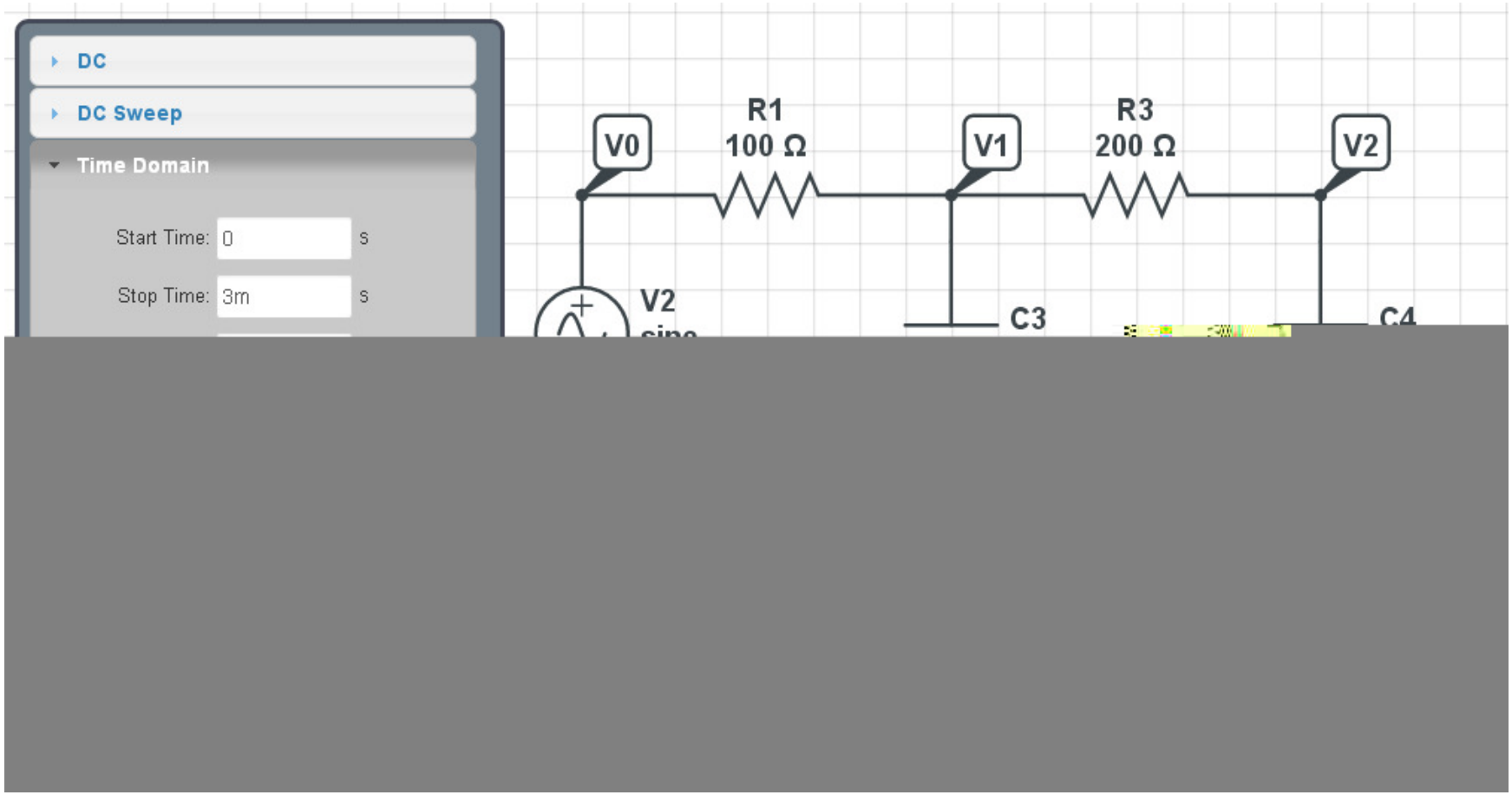


To find the voltages, click on

- Simulate
- + Add Expression (Select V0, V1, and V2)
- Run DC Solver
- Voltages should match what you calculated with Matlab and measured in lab

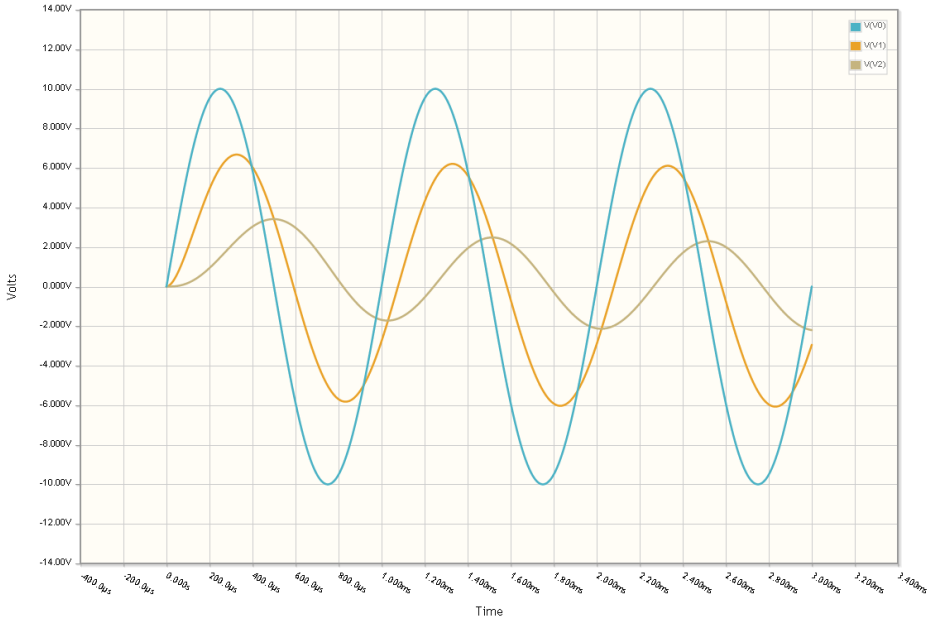


You can also look at transient simulation. Change the source to an AC signal and change R2 and R4 to capacitors



Now run a Timer Domain simulation

- The input is a 1kHz sine wave. Run the simulation for a few cycles (3 cycles or 3ms)
- Make the time step 100 to 1000x smaller (3us gives 1000 points on the following graph). If you make the time step too small, it will take some time to finish the simulation.
- + Add expression to see V0, V1, and V2
- Click on Run Time Domain Simulation
- Should be the same as you measure in lab with an oscilloscope



Introduction to Matlab

Becoming familiar with MATLAB

- The console
- The editor
- Scripts
- Functions
- Graphics Window

General environment and the console

When you start up Matlab, the screen should look something like this:

I usually close everything doen except the command window.

The command window is like a calculator:

```
>> (2 + 3)*5

ans =   25

>> a = (2+3)*5

a =  25

>> b = 1.3 * a^3

b =  2.0313e+004
```

Matrices in Matlab

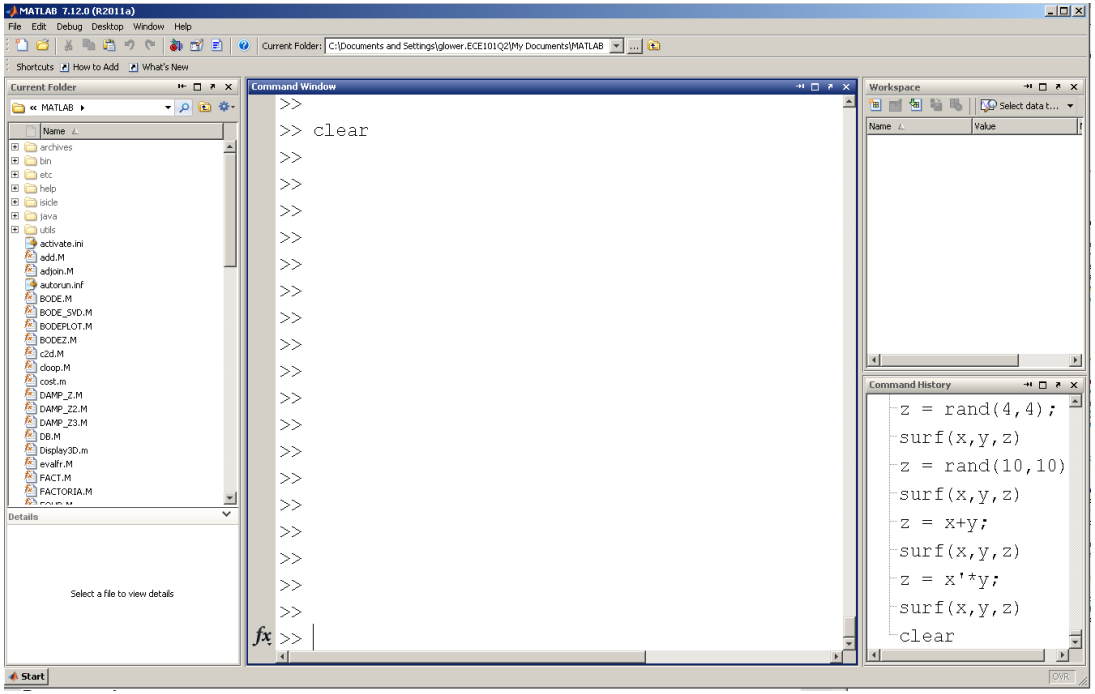
The syntax to input a matrix are:

```
[       start of a matrix
,       next column (a space
also works)
;       next row
]       end of matrix.
```

Example:

```
A = [1,2,3 ; 4,5,6]

     1     2     3
     4     5     6
```



Matrix Addition: Dimensions must match exactly

- 2x3 + 2x3 = 2x3
- 2x3 + 2x4 = error

```
A = [1,2,3 ; 4,5,6]

     1     2     3
     4     5     6

B = [2,2,2 ; 3,3,3]

     2     2     2
     3     3     3

C = A + B

     3     4     5
     7     8     9
```

Matrix Multiplication:

- The inner dimenstion must match

$$C_{mn} = A_{mx} B_{xn}$$

Semi-Colon

- ; Matlab does the opeation and does not print the result

```
 x = 2*pi;
% result is computed and stored in x but isn't displayed

 x = 2*pi
% ditto but the result is displayed.

    6.2832
```

Display Settings

```
format short
pi

   3.1416

format long
pi

  3.141592653589793


format shorteng
pi^30

  821.2893e+012
```

for loops:

```
x = zeros(1,5);
for i=1:5
  x(i) = i*i;
  end

x =     1     4     9    16    25
```

while loop

```
N = 1;
while(rand > 0.1)
   N = N + 1;
   end

N = 6
```

if - if else end

```
if(rand < 0.1)
   die = 6;
else
   die = ceil( 6*rand );
end
```

Scripts and Functions in Matlab

Matlab is also a programming language. This allows you to run the same code over and over without having to constantly retyping in the code over and over again.

- Scripts are like code you type in the command window. When you run a script, it is like you just typed in the code in the script in the command window
- Functions are subroutines. They allow you to create new commands in Matlab which can be called by other new functions and so on.

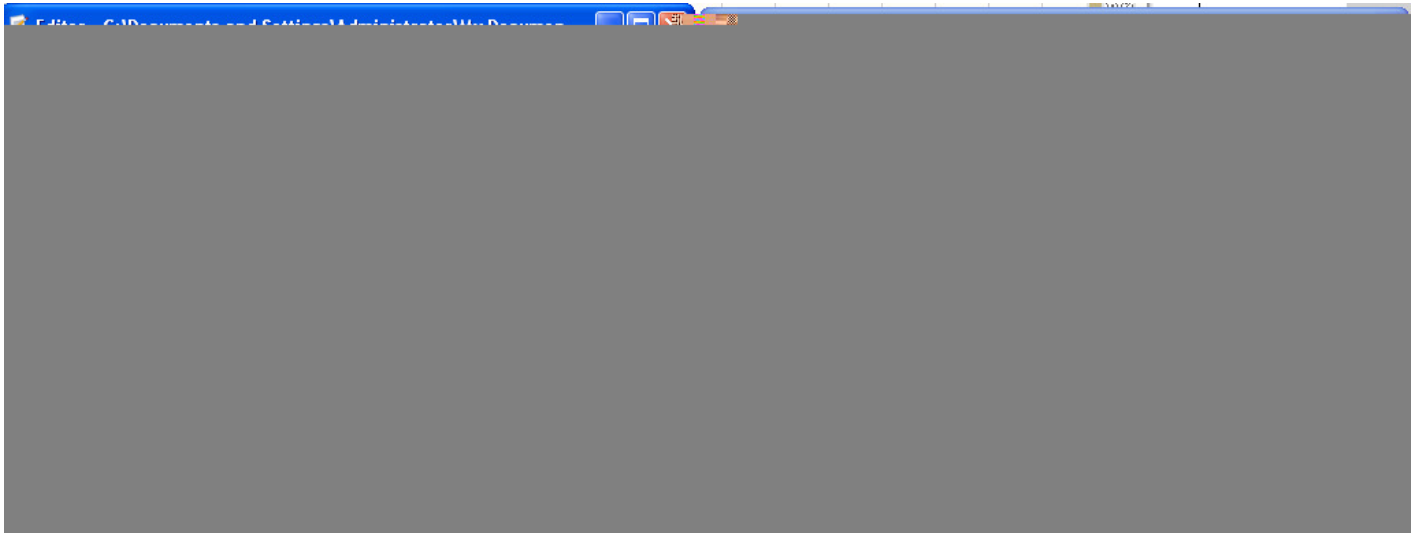
This is why Matlab is used by industry so much: you can create a library of functions and scripts which allow you to design your product.

Scripts:

Suppose you want to see the distribution of rolling two six-sided dice (2d6) 100 times. This is where scripts shine.

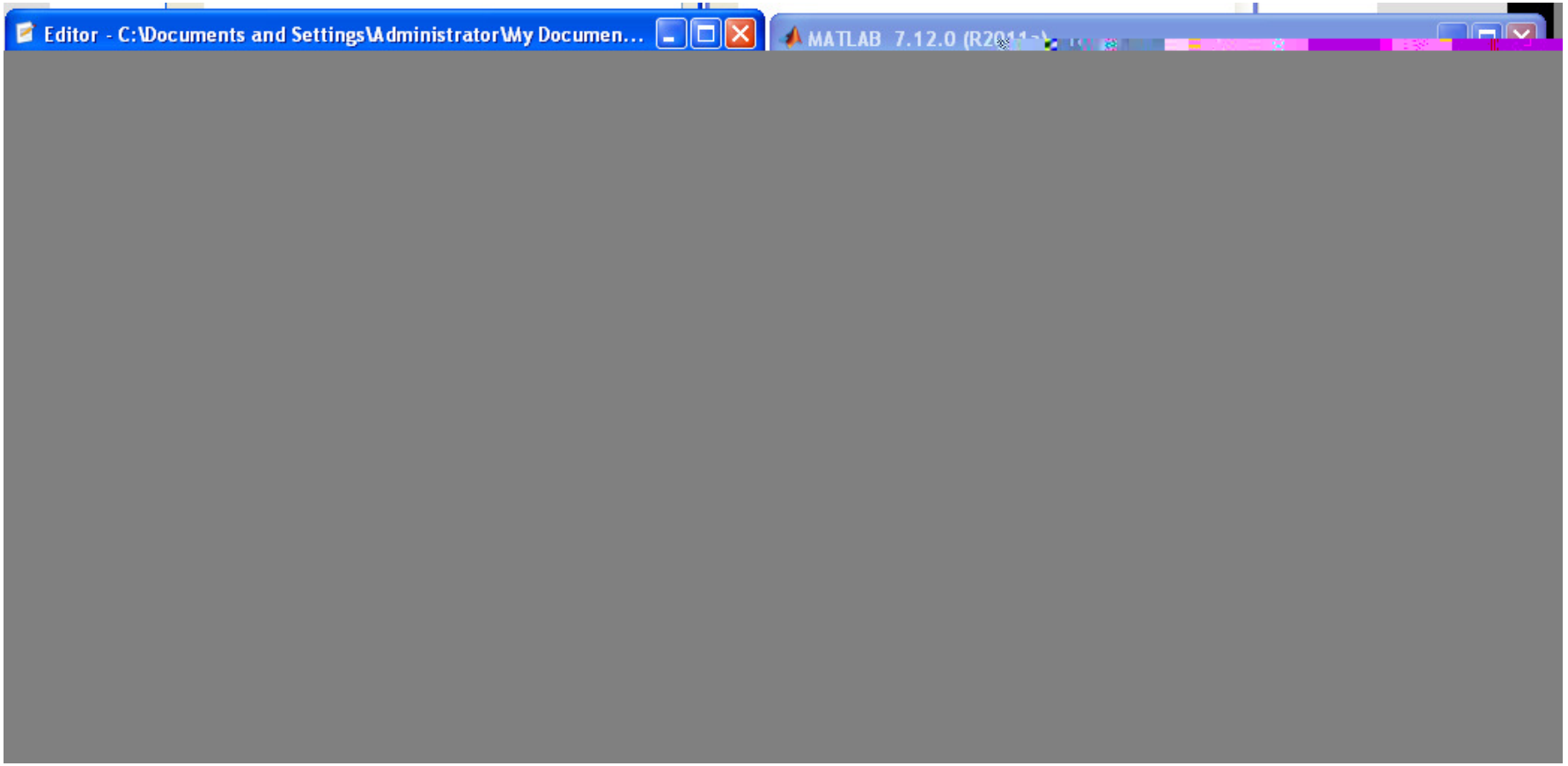
- Start with a New Script.
- Get the code to produce 2 random numbers
- Click on the green arrow to run the script (F5)

note: Save the script in the Matlab directory. If you save it somewhere else on your computer, Matlab won't be able to find it.



Once you are convinced the script works,

- Get it to sum the two dice
- Then repeat 100 times



The beauty of scripts is that

- If you want to see what happens if you run it a second time, you can just by clicking the green arrow again.
- If you want to roll the dice 1000 times instead, change one line of code (i = 1:100 becomes 1:1000)

Functions

A function is a subroutine. It allows you to create new functions in Matlab that can be used over and over again.

For example, let's create a function called NDice. It will be passed two parameters

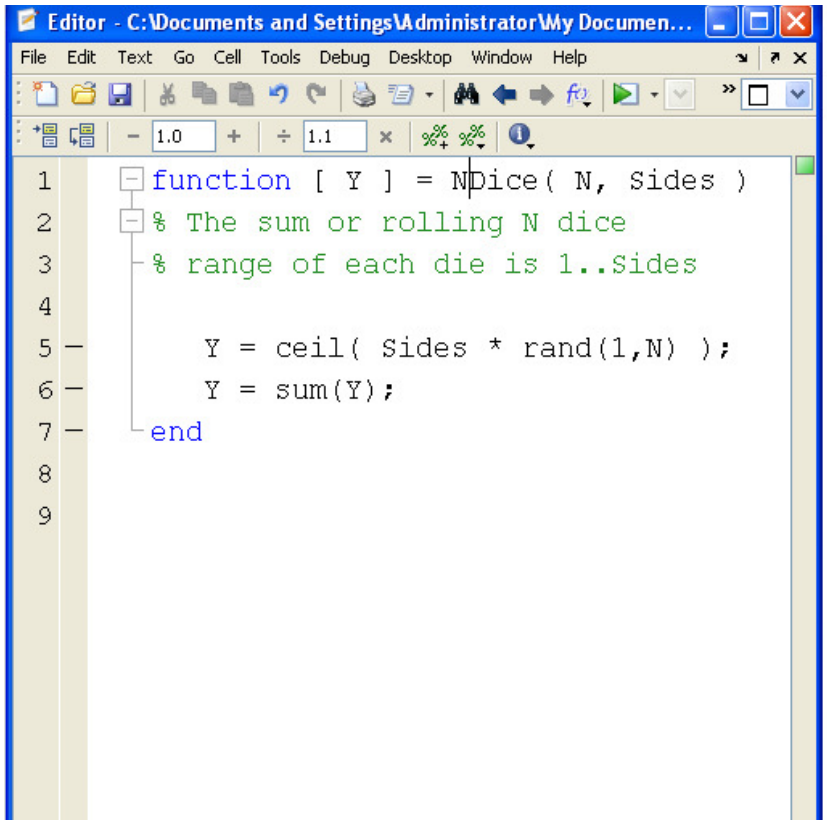
- N: The number of dice
- Sides: The range of numbers (1..6, 1..12, etc.)

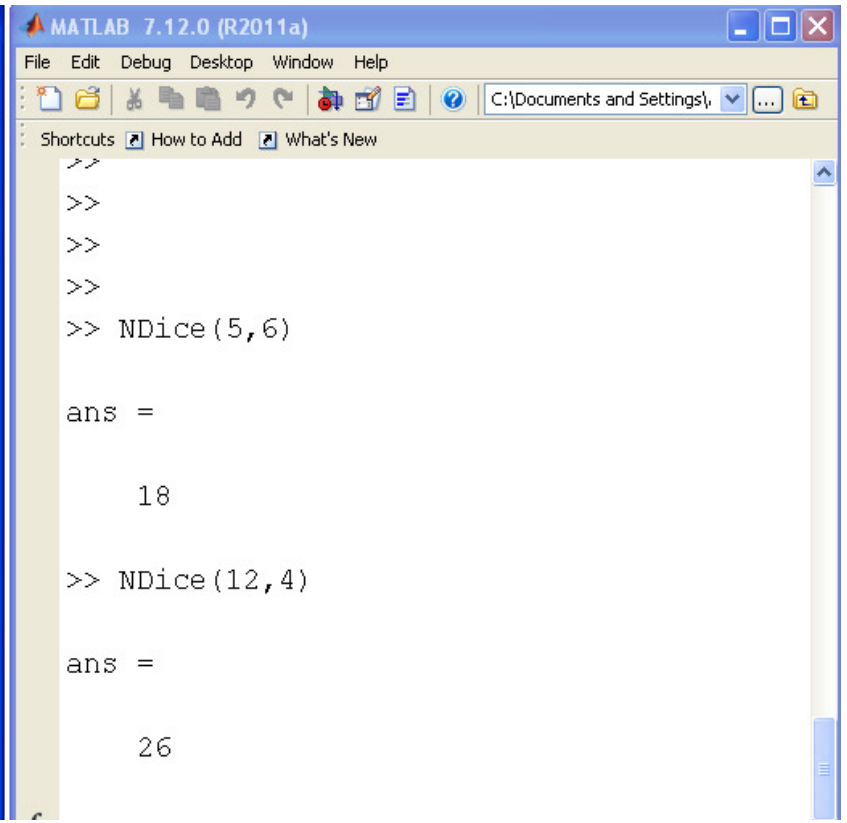
In Matlab,

- click on New Function
- Create the subroutine
- Save with the same name as the function

With that you just created a new function in Matlab called NDice. From the command window, you can

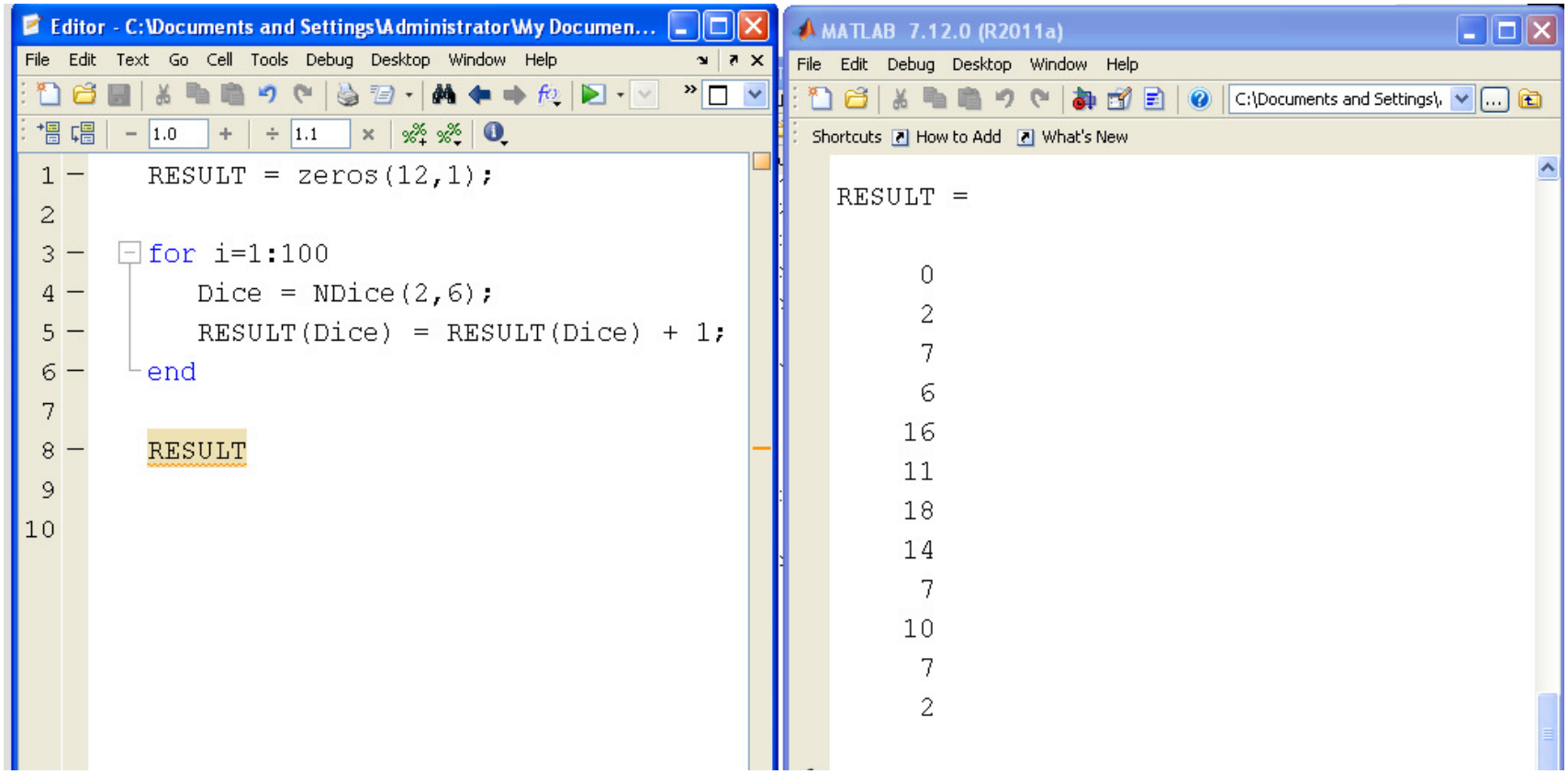
- Find the sum of rolling 5d6 (total is 18)
- Find the sum of rolling 12d4 (total is 26)





Now that you have this function, you can clean up the script

- Note that since this script uses a function we just create, other people won't be able to run this script unless they too create the function NDice.



Plotting Functions in Matlab:

Matlab has some pretty good graphics capabilities.

Matlab Plot Command	x axis	y axis	type of function
plot(x,y)	linear	linear	$y = ax + b$
semilogx(x,y)	log()	linear	$y = a \cdot e^{bx}$
semilogy(x,y)	linear	log()	$y = a + b\ln(x)$
loglog(x,y)	log()	log()	$y = a \cdot b^{x}$
subplot(abc)	Create 'a' rows, 'b' columns of graphs. Starting at #c		

For example, plot the function

$$y = 3x + 4$$

```
x = [0:1:10]';
y = 3*x + 4;

subplot(131)
plot(x,y);  % connect the points with a line

subplot(132)
```

```
 plot(x,y,'.');
% plot a dot at each point

 subplot(133);
 plot(x,y,'.-');
% connect the points and add a dot
```