Matrix operations
Scripts

Matrix transpose

- if A is an m x n matrix then the transpose of A is an n x m matrix where the row vectors of A are written as column vectors matrix transpose

```
>> u = [1 2 3];
>> v = u'
v =
     1
     2
     3

>> A = [1 2 3;
        4 5 6];
>> B = A'
B =
     1     4
     2     5
     3     6
```

Arithmetic operations with arrays

- you can perform element-by-element arithmetic with two arrays of the same size

operator	name
`+`	addition
`-`	subtraction
`.*`	multiplication
`./`	right array division
`.\`	left array division

array addition

```
>> u = [1 2 3];
>> v = [7 8 9];
>> w = u + v
w =
     8    10    12

>> A = [1 2 3;
        4 5 6];
>> B = [6 5 4;
        3 2 1];
>> C = A + B
C =
     7     7     7
     7     7     7
```

array subtraction

```
>> u = [1 2 3];
>> v = [7 8 9];
>> w = u – v
w =
    -6    -6    -6

>> A = [1 2 3;
        4 5 6];
>> B = [6 5 4;
        3 2 1];
>> C = A – B
C =
    -5    -3    -1
     1     3     5
```

```
>> u = [1 2 3];
>> v = [7 8 9];
>> w = u .* v
w =
     7    16    27

>> A = [1 2 3;
        4 5 6];
>> B = [6 5 4;
        3 2 1];
>> C = A .* B
C =
     6    10    12
    12    10     6
```

array multiplication

in mathematics, called the Hadamard product or the Schur product

```
>> u = [1 2 3];
>> v = [7 8 9];
>> w = u ./ v
w =
    0.1429    0.2500    0.3333

>> A = [1 2 3;
        4 5 6];
>> B = [6 5 4;
        3 2 1];
>> C = A ./ B
C =
    0.1667    0.4000    0.7500
    1.3333    2.5000    6.0000
```

right array division

the elements in **u** divided by the elements in **v**

the elements in **A** divided by the elements in **B**

```
>> u = [1 2 3];
>> v = [7 8 9];
>> w = u .\ v
w =
    7     4     3


>> A = [1 2 3;
        4 5 6];
>> B = [6 5 4;
        3 2 1];
>> C = A .\ B
C =
    6.0000    2.5000    1.3333
    0.7500    0.4000    0.1667
```

left array division

the elements in **v** divided by the elements in **u**

the elements in **B** divided by the elements in **A**

Arithmetic operations with arrays

- you can perform element-by-element arithmetic with an array and a scalar

operator	name
`+`	addition
`-`	subtraction
`*`	multiplication
`/`	right division
`\`	left division
`.^`	array power

```
>> u = [1 2 3];
>> w = 2 + u
w =
    3     4     5

>> A = [1 2 3;
        4 5 6];
>> C = A + 10
C =
    11    12    13
    14    15    16
```

array scalar addition

array scalar subtraction

```
>> u = [1 2 3];
>> w = 2 - u
w =
    1     0     -1

>> A = [1 2 3;
        4 5 6];
>> C = A - 10
C =
    -9    -8    -7
    -6    -5    -4
```

```
>> u = [1 2 3];
>> w = 2 * u
w =
    2     4     6

>> A = [1 2 3;
        4 5 6];
>> C = A * 10
C =
    10    20    30
    40    50    60
```

array scalar multiplication

array scalar division

```
>> u = [1 2 3];
>> w = u / 2
w =
    0.5000    1.0000    1.5000

>> A = [1 2 3;
        4 5 6];
>> C = 10 \ A
C =
    0.1000    0.2000    0.3000
    0.4000    0.5000    0.6000
```

array power

```
>> u = [1 2 3];
>> w = u .^ 2
w =
    1     4     9

>> A = [1 2 3;
        4 5 6];
>> C = A .^ 2
C =
     1     4     9
    16    25    36
```

Example: Gaussian elimination

- see http://en.wikipedia.org/wiki/Gaussian_elimination#Example_of_the_algorithm

```
>> A = [2 1 -1;
       -3 -1 2;
       -2 1 2]

A =
     2     1    -1
    -3    -1     2
    -2     1     2

>> x = [8; -11; -3]

x =
     8
   -11
    -3
```

```
>> B = [A x]  % the augmented matrix [A | x]

B =

     2     1    -1     8
    -3    -1     2   -11
    -2     1     2    -3

>> B(2, :) = B(2, :) + (3 / 2) * B(1, :)

B =
    2.0000    1.0000   -1.0000    8.0000
         0    0.5000    0.5000    1.0000
   -2.0000    1.0000    2.0000   -3.0000
```

```
>> B(3, :) = B(3, :) + B(1, :)

B =
    2.0000    1.0000   -1.0000    8.0000
         0    0.5000    0.5000    1.0000
         0    2.0000    1.0000    5.0000

>> B(3, :) = B(3, :) - 4 * B(2, :)

B =
    2.0000    1.0000   -1.0000    8.0000
         0    0.5000    0.5000    1.0000
         0         0   -1.0000    1.0000
```

keep following the Wikipedia example to get the row reduced echelon form

Example: Gaussian elimination

- you could also use the MATLAB function **`rref`**

```
>> rref(B)   % row reduced echelon form of B

ans =

     1     0     0     2
     0     1     0     3
     0     0     1    -1
```

Scripts

MATLAB Scripts

- a script is text file containing a *sequence* of MATLAB commands
  - each command usually occurs on a separate line of the file
- MATLAB can run the commands in a script by reading the file and interpreting the text as MATLAB commands
  - commands are run in order that they appear in the script file

MATLAB Scripts

- the filename of a MATLAB script always has the following form:

`yourScriptName.m`

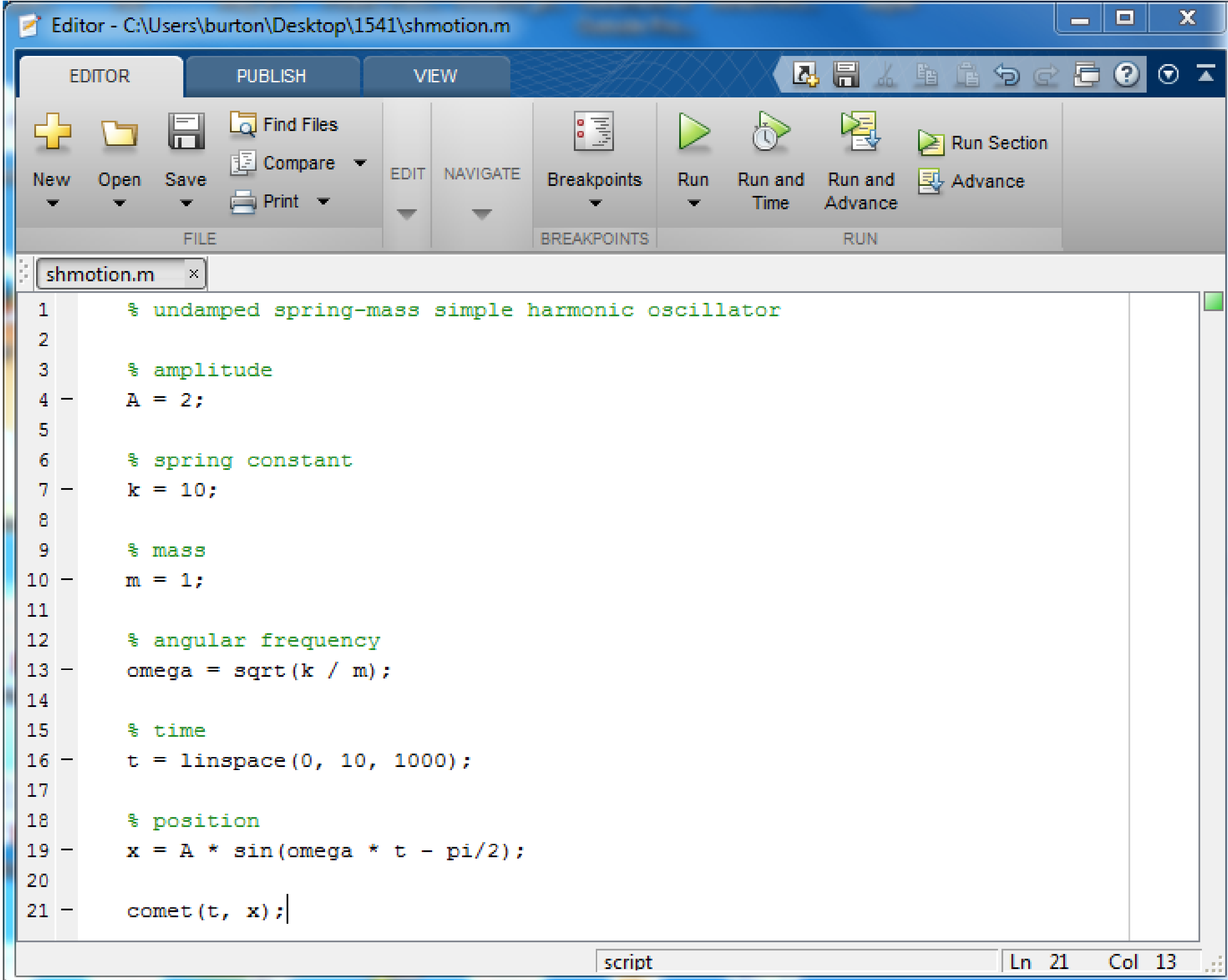
where **`yourScriptName`** must be a valid MATLAB variable name

- i.e., must begin with a letter and may only contain letters and spaces and underscores
  - no spaces or symbols!

Script example

- an undamped spring-mass system is an example of a simple harmonic oscillator
- the position of the mass is given by

$$x(t) = A \sin\left(\sqrt{\frac{k}{m}}\,t - \frac{\pi}{2}\right)$$


Editor - C:\Users\burton\Desktop\1541\shmotion.m
EDITOR
PUBLISH
VIEW
Find Files
Compare
Print
New
Open
Save
FILE
EDIT
NAVIGATE
Breakpoints
BREAKPOINTS
Run
Run and Time
Run and Advance
Run Section
Advance
RUN
shmotion.m
% undamped spring-mass simple harmonic oscillator
% amplitude
A = 2;
% spring constant
k = 10;
% mass
m = 1;
% angular frequency
omega = sqrt(k / m);
% time
t = linspace(0, 10, 1000);
% position
x = A * sin(omega * t - pi/2);
comet(t, x);
script
Ln 21
Col 13

MATLAB Scripts

- MATLAB will "run" the script if you type in the name of the script in the command window
  - the script must saved in a folder that is on the current MATLAB path
    - the current MATLAB path always includes the current working folder shown the MATLAB address bar
- you will find it useful to organize all of your scripts and functions in a common folder
  - see the **`path`** command (and its related functions)

MATLAB Scripts

- a script can create new variables, or it can re-use existing variables in the workspace
  - note: this means that a script can overwrite an existing variable in the workspace, too