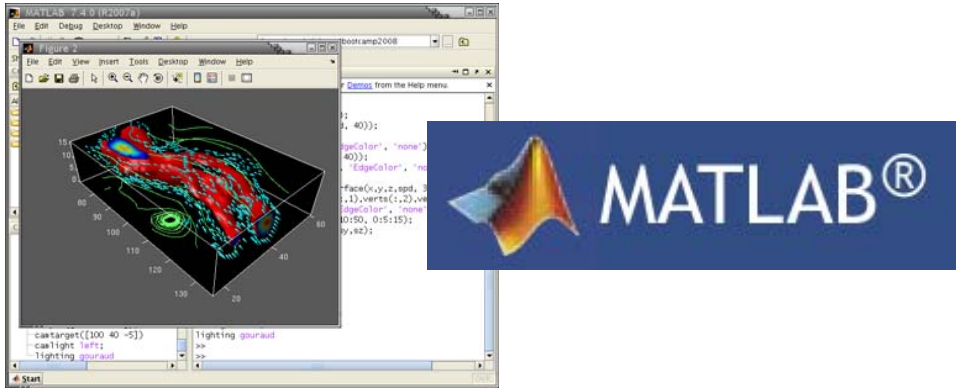
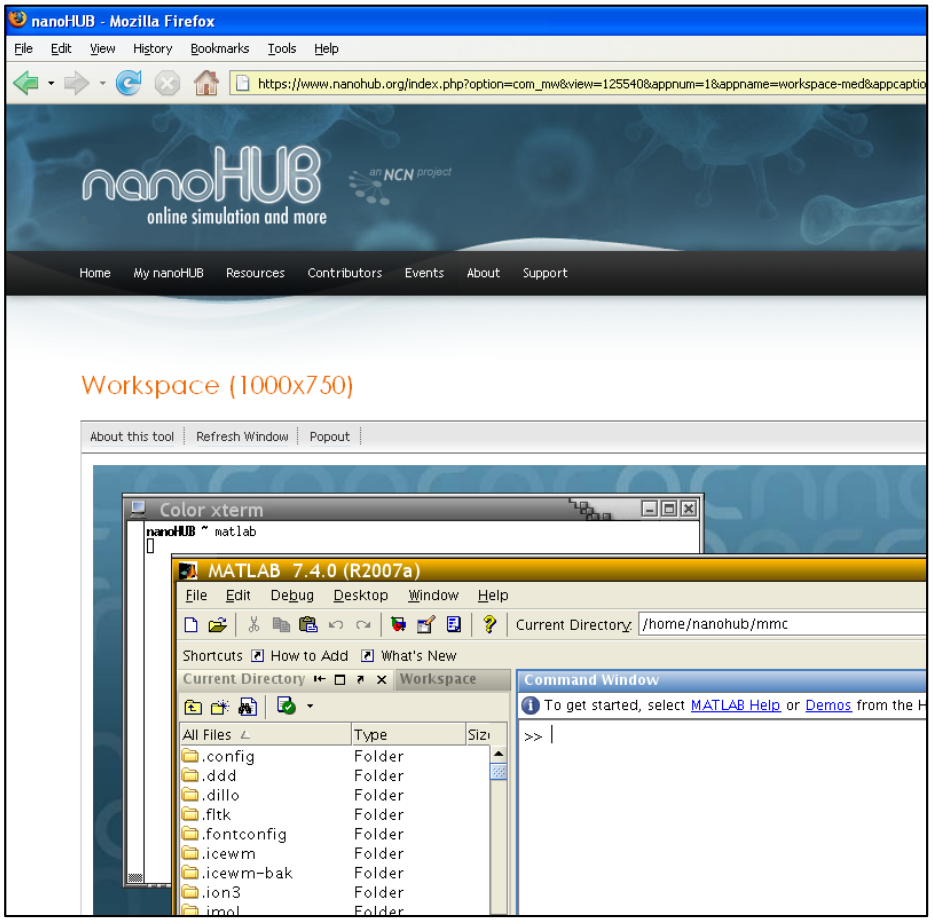
Introduction to Scientific Programming in MATLAB



Michael McLennan
Software Architect
HUBzero™ Platform for Scientific Collaboration

Accessing MATLAB

Start a workspace and type `matlab`



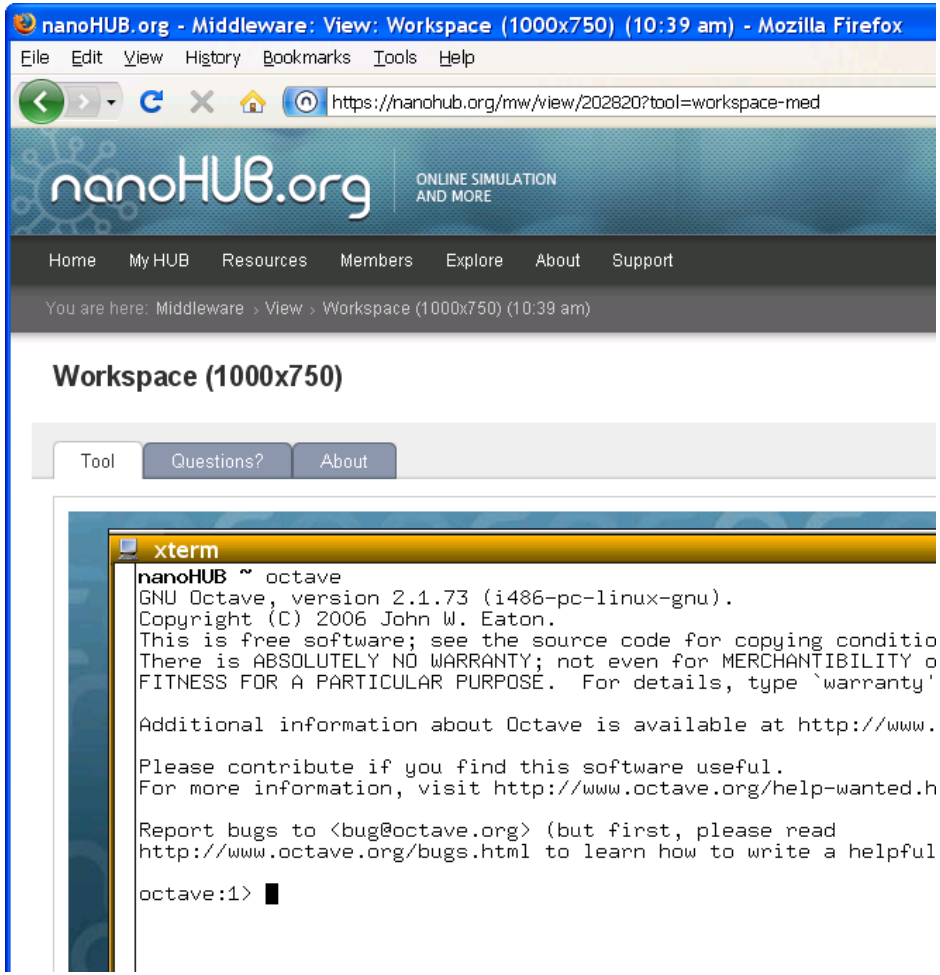
NOTE: MATLAB requires a license from Purdue, so this works only when accessing from Purdue's campus.

Try some simple commands:

```
>> x=1
>> y=2
>> z=2*x + y
>> exit
```

Free Clone: GNU Octave

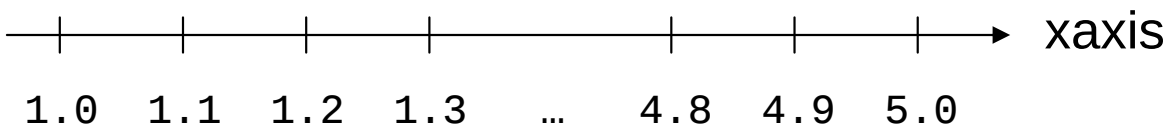
In your workspace, type `octave`



- ☑ Runs almost all MATLAB scripts
- ☑ Creates plots
- ☒ Missing fancy debugger
- ☒ Missing “simulink” toolbox

Using Vectors

Create a series of number like this:




```
xaxis = 1:0.1:5
```

start with this
increment by this
end with this

```
xaxis = linspace(1,5,41)
```

start with this
end with this
number of points

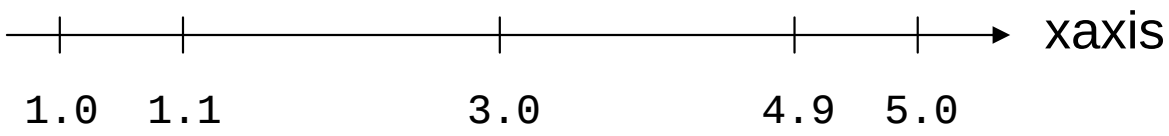
Try this:

```
>> xaxis = 1:0.1:5;
>> length(xaxis)
>> xaxis
```

Try it with/without the semicolon.
What does the semicolon do?

More Vectors

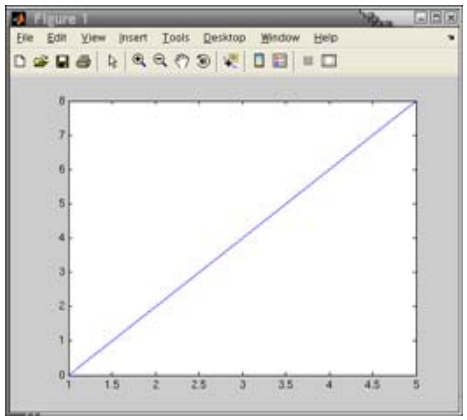
Suppose you have an irregular spacing:




```
xaxis = [1.0, 1.1, 3.0, 4.9, 5.0]
```

square brackets

comma-separated list of values



Try this:

```
>> y = 2*xaxis - 2
>> plot(xaxis,y)
>> y = 3*xaxis*xaxis + 1
>> y = 3*xaxis.*xaxis + 1
```

Multiply element by element:
[1.0 1.1 …] [1.0 1.1 …]

Matrices

Suppose you want to define a matrix, like this:

$$a = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

in MATLAB, that's

```
>> a = [-1 -1 -1 ; 0 0 0 ; 1 1 1]
```

$$a^T = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

in MATLAB, that's

```
>> a'
```

Try this:

```
>> a * a'
>> a' * a
>> a' * a + 1
```

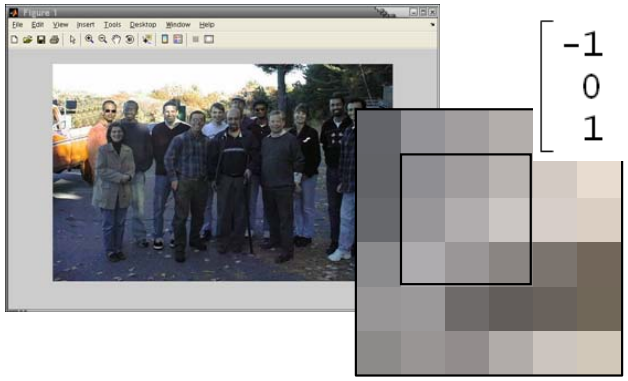
Try these built-in functions:

```
>> zeros(3)
>> ones(4)
>> eye(2)
>> a * a' + eye(3)
```

Image Processing

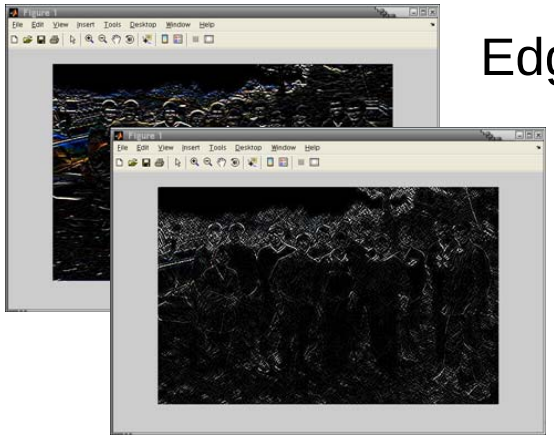
Try this:

```
>> im = imread('/apps/matlab/7.5/toolbox/simulink/simulink/simulinkteam.jpg');
>> figure;
>> imshow(im);

>> a = [-1 -1 -1 ; 0 0 0 ; 1 1 1]
>> im2 = imfilter(im, a, 'conv');
>> imshow(im2);

>> im2 = imfilter(im, a*a', 'conv');
>> imshow(im2);
```



$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

'conv' operation



Edge detection!

charcoal effect

Octave Users:
Works in Octave 3.0
For earlier versions, download imread from

http://www.cs.helsinki.fi/u/ahyvarin/kurssi/imread.m

Matrix Indexing

$$a = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

in MATLAB, that's

```
>> a = [-1 -1 -1 ; 0 0 0 ; 1 1 1]
```

Try this:

```
>> a(1,1)
>> a(2,1)
>> a(1,2)
```

column index
row index

Try this:

```
>> a(1,1:3)
>> a(1,:)
```

whole row

Try this:

```
>> a(2,2) = 3
>> a
```

Pick apart image pixel by pixel…

```
>> im(1,5)
>> im(3, :)
>> im(1:100, 1:100)
```

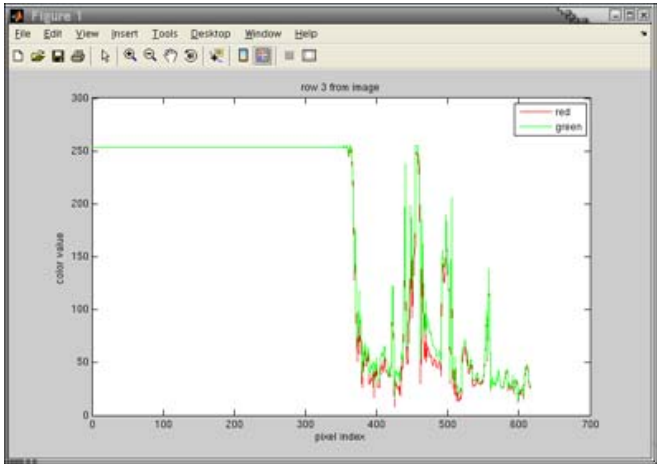
Back to Plotting

Try this:

```
>> rv = im(3, :, 1)
```

first component: r g b
all column elements
row index

```
>> plot(1:length(rv), rv, 'r')
>> title('row 3 from image')
>> xlabel('pixel index')
>> ylabel('color value')
>> hold
>> gv = im(3, :, 2)
>> plot(1:length(gv), gv, 'g')
>> legend('red','green')
```

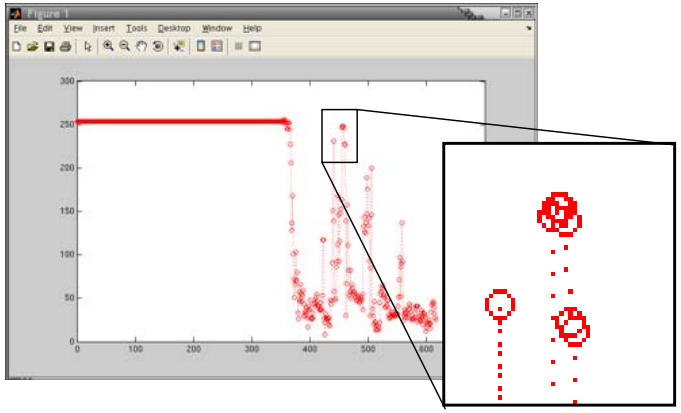


Plotting Options

Try this:

```
>> clf
>> plot(1:length(rv), rv, 'r:o')
```

red, dotted line, circles



r	Red	-	Solid line (default)	+	Plus sign
g	Green	--	Dashed line	o	Circle
b	Blue	:	Dotted line	*	Asterisk
c	Cyan	-.	Dash-dot line	.	Point
m	Magenta			x	Cross
y	Yellow			s	Square
k	Black			d	Diamond
w	White			^	Upward-pointing triangle
				v	Downward-pointing triangle
				>	Right-pointing triangle
				<	Left-pointing triangle
				p	Five-pointed star (pentagram)
				h	Six-pointed star (hexagram)

Functions

file: edgematrix.m

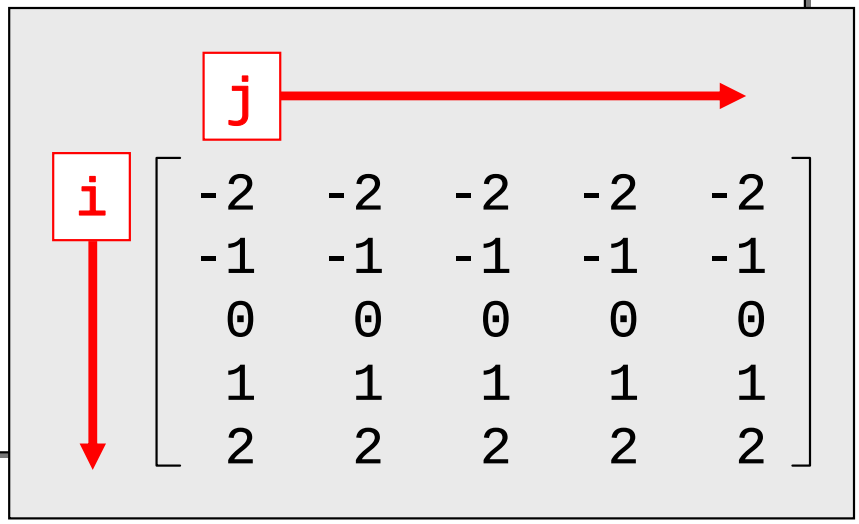
```
% returns edge detection matrix of size n
% orient is 'horz' for horizontal edges, and 'vert' for vertical
function m = edgematrix(n,orient)
  half = floor(n/2);
  n = 2*half + 1;
  for i=1:n
    for j=1:n
      if strcmp(orient,'horz')
        m(i,j) = j-half-1;
      else
        m(i,j) = i-half-1;
      end
    end
  end
```

```
im2 = imfilter(im, edgematrix(5,'horz'), 'conv');
```

Loops

file: edgematrix.m

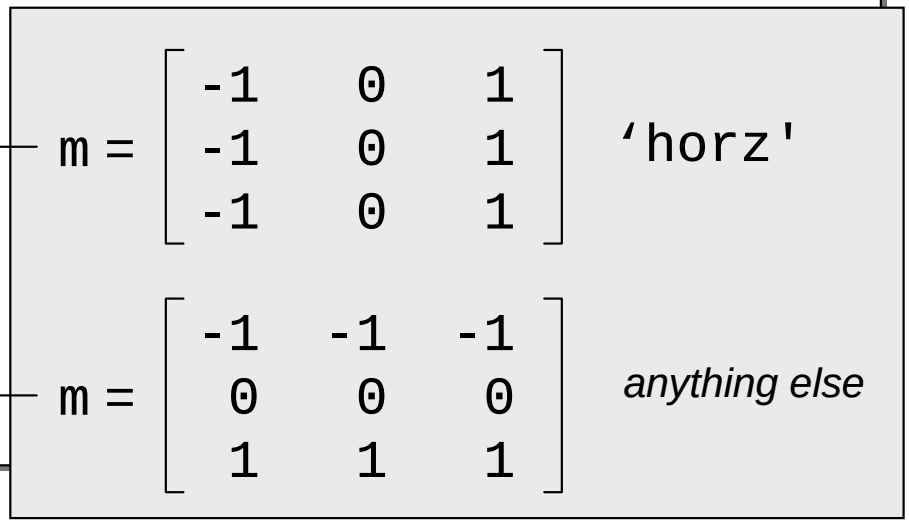
```
% returns edge detection matrix of size n
% orient is 'horz' for horizontal edges, and 'vert' for vertical
function m = edgematrix(n,orient)
  half = floor(n/2);
  n = 2*half + 1;
  for i=1:n
    for j=1:n
      if strcmp(orient,'horz')
        m(i,j) = j-half-1;
      else
        m(i,j) = i-half-1;
      end
    end
  end
```



Conditionals

file: `edgematrix.m`

```
% returns edge detection matrix of size n
% orient is 'horz' for horizontal edges, and 'vert' for vertical
function m = edgematrix(n,orient)
  half = floor(n/2);
  n = 2*half + 1;
  for i=1:n
    for j=1:n
      if strcmp(orient,'horz')
        m(i,j) = j-half-1;
      else
        m(i,j) = i-half-1;
      end
    end
  end
```



Programming the MATLAB Way

file: edgematrix.m

```
% returns edge detection matrix of size n
% orient is 'horz' for horizontal edges, and 'vert' for vertical
functi   m = edg   trix(n,orient)
  half   loor   /2);
  n = 2*h   +1;
  for i=1:
    for
```

```
% returns edge detection matrix of size n
% orient is 'horz' for horizontal edges, and 'vert' for vertical
function m = edgematrix2(n,orient)
  half = floor(n/2);
  o = ones(2*half + 1);
  m = [-half:half]' * o(1,:)
  if strcmp(orient,'horz')
     m = m'
  end
```

$$\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Simple Input/Output

file: `hello.m`

```
% prompt for a phrase and say "hello"
disp('Who are you?');
name = input('Enter your name: ','s');
age = input('Enter your age: ');

mesg = sprintf('Hello, %s!', name);
disp(mesg);

file = input('Enter a file name: ','s');
fid = fopen(file,'w');
fprintf(fid, '%s is %d years old\n', name, age);
fclose(fid);
```

String input

Numbers and other stuff

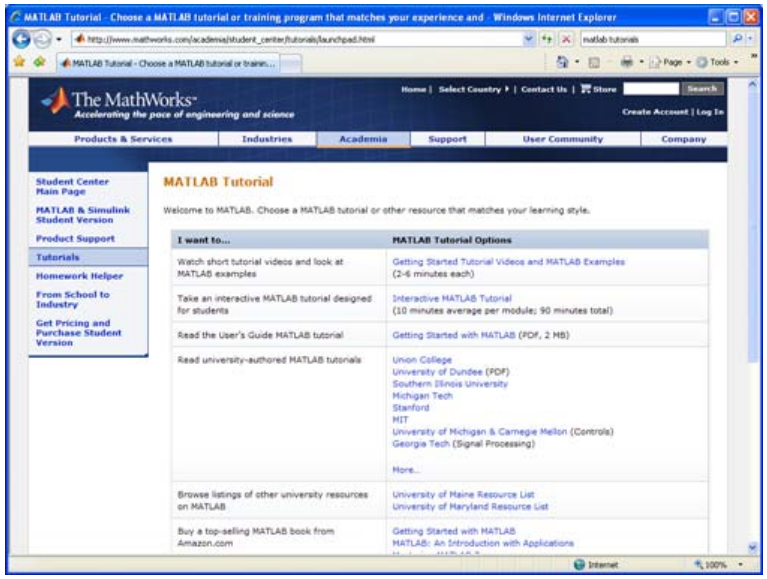
Use script name as a command to invoke the script

```
>> hello
Who are you?
Enter your name: Michael
Enter your age: 43
Hello, Michael!
Enter a file name: info.txt
```

Other Resources

Tutorials at MathWorks

http://www.mathworks.com/academia/student_center/tutorials/launchpad.html



MATLAB DOs and DONTs

http://nanohub.org/resources/1279