TECHNISCHE
UNIVERSITÄT
DRESDEN

Introduction to Matlab

Advanced Plotting, Control Flow Statements Functions & Integration

Pouyan R. Fard

Dresden, 18.11.2016

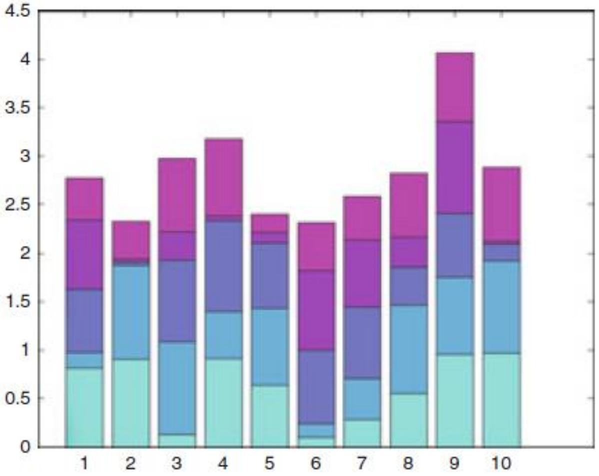
DRESDEN
concept
Exzellenz aus
Wissenschaft
und Kultur

Today’s Plan

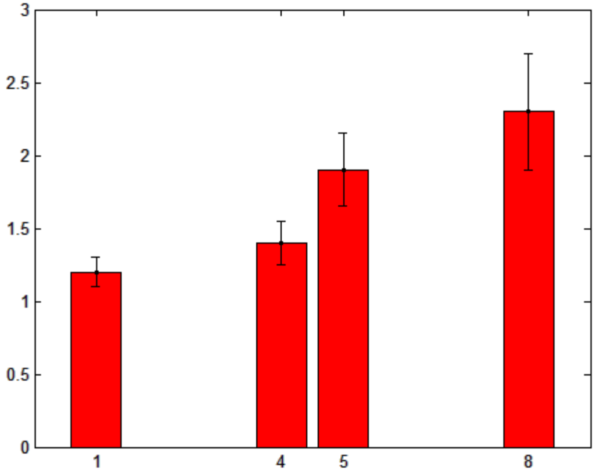
Date	Topics	Exercise/Project
14.10	Kick-off presentation	
21.10	Intro, basic operations	In-class exercise
	Data Handling: vectors, matrices, variables	
4.11	Basic and advanced plotting	In-class exercise
	Scripts and functions	
18.11	Control Flow statements	In-class exercise
	Debugging and integration	
9.12	Data analysis and statistics	In-class exercise Project Distribution
	Sound, images and videos	
20.01	Experimental stimuli and GUI	In-class exercise Project Deadline
	Project Presentation	

Bar Plots

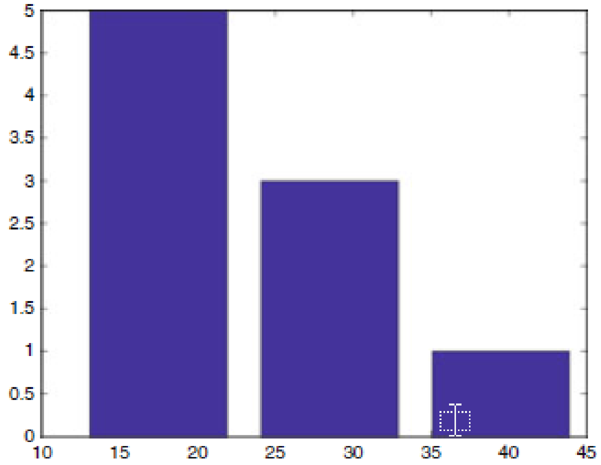
```
>> bar(rand(10,5),'stacked');
>> colormap(cool);
```



```
x=[1,4,5,8];
RT=[1.2,1.4,1.9,2.3];
SD=[0.1,0.15,0.25,0.4];
bar(x,RT,'w'); hold on;
errorbar(x,RT,SD,'.k');
```

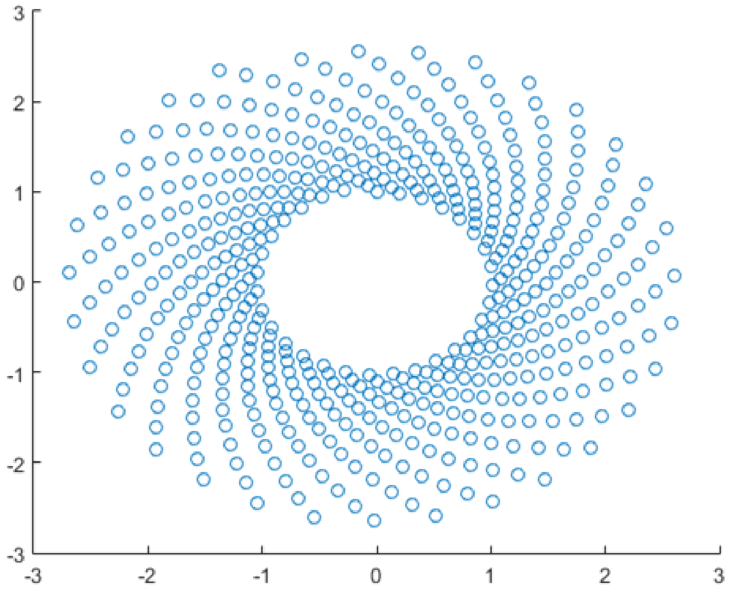


```
Ages=[22,25,23,22,45,12,34,33,21];
[N,xc]=hist(Ages,3);
bar(xc,N);
```

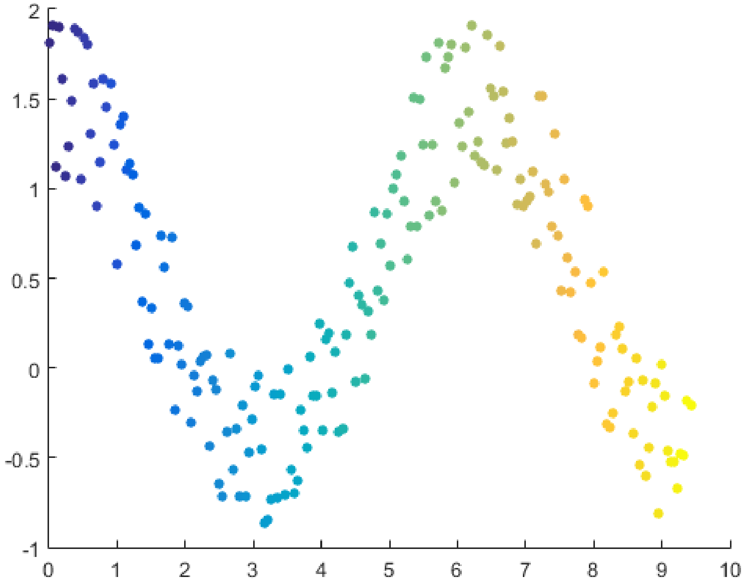


Scatter Plots

```
theta = linspace(0,1,500);
x = exp(theta).*sin(100*theta);
y = exp(theta).*cos(100*theta);
s = scatter(x,y);
```

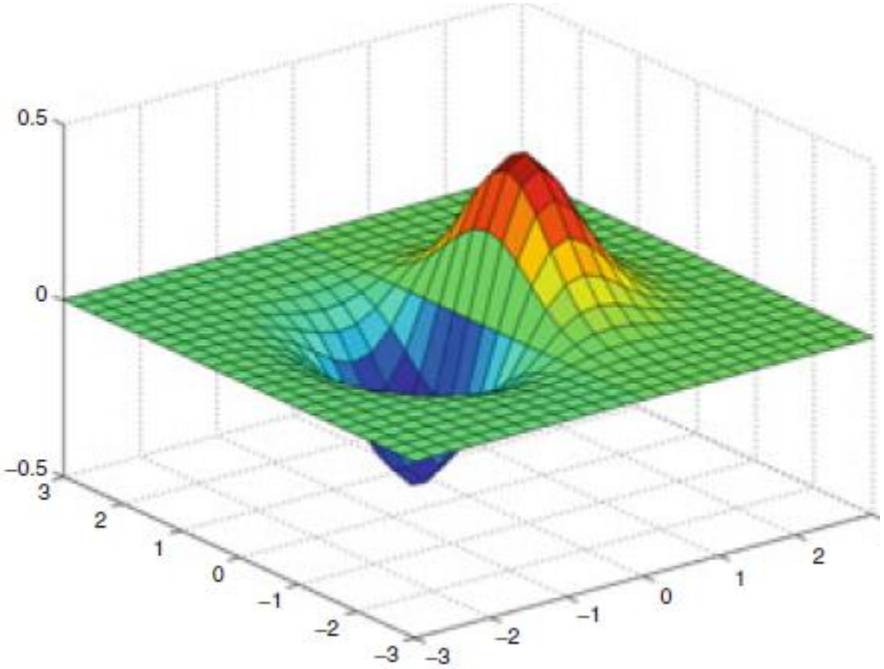


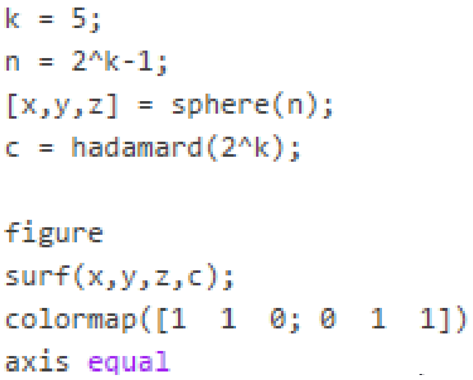
```
x = linspace(0,3*pi,200);
y = cos(x) + rand(1,200);
a = 25;
c = linspace(1,10,length(x));
scatter(x,y,a,c,'filled')
```



3D Plotting

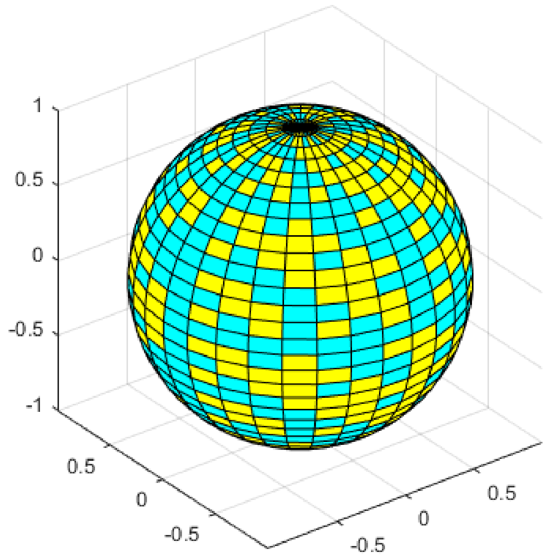
```
>> a=[-3:0.25:3];
>> b=[-3:0.25:3];
>> [X,Y]=meshgrid(a,b);
>> Z= X.*exp(-X.^2-Y.^2);
>> surf(X,Y,Z);
```





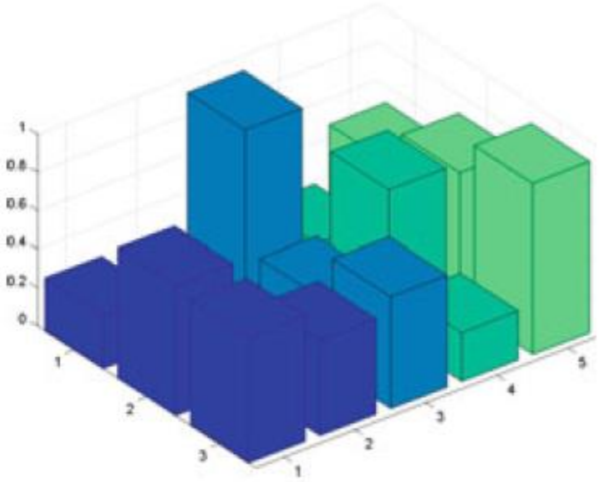
```
k = 5;
n = 2^k-1;
[x,y,z] = sphere(n);
c = hadamard(2^k);

figure
surf(x,y,z,c);
colormap([1  1  0; 0  1  1])
axis equal
```

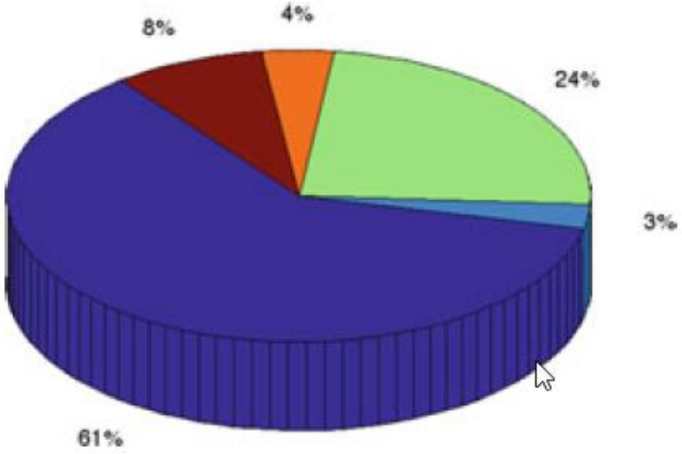


3D Plotting

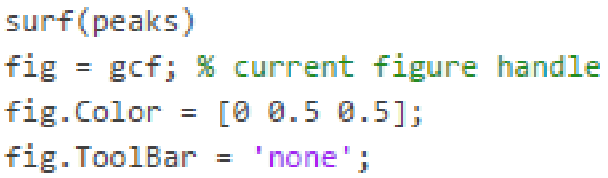
```
>> y=rand(3,5);
>> bar3(y);
>> colormap(winter);
```



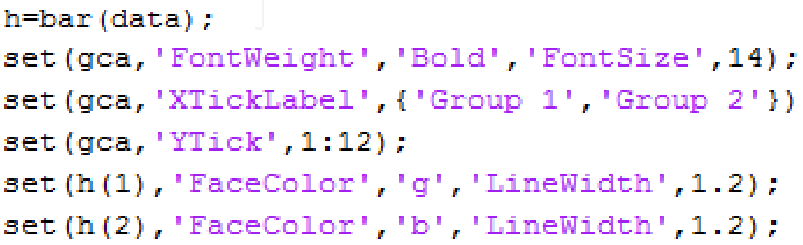
```
>> y=rand(5,1);
>> pie3(y);
>> axis square; grid off;
```



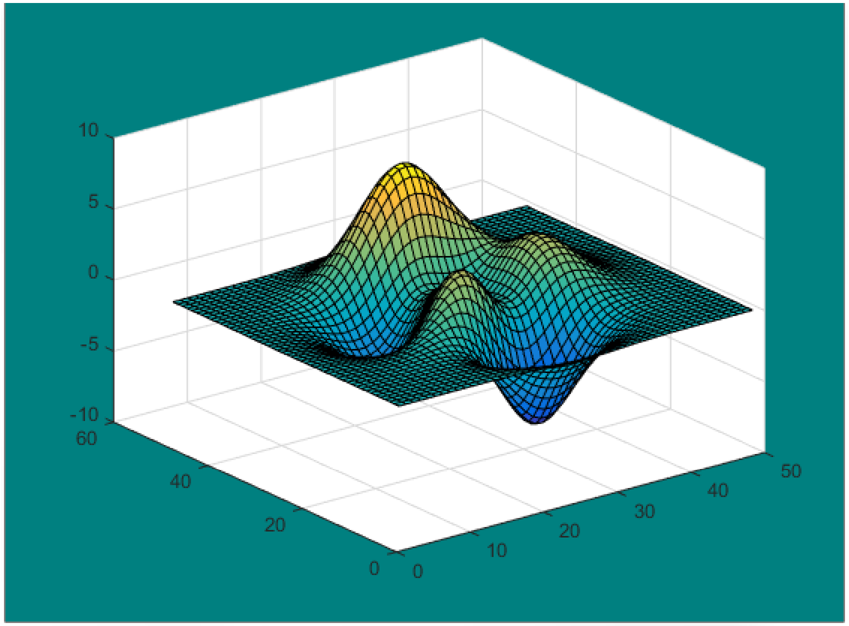
Graphics Handles

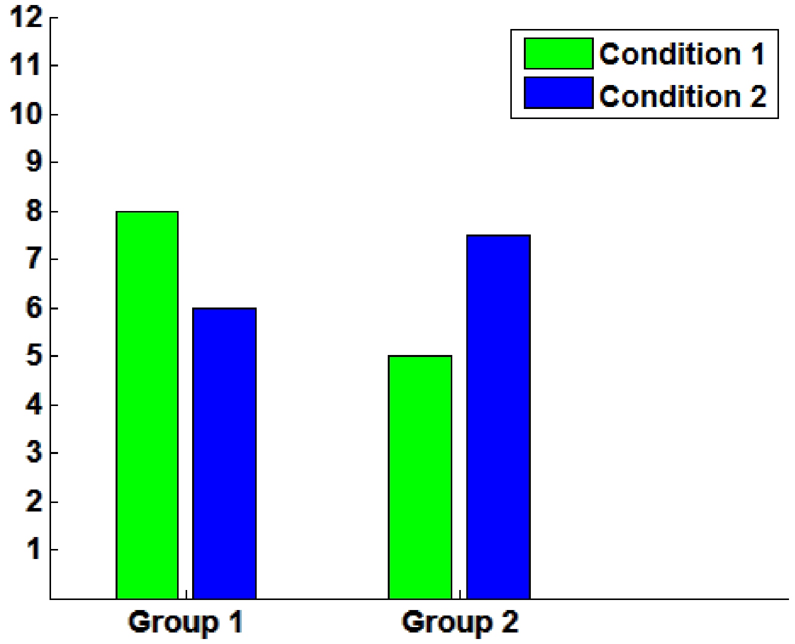


```
surf(peaks)
fig = gcf; % current figure handle
fig.Color = [0 0.5 0.5];
fig.ToolBar = 'none';
```



```
h=bar(data);
set(gca,'FontWeight','Bold','FontSize',14);
set(gca,'XTickLabel',{'Group 1','Group 2'})
set(gca,'YTick',1:12);
set(h(1),'FaceColor','g','LineWidth',1.2);
set(h(2),'FaceColor','b','LineWidth',1.2);
```



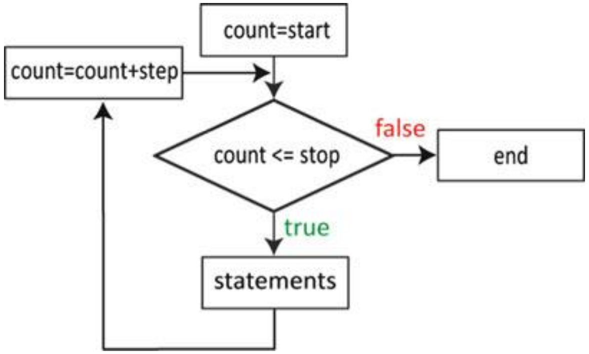


Control Flow Statements

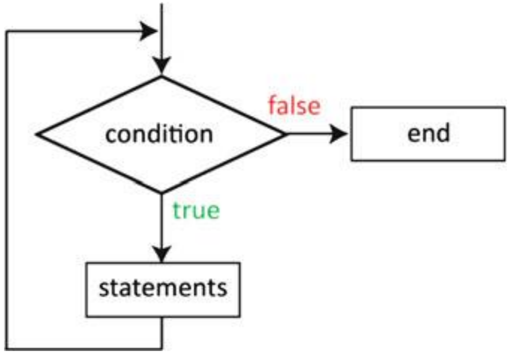
```
if condition1
        Statements1
elseif condition2
        Statements2
elseif condition3
        Statements3
else
        Statements4
end
```

```
switch condition
    case fact1
            Statements1
    case fact2
            Statements2
    case fact3
            Statements3
    otherwise
            StatementsOtherwise
end
```

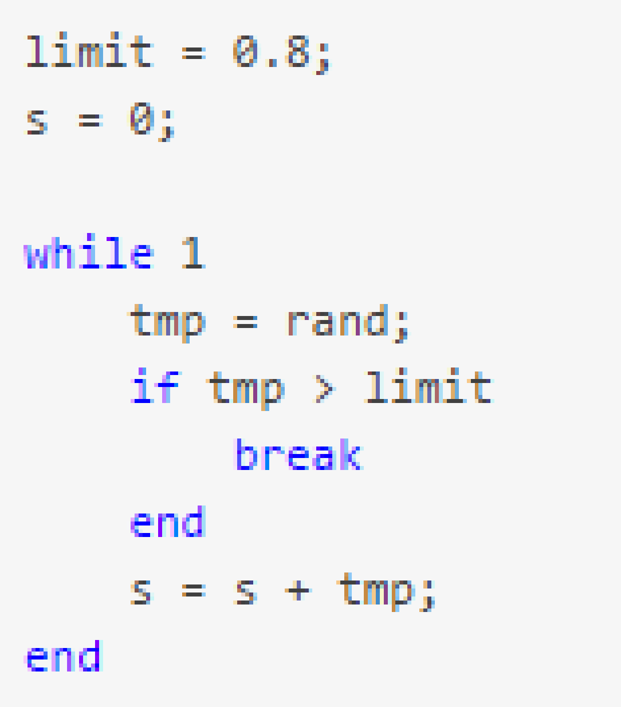
```
for count = start:step:stop;
    statements
end;
```



```
while condition
  statements
end
```

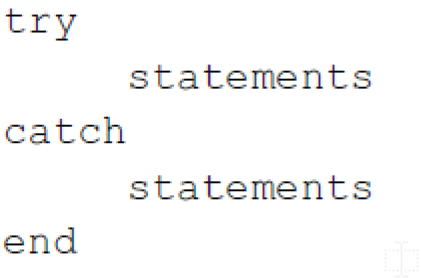


Control Flow Statements



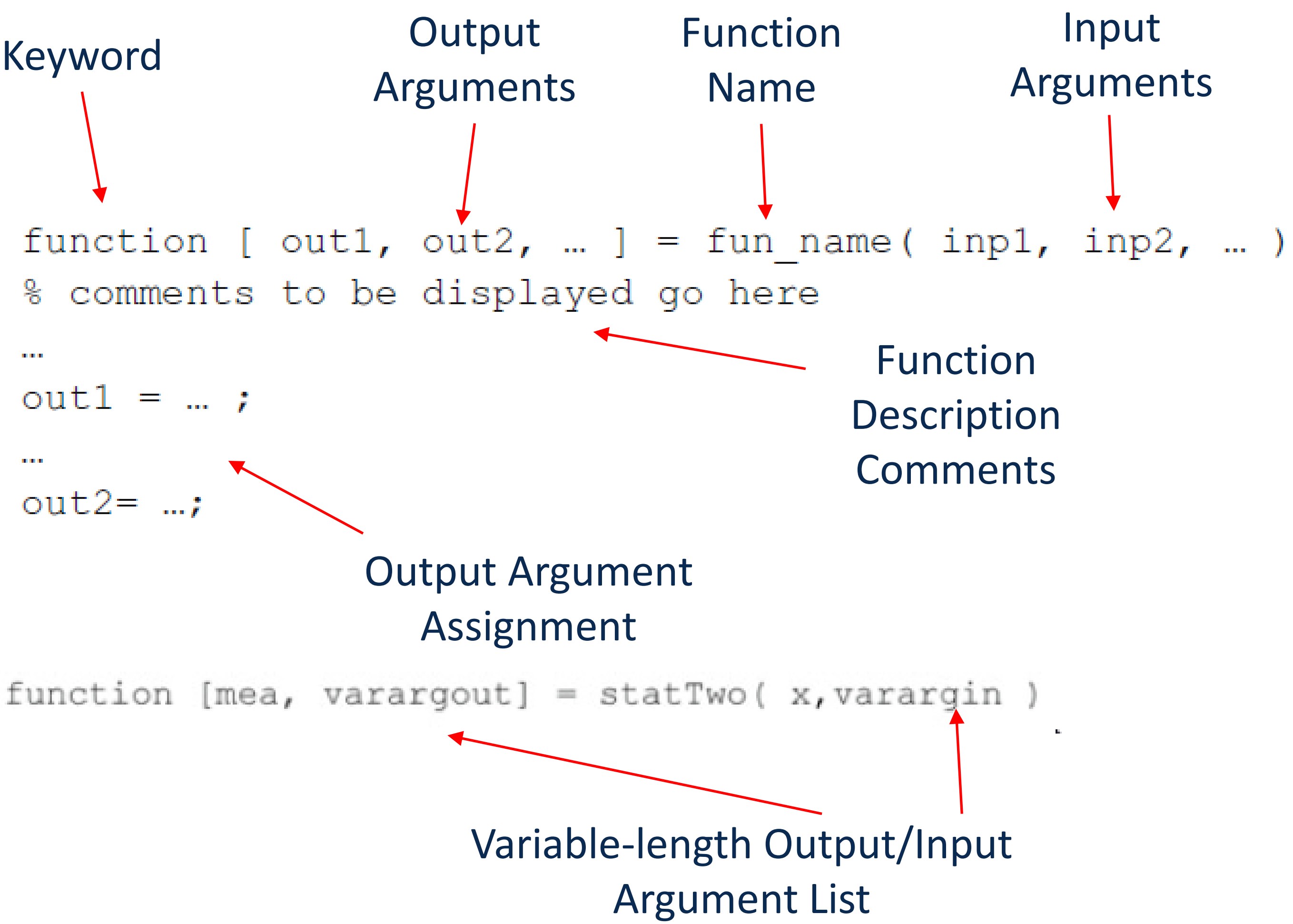
```
limit = 0.8;
s = 0;

while 1
    tmp = rand;
    if tmp > limit
        break
    end
    s = s + tmp;
end
```



```
try
        statements
catch
        statements
end
```

Functions



Keyword

Output Arguments

Function Name

Input Arguments

```
function [ out1, out2, … ] = fun_name( inp1, inp2, … )
% comments to be displayed go here
…
out1 = … ;
…
out2= …;
```

Function Description Comments

Output Argument Assignment

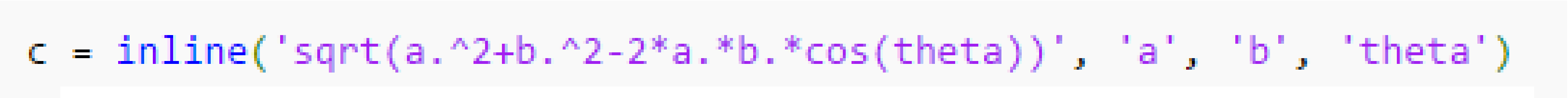
```
function [mea, varargout] = statTwo( x,varargin )
```

Variable-length Output/Input Argument List

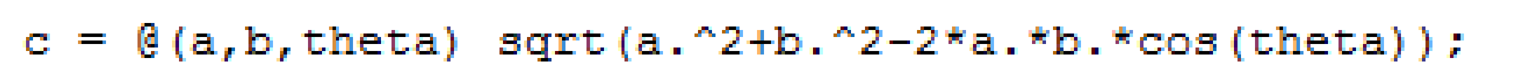
Functions

- In-line functions:

$$c(a,b,\theta) = \sqrt{a^2 + b^2 - 2abcos(\theta)}$$

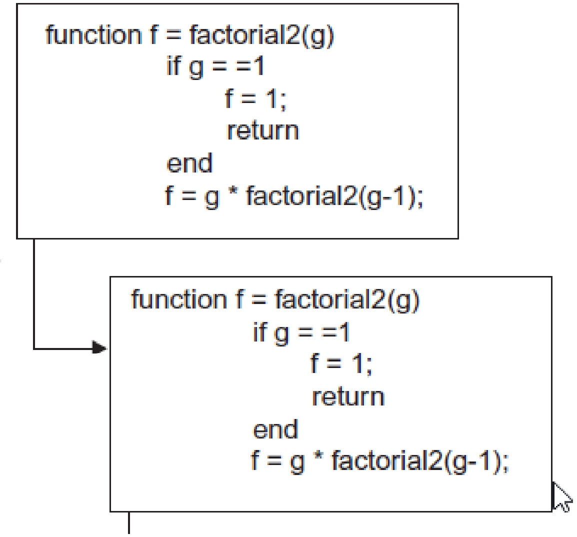


```
c = inline('sqrt(a.^2+b.^2-2*a.*b.*cos(theta))', 'a', 'b', 'theta')
```



```
c = @(a,b,theta) sqrt(a.^2+b.^2-2*a.*b.*cos(theta));
```

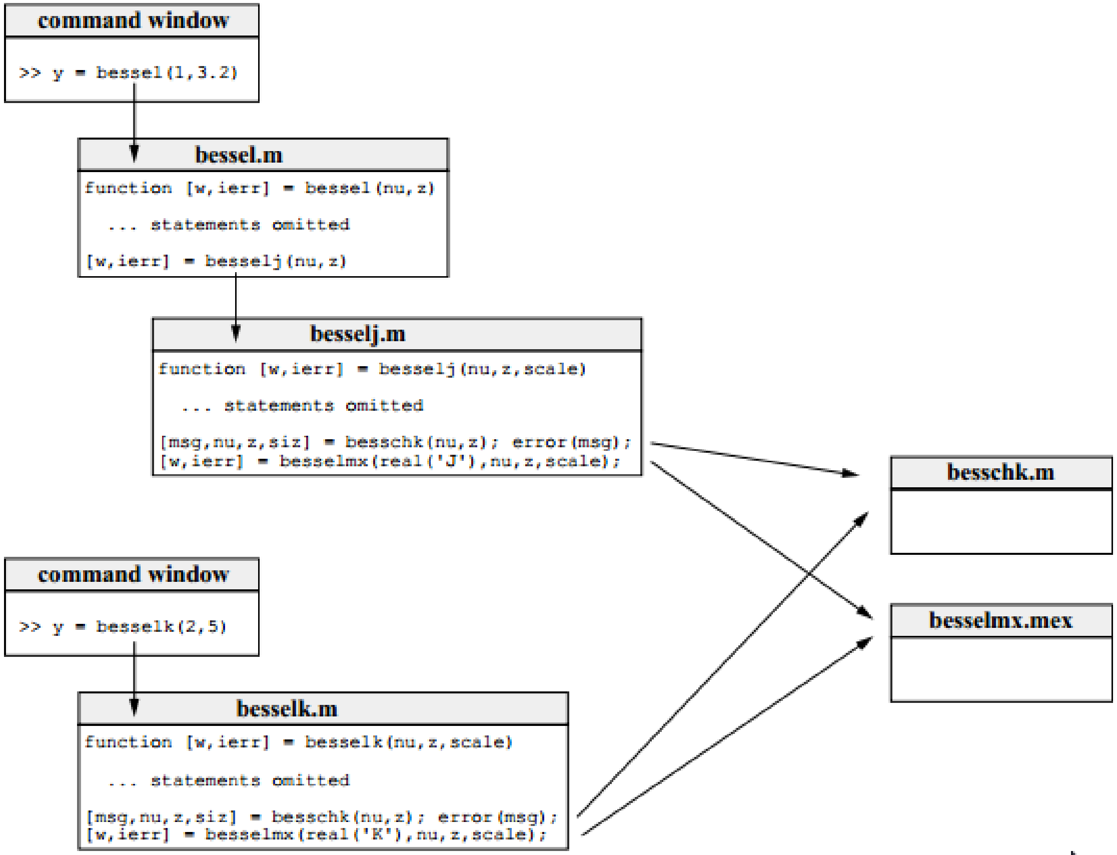
- Recursive functions:



```
function f = factorial2(g)
        if g = =1
            f = 1;
            return
        end
        f = g * factorial2(g-1);
```

```
function f = factorial2(g)
        if g = =1
            f = 1;
            return
        end
        f = g * factorial2(g-1);
```

Integration and Modular Programming



command window

```
>> y = bessel(1,3.2)
```

bessel.m

```
function [w,ierr] = bessel(nu,z)
   ... statements omitted
[w,ierr] = besselj(nu,z)
```

besselj.m

```
function [w,ierr] = besselj(nu,z,scale)
   ... statements omitted
[msg,nu,z,siz] = besschk(nu,z); error(msg);
[w,ierr] = besselmx(real('J'),nu,z,scale);
```

besschk.m

besselmx.mex

command window

```
>> y = besselk(2,5)
```

besselk.m

```
function [w,ierr] = besselk(nu,z,scale)
   ... statements omitted
[msg,nu,z,siz] = besschk(nu,z); error(msg);
[w,ierr] = besselmx(real('K'),nu,z,scale);
```

References

- **MATLAB for Psychologists (2012),** Borgo, M., Soranzo, A., Grassi, M., Springer-Verlag, 2012, ISBN. 978-1-4614-2196-2.

  - Chapter 3-4., pp. 47-82.

- **MATLAB for Neuroscientists, 2nd Ed: An Introduction to Scientific Computing (2014),** *Wallisch, P., Lusignan, M.E., Benayoun, M.D., Baker, T.I., Dickey, A.S. and Hatsopoulos, N.G.,* Academic Press, ISBN. 978-0123838360.

  - Chapter 2. pp. 7-114.

- **MATLAB help:**
  - http://www.mathworks.com/help/matlab/ref/subplot.html
  - http://www.mathworks.com/help/matlab/ref/surf.html
  - http://www.mathworks.com/help/matlab/ref/scatter.html
  - http://www.mathworks.com/help/matlab/ref/gcf.html