Introduction to MATLAB

Exercises Multiple View Geometry

April 29th, 2013

Julia Bergbauer

Why MATLAB?

- Numerical Computing Environment
- Allows Matrix Manipulations
- Plotting of Functions and Data
- Widely used in Academic and Research Institution

MATLAB Student Version

matlab.rbg.tum.de

↓

Login using your MyTUM-Account

Matrices and Vectors

```
zeros
eye
ones
rand
```

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

```
repmat(A,n,m)
[eye(2);ones(2)]
[eye(2) rand(2)]
A'
```

Useful Commands

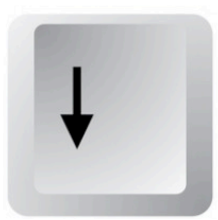
`Ctrl+C`

Help browser

`doc`

`clc`

`help`



`clear`

`,` `;`

Vectors

```
v = 1:1:10        ⇔        v = 1:10
v = 1:-3:-10

v  =   START   :   STEPSIZE   :   END


v = linspace(1,-10,3)

v = linspace(START , END , nElements)
```

Operators

Element-Wise Operations !!

`.*` `./` `.^`

$$\begin{pmatrix} 1 & 2 & 2 \\ 0 & 2 & 2 \end{pmatrix} \;.*\; \begin{pmatrix} 5 & 0 & 2 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 5 & 0 & 4 \\ 0 & 2 & 0 \end{pmatrix}$$

```
  A      .*     B      =     C
(mxn)         (mxn)        (mxn)
```

Operators

Element-Wise Operations !!

.*   ./   .^

$$\begin{pmatrix} 2 \\ 3 \\ 4 \end{pmatrix} .* \begin{pmatrix} 2 \\ 3 \\ 4 \end{pmatrix} \quad \text{vs.} \quad \begin{pmatrix} 2 & 3 & 4 \end{pmatrix} * \begin{pmatrix} 2 \\ 3 \\ 4 \end{pmatrix}$$

Element-Wise Operations

$$A = \begin{pmatrix} 1 & 2 & 2 \\ 0 & 2 & 2 \end{pmatrix}$$

$$B = \begin{pmatrix} 5 & 0 \\ 0 & 1 \\ 2 & 0 \end{pmatrix}$$

$$c = \begin{pmatrix} 2 & 3 & 4 \end{pmatrix}$$

```
A^2
A + 5
A - 2
A * 5
A / 2
B * A
c * B
A * B
A * c`
```

Element-Wise Operations

$$A = \begin{pmatrix} 1 & 2 & 2 \\ 0 & 2 & 2 \end{pmatrix}$$

$$B = \begin{pmatrix} 5 & 0 \\ 0 & 1 \\ 2 & 0 \end{pmatrix}$$

$$c = \begin{pmatrix} 2 & 3 & 4 \end{pmatrix}$$

```
A.^2
A + 5
A - 2
A * 5
A / 2
B * A
c * B
A * B
A * c`
```

Functions

`sum` `min` `abs` `sin` `sort` `cos` `max` `exp` `floor` `sqrt` `numel` `log` `length` `size`

`length(A(:))` `x(i)` `A(1,end)` `A(i,j)`

Indexing

```
x = [-3 -2 -1 0 1 2 3]
M = [1 2; 2 1]
A = [1 2; 3 4; 5 6]
B = zeros(3)
```

```
x(2:4)                      A(2:end,:)
x([2 3 end end 1])          A(1,:)
x(M)                        A(:,2)
                            B(2:2:end) = 1
```

Operators

```
&    |    ~
~=
==
>
<
<=
>=
```

```
A = rand(2)

A<=0.2 | A>=0.8

L = A>0.5

f = find(A>0.5)

A(f)
```

Functions

File name: `funcname.m`

```
function [output_args] = funcname(input_args)
...







end
```

Functions

File name: `funcname.m`

```
function [output_args] = funcname(input_args)
...
    for i = 1:5
       ...
    end

    while (i < 5)
       ...
    end

    if (...)
       ...
    elseif (...)
       ...
    else
       ...
    end
end
```

Anonymous Functions and Plots

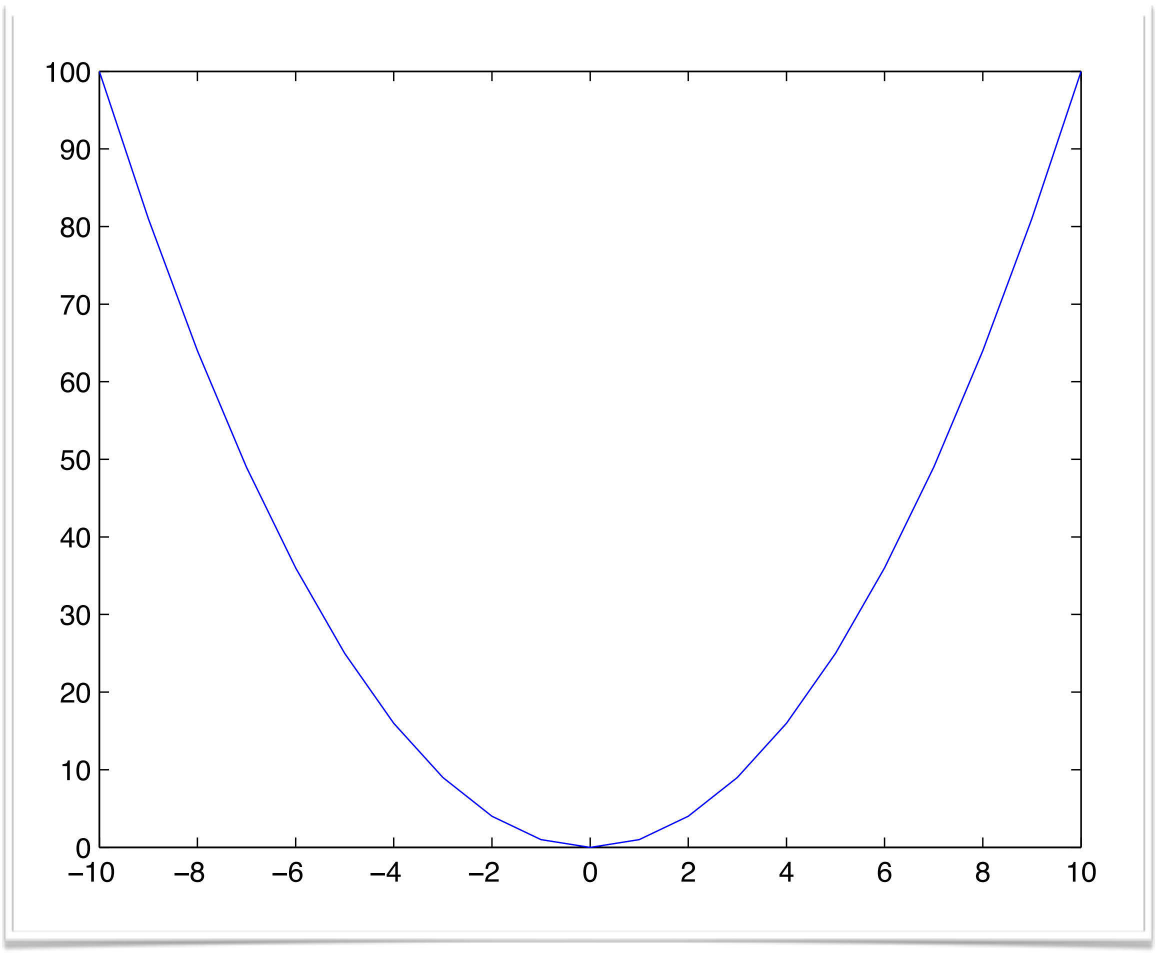
```
f = @(x) x^2
f(5)

x = -10:1:10
plot(x,f(x))
```

Anonymous Functions and Plots

```
f = @(x) x.^2
f(5)

x = -10:1:10
plot(x,f(x))
```



Anonymous Functions and Plots

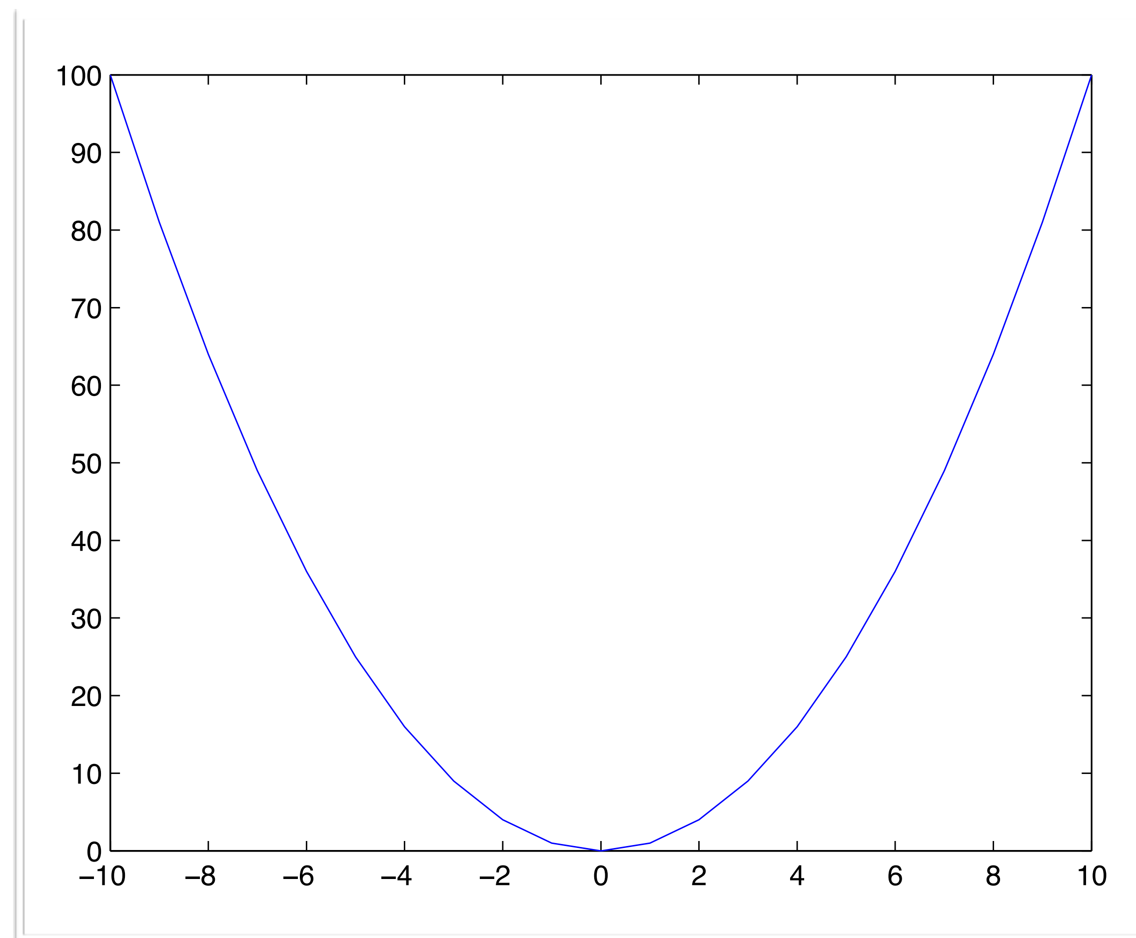
```
f = @(x) x.^2

f(5)

x = -10:1:10

plot(x,f(x))
```

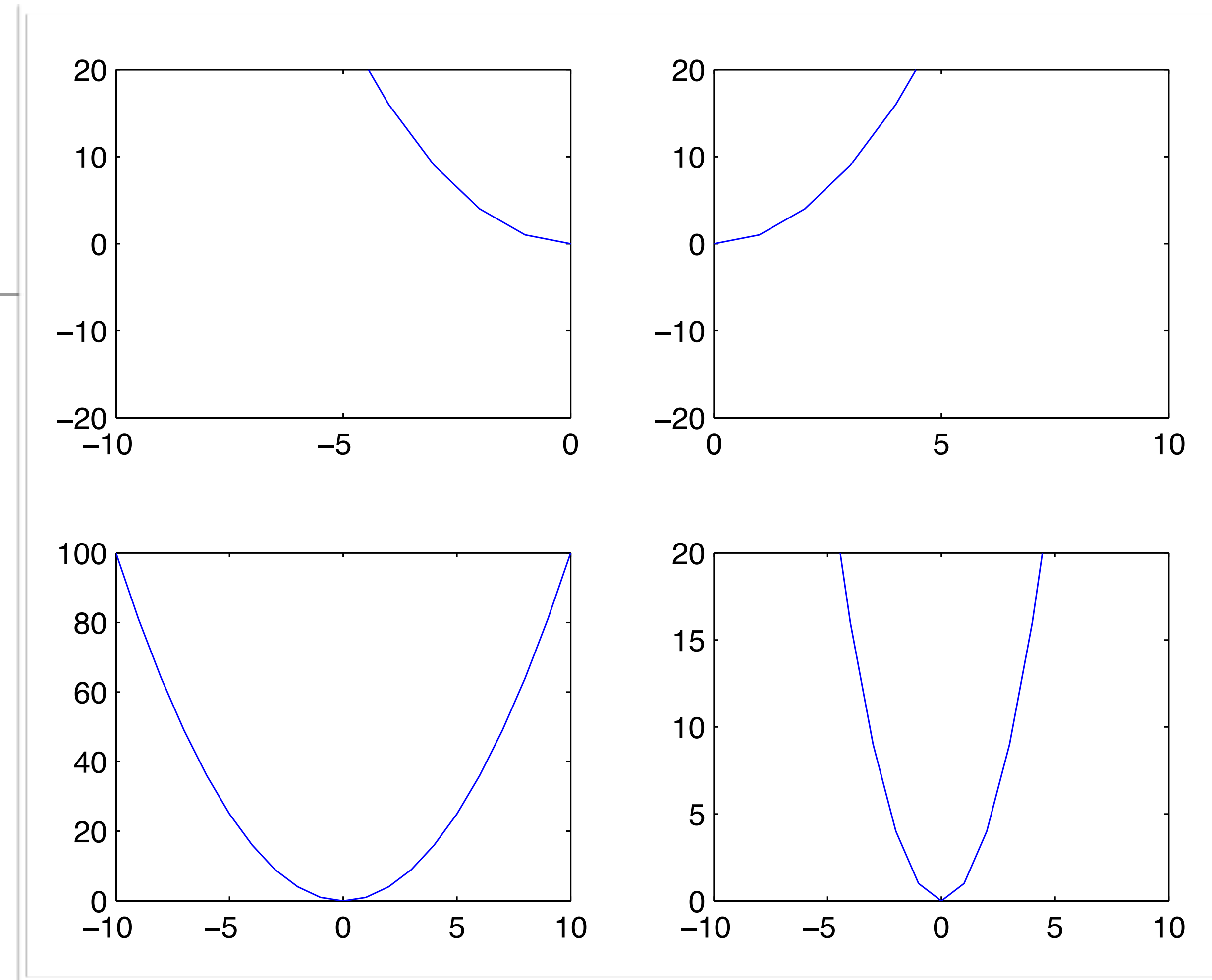
'r' '--r' '-.g'




```
axis([xmin xmax ymin ymax])

hold on
```

Subplot




```
figure(1)
hold on

subplot(2,2,1)
plot(..)
axis([xmin xmax ymin ymax])

subplot(2,2,2) , plot(..)
...
```

Questions?

https://vision.in.tum.de/teaching/ss2013/mvg2013

Julia Bergbauer
julia.bergbauer@tum.de

MATLAB Kurzhilfe

http://www-m1.ma.tum.de/foswiki/pub/M1/Lehrstuhl/BorisVonLoesch/matlabCS.pdf

Exercise 1

Write a function `approxequal(x, y, eps)`

comparing two vectors *x* and *y* if they are almost equal, i.e.:

$$\text{for all indices } i: \quad || x_i - y_i || \leqslant eps$$

The output should be logical 1 or 0.

If the input consists of two matrices, your function should compare the columns of the matrices if they are almost equal.

In this case, the output should be a vector with logical values 1 or 0.

Exercise 1 - Solution

```
function l = approxequal(x, y, eps)
    l = all(abs(x-y) <= eps);
end
```

Exercise 2

Write a function `addprimes(s, e)`

returning the sum of all prime numbers between *s* and *e*.

Use the Matlab-function `isprime`.

Exercise 2 - Solution

```
function out = addprimes(s, e)
    z = s:e;
    out = sum(z(isprime(z)));
end
```