Mathematical Concepts (G6012)

Lecture 12

Thomas Nowotny
Chichester I, Room CI-105
Office hours: Tuesdays 15:00-16:45
T.Nowotny@sussex.ac.uk

Matrix multiplication

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} =$$

$$\begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} & a_{11}b_{13} + a_{12}b_{23} + a_{13}b_{33} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} & a_{21}b_{13} + a_{22}b_{23} + a_{23}b_{33} \\ a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31} & a_{31}b_{12} + a_{32}b_{22} + a_{33}b_{32} & a_{31}b_{13} + a_{32}b_{23} + a_{33}b_{33} \end{pmatrix}$$

The squares illustrate how things combine, analogous for the other fields.

Properties of Matrix Multiplication

Associativity: $A \cdot (B \cdot C) = (A \cdot B) \cdot C$

Not commutative (!!!): $A \cdot B \neq B \cdot A$

Under certain circumstances the Inverse of a matrix exists:

$$A \cdot A^{-1} = \mathbf{1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

So-called “right inverse”

Non-square matrices

Matrices do not have to be square:

$$A = \begin{pmatrix} -5 & 2 & 1 \\ 0 & 3 & -1 \end{pmatrix} \in \mathcal{M}(2,3)$$

$$B = \begin{pmatrix} 1 & -1 \\ 0 & 3 \\ 2 & -2 \end{pmatrix} \in \mathcal{M}(3,2)$$

$$A \cdot B =$$

Multiplication

$$\begin{pmatrix} -5 & 2 & 1 \\ 0 & 3 & -1 \end{pmatrix} \begin{pmatrix} 1 & -1 \\ 0 & 3 \\ 2 & -2 \end{pmatrix} = \begin{pmatrix} -3 & 9 \\ -2 & 11 \end{pmatrix}$$

What about $B * A$?

Matrix transpose

Transposition is the operation where lines and columns are swapped. Or a reflection along the diagonal, if you want:

$$A^T = (a_{ij})^T = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}^T$$

$$= \begin{pmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{pmatrix} = (a_{ji})$$

Properties of transposition

- In “component notation” it looks quite minimal:

$$A = (a_{ij}) \,, \quad B = (b_{ij}) = A^T$$

$$\Rightarrow b_{ij} = a_{ji}$$

- Row vectors become column vectors (and vice versa)
- m x n matrix becomes a n x m matrix

Scalar product

The scalar product of two vectors is defined as

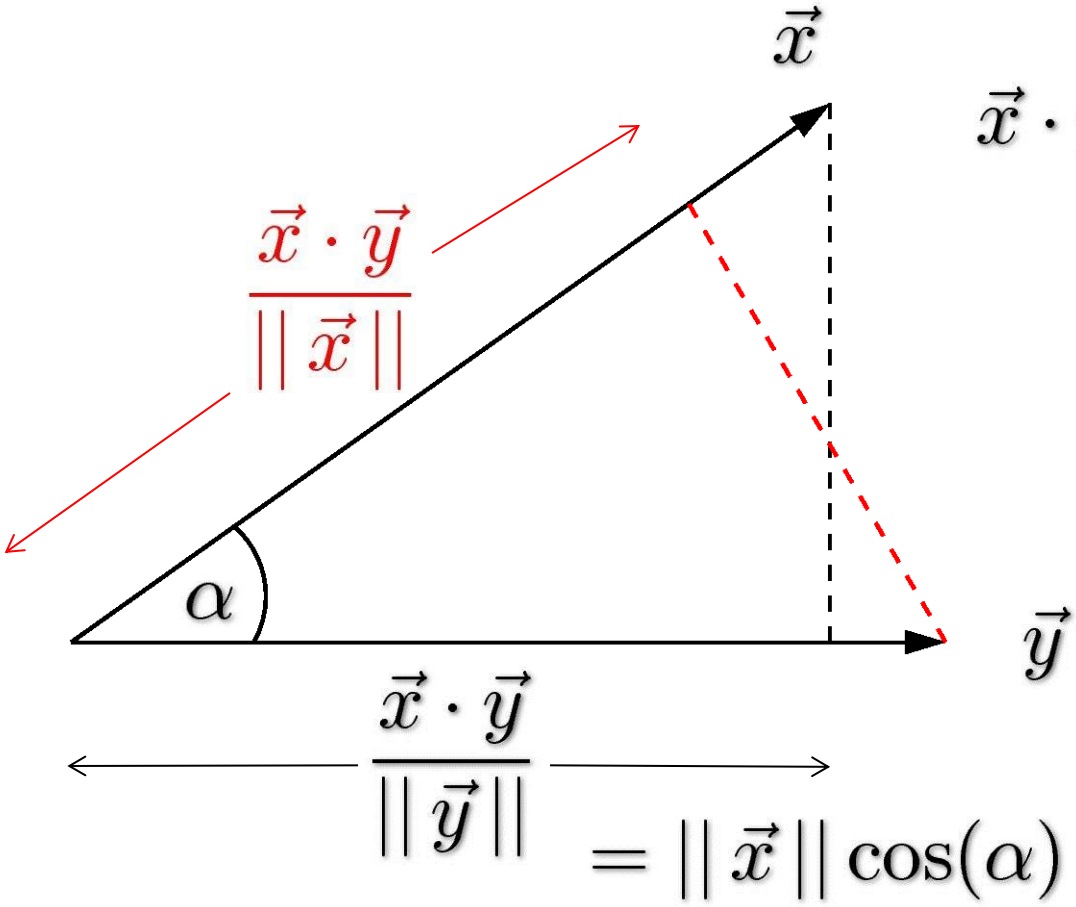
$$\vec{x} \cdot \vec{y} := \sum_{i=1}^{3} x_i y_i$$ also denoted as $\langle \vec{x}, \vec{y} \rangle$

Interpretation: *(in a moment) …*

It is a special case of Matrix multiplication:

$$\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = x_1 y_1 + x_2 y_2 + x_3 y_3$$

Interpretation of scalar product



$$\vec{x} \cdot \vec{y} = ||\vec{x}||\,||\vec{y}||\,\cos(\alpha)$$

Strictly speaking one should write …

$\langle \vec{x}, \vec{y} \rangle = \vec{x}^T \cdot \vec{y}$ for the scalar product.

Length and distances

- Euclidean norm (length)

$\vec{x} \in \mathbb{R}^n$

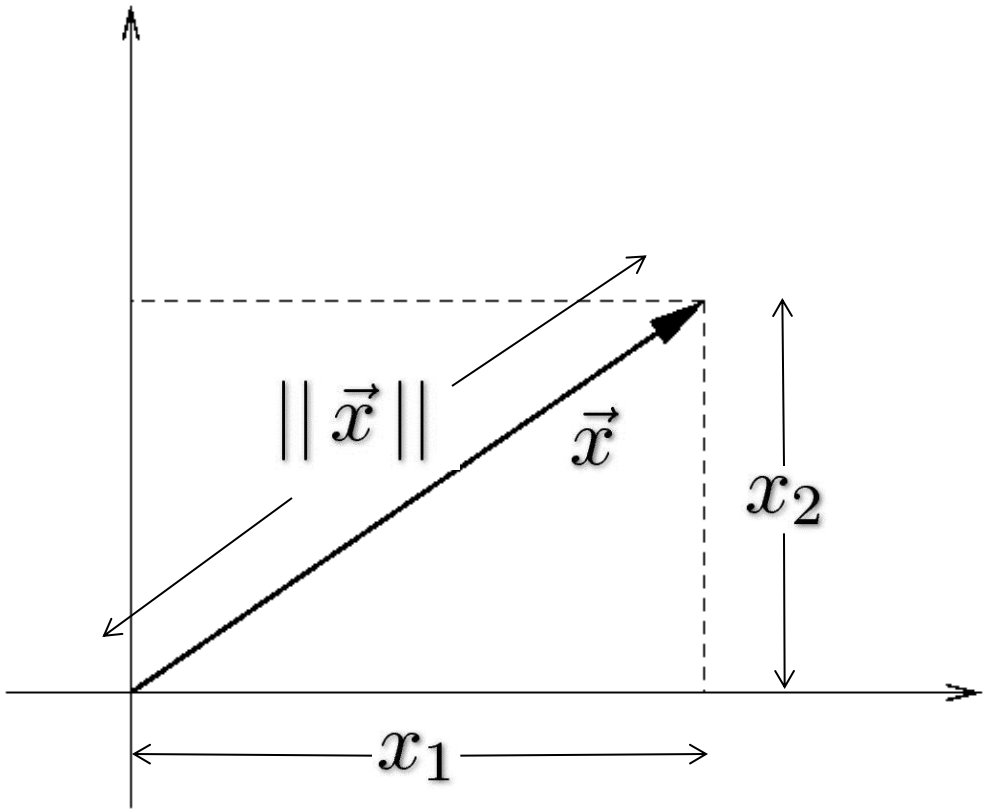
Norm of $\vec{x}$ is $||\vec{x}|| := \sqrt{\sum_{i=1}^{n} x_i^2} = \sqrt{\vec{x} \cdot \vec{x}}$

It is also called “2-norm”. Why this is our “natural” notion of length: BB

Remark: There are many other notions of length

BB

Why the definition of length matches our intuition for length



$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$||\vec{x}||^2 = x_1^2 + x_2^2$$

This is the **Pythagorean theorem** (check Wikipedia if never heard of it)

How does Length become Distance?

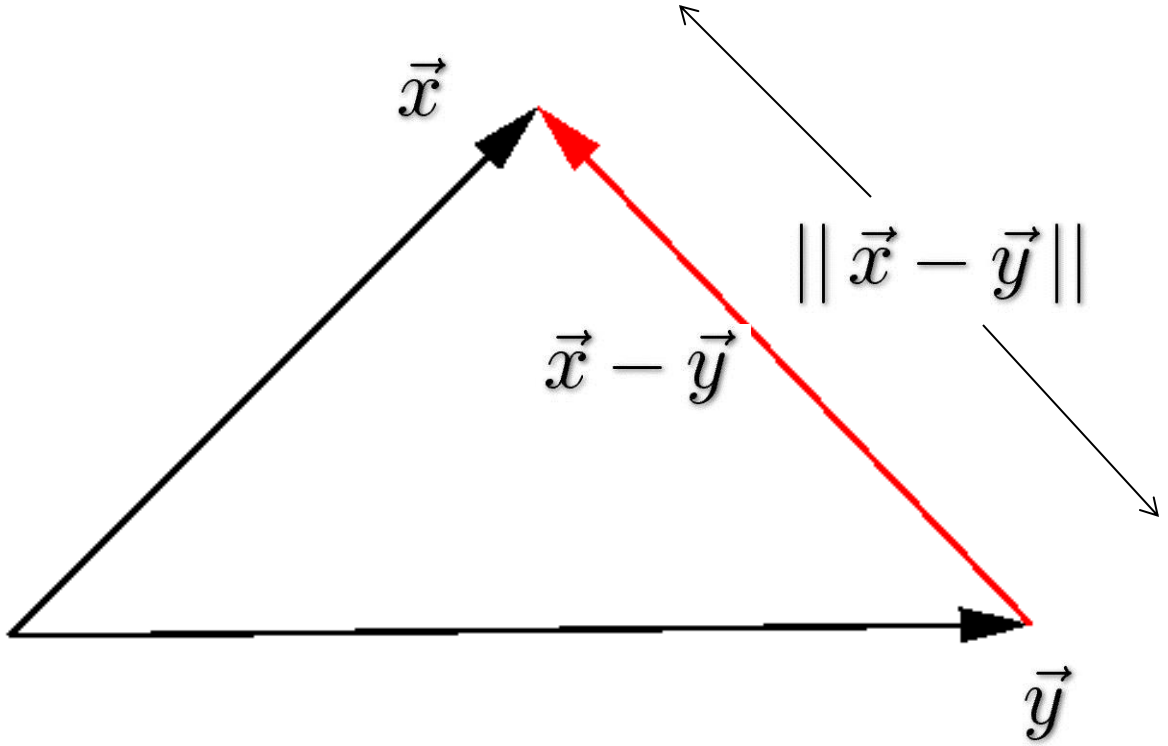
The distance between two vectors (points) is the length of the difference:

$$\vec{x},\ \vec{y} \in \mathbb{R}^n$$

$$d(\vec{x},\vec{y}) = ||\vec{x} - \vec{y}||$$

It is called **Euclidean distance**. Geometric interpretation: BB

BB Distance



The norm (length) $||\vec{x}-\vec{y}||$ of $\vec{x}-\vec{y}$ is the distance from $\vec{x}$ to $\vec{y}$

Some properties you would like to know

A norm or distance is always positive or 0.

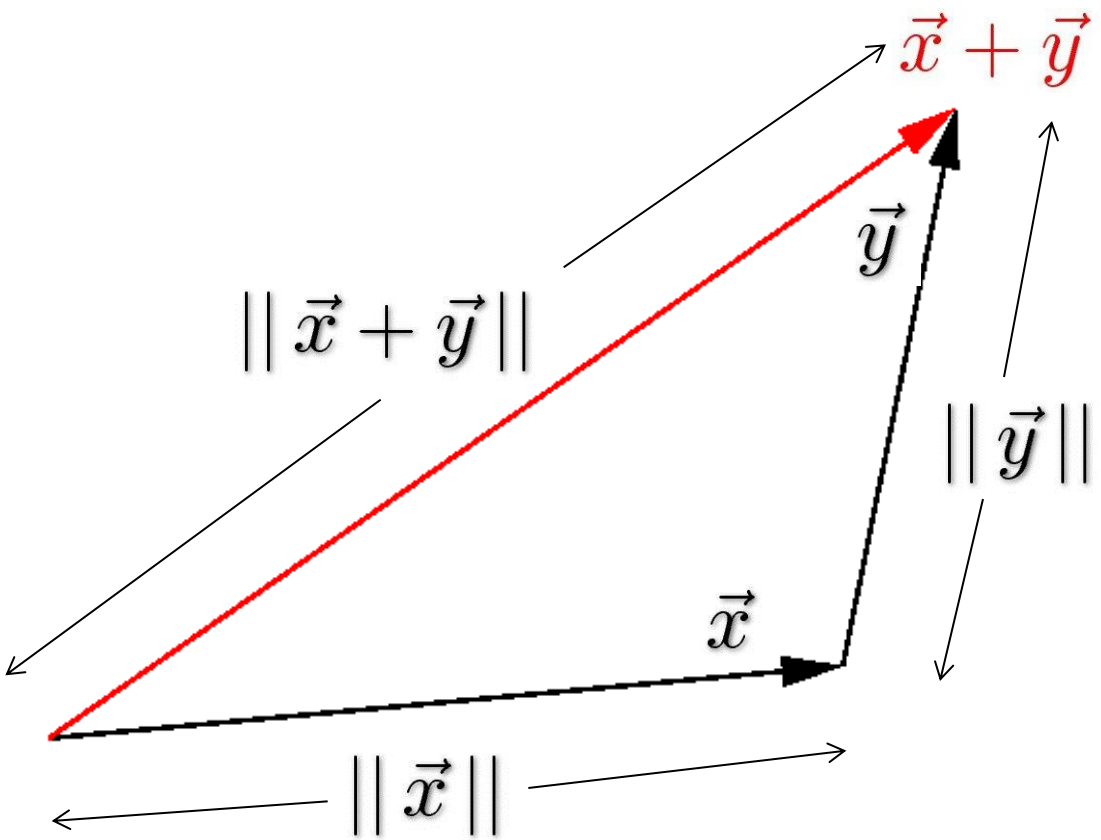
Scaling vectors: $||\, a\,\vec{x}\,|| = |a|\;||\vec{x}\,||$

Triangle inequality:

$$||\,\vec{x}+\vec{y}\,|| \;\leq\; ||\,\vec{x}\,|| + ||\,\vec{y}\,||$$

Geometric meaning … **BB**

BB Geometric meaning of Triangle Inequality



The triangle inequality means that going along the direct way ($||\vec{x}+\vec{y}||$) in a triangle is always shorter than (or equal to) going along the two other sides ($||\vec{x}||+||\vec{y}||$)

Other common norms

- The 1-norm: $||\vec{x}||_1 = \sum_{i=1}^{n} |x_i|$

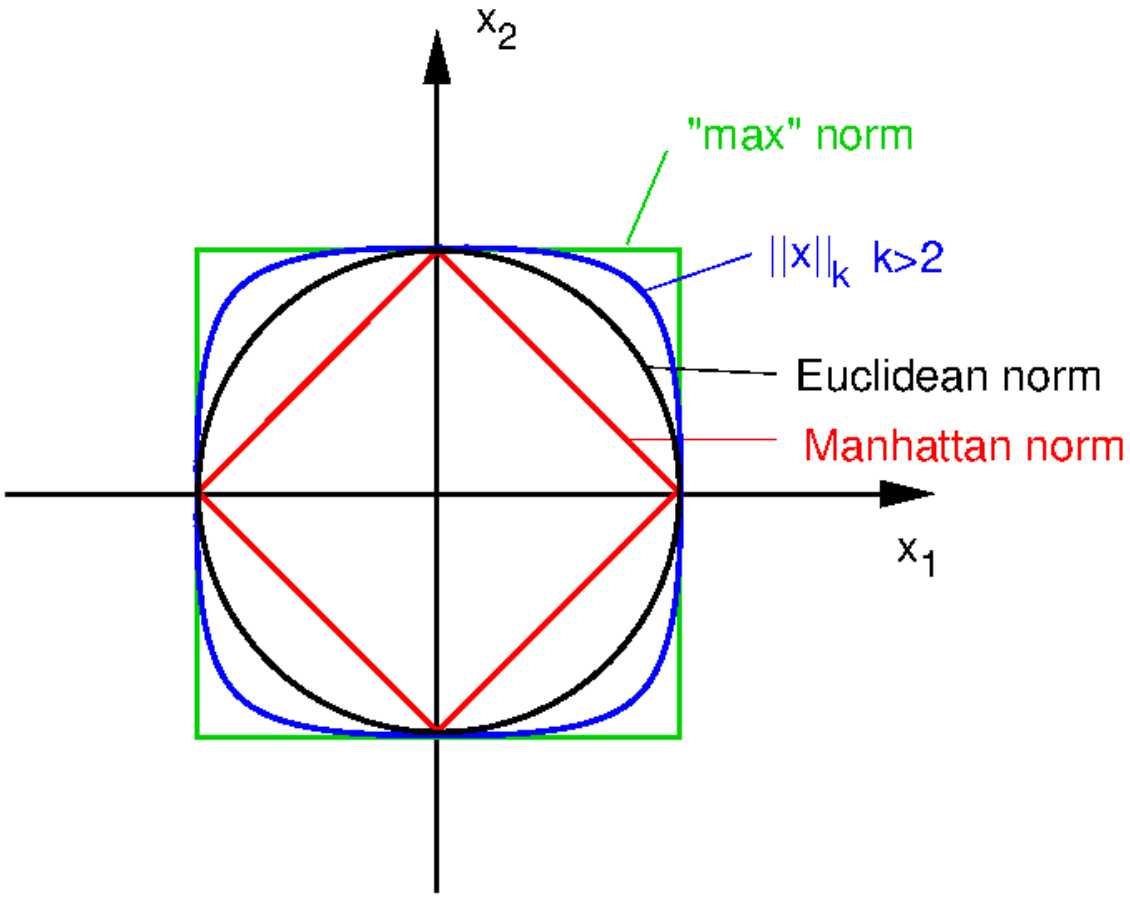
  (also called “Manhattan Distance” – why?)

- The n-norm: $||\vec{x}||_k = \left(\sum_{i=1}^{n} |x_i|^k \right)^{\frac{1}{k}}$

- The ∞ -norm: $||\vec{x}||_\infty = \max_{1 \le k \le n} |x_k|$

BB “isolines”

Lines of points which are the same distance from the origin have different shapes for different norms:



DEMO Vectors in Matlab

```
>> x = [ 1 -1 0 ]
x=  1 -1 0

>> y = [ 1; -1; 0 ]
y =
     1
    -1
     0

>>
```

DEMO Matrices in Matlab

```
>> A = [ 1 -1 0; 1 2 3; 3 -1 2 ]
A=
     1   -1   0
     1    2   3
     3   -1   2

>> A*y
ans =
      2
     -1
      4
>>
```

DEMO Accessing elements

```
>> A = [ 1 -1 0; 1 2 3; 3 -1 2 ];
>> A(1,1)
ans =
     1
>> A(1)
ans =
     1
>> A(1,:)
ans =
     1 -1 0
```

DEMO Accessing elements

```
>> A = [ 1 -1 0; 1 2 3; 3 -1 2 ];
>> A(:,1)
ans =
     1
     1
     3
>> A(2:3,1)
ans =
     1
     3
```

DEMO Transposition

```
>> A = [ 1 -1 0; 1 2 3; 3 -1 2 ]
A  =
     1   -1    0
     1    2    3
     3   -1    2
>> A'
ans =
     1    1    3
    -1    2   -1
     0    3    2
>>
```

DEMO Scalar product

```
>> x= [ 0; 1; 2 ]
x=
    0
    1
    2
>> y= [ 2; 0; -1 ]
Y=
    2
    0
   -1
>> x'*y
ans =
    -2
```

DEMO Errors

- If you try to use the value of an element outside of a matrix, it is an error:

```
>> A = [ 1 -1 0; 1 2 3; 3 -1 2 ]
A=
     1   -1   0
     1    2   3
     3   -1   2

>> t=A(4,5)

   Index exceeds matrix dimensions
```

- On the other hand, if you store a value in an element outside of the matrix, the size increases to accommodate the newcomer. Other created spaces are filled with 0.

DEMO Colon operator

- The colon operator, : , is one of MATLAB's most important operators. It occurs in several different forms. The expression *1:10* is a row vector containing the integers from 1 to 10

```
>> 1:10

ans =
   1  2  3  4  5  6  7  8  9  10
```

- To obtain non unit spacing, specify an increment. For example:

```
>> 100:-7:50

ans =
   100  93  86  79  72  65  58  51
```

DEMO Built-in functions

- MATLAB provides five functions that generate basic matrices:
  - `zeros` – all zeros
  - `ones` – all ones
  - `rand` – uniformly distributed random elements
  - `randn` – normally distributed random elements
  - `eye` – identity matrix
- Some examples:

```
>> F=5*ones(3,3)

  F =
    5 5 5
    5 5 5
    5 5 5
```

```
>> R=randn(4,4)

  R =
    1.0668   0.2944 -0.6918 -1.4410
    0.0593 -1.3362   0.8580   0.5711
   -0.0956   0.7143   1.2540 -0.3999
   -0.8323   1.6236 -1.5937   0.6900
```

DEMO MATLAB files and programs

- For example, create a file called factbar.m that contains these MATLAB commands:

```
% investigate the factorial explosion

r=ones(1,6);
for n=2:6
  r(n)=n*(r(n-1);
end;
bar(r);
```

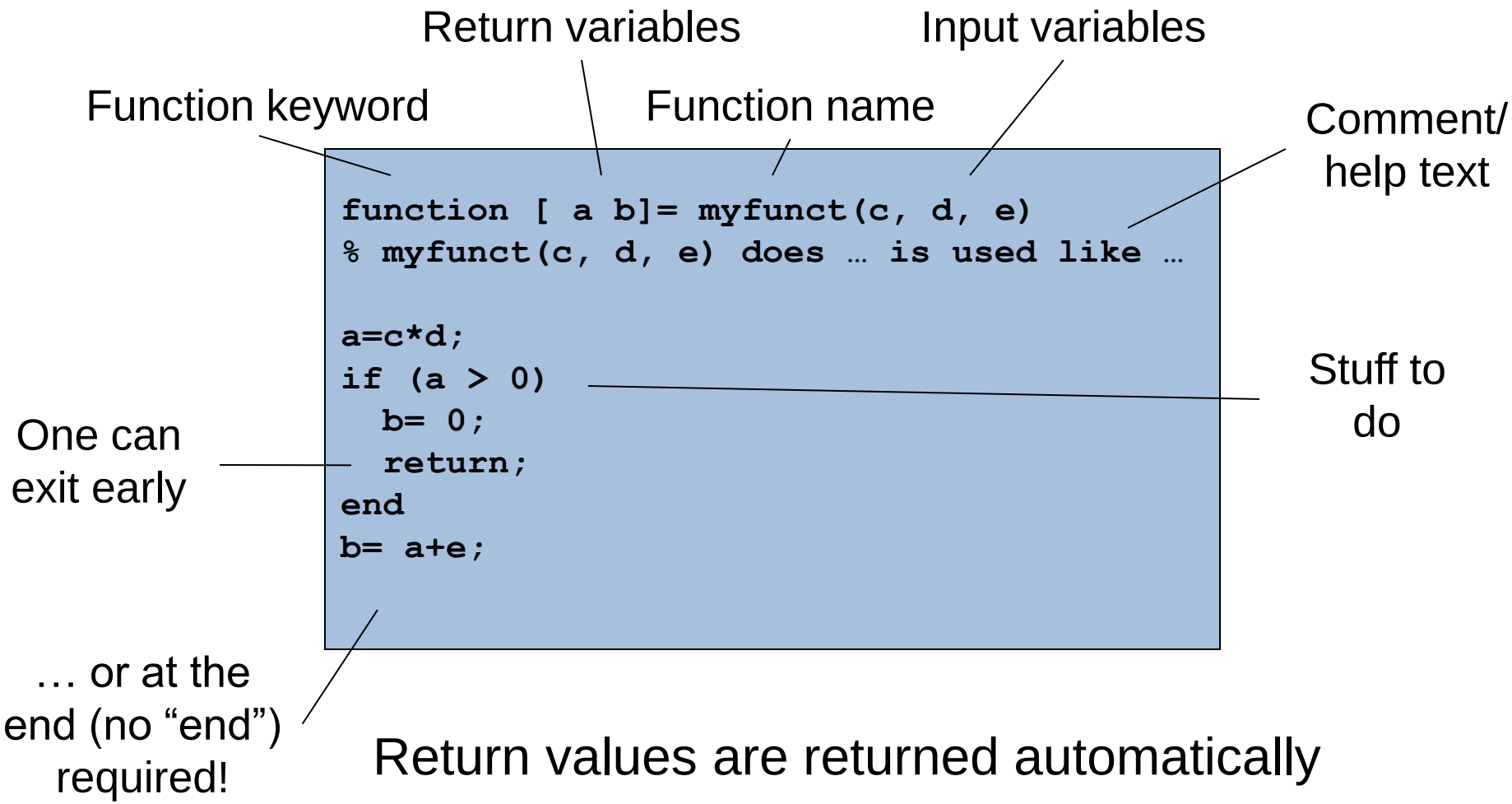
- This is a **script** (rather than a **function**) because it doesn't take any inputs or give any outputs.

DEMO M-file functions

- Functions are M-files that can accept input arguments and return output arguments; the name of the M-file and the function should be the same (**WARNING: If they are not the same, the file name overrides!**).
- Functions operate on variables within their own workspace

```
function f=myfact(n)
% MYFACT(N) computes N! using an iterative method
f=1;
if (n>1)
  for m=2:n
    f=m*f:
  end;
elseif (n<0)
  error('negative factorial attempted');
end;
```

M-file functions



Calling your own functions

```
function [ a b]= myfunct(c, d, e)
…
end;
```

myfunct.m

```
>> [ apple orange] = myfunct(candy, d, e);
```

If you do not provide multiple variables for return values, only one of the return values will be considered (and goes into “ans”)

A bit more detail …

- Variables in Matlab are passed by value, i.e. the content of the variables outside the function remains unchanged

```
function [ a b]= myfunct(c, d, e)
  e= 5;
  …
end;
```

```
>> e= 2;
>> [ apple orange] = myfunct(candy, d, e);
>> e
e=
   2
```

Example: visualising matrix action

"testMatrixSphere.m" draws a sphere and applies a matrix to it repeatedly

```
>> a= 0.2
>> A= [ cos(a)  sin(a) 0;
        -sin(a) cos(a) 0;
        0       0      1]
>> B= [ 0       0      1;
        0  cos(a) sin(a);
        0 -sin(a) cos(a)]
>> testMatrixSphere(A,10);
>> testMatrixSphere(B,10);
>> testMatrixSphere(A*B,10);
```

File management

- MATLAB uses a search path, or a list of directories, to determine how to execute functions. When we call a standard function, MATLAB executes the first M-file on the path that has the specified name.
- We can override this behaviour using special private directories and sub-functions. The command `path` shows the search path on any platform.
- MATLAB provides several generic operating system commands for manipulating and managing files:

File management

Command	Description
`what`	Return a listing of all M-files in the current directory of folder
`dir`	List all files in the current directory or folder
`ls`	Same as dir
`type test`	Display the M-file **`test.m`** in the command window
`delete test`	Delete the M-file **`test.m`**
`cd path`	Change to directory of folder given by **`path`**
`chdir path`	Same as **`cd path`**
`cd`	Show present working directory or folder (unlike UNIX)
`chdir`	Same as **`cd`**
`pwd`	Same as **`cd`**
`which test`	Display the directory path to **`test.m`**

“Toolboxes”

- Functions and scripts can call each other
- A collection of functions/scripts in a directory can form a complex, large program (much like a java `.jar` library)
- Existing toolboxes are such libraries

Alternatives to Matlab

- Python (numpy, scipy and matplotlib)
- Octave
- Mathematica