University of Applied Sciences

SAXION

# Mathematical Modelling Functions in Matlab & Numerical integration

*Ronald Tangelder, Jan de Wilde*

## Outline today

- Functions in MATLAB
- Numerical Integration
- Execution time
  - Matrix style versus standard programming style
  - Measuring execution time with tic and toc
  - Measuring execution time with by using the profiler

## Matlab functions

- If you want to make a function in Matlab, you should make a separate Matlab file with the <u>same name</u> as the function (unless we use an inline function (see later))

- This separate Matlab file should be (placed) in the working directory to be able to be used (or in a directory, which is included in the search path).

## Matlab functions: search order

- 1st: function defined in same m-file or mlx-file (at the end of the file, be careful: can overrule a built-in function)
- 2nd: built-in function
- 3rd: function in the same directory
- 4th: function somewhere else in the path

## A simple function (1)

```
function result=doubleIt(a)
% This comment line gives help info
% This comment line gives help info too
  result = 2*a;
end
```

- Unlike C++ and Java, there is no declaration of variables.
- Variable names should match.
- The scope (of `result` and `a`) is limited to the body of the function, i.e. outside the function `result` and `a` are unknown.

## A simple function (2)

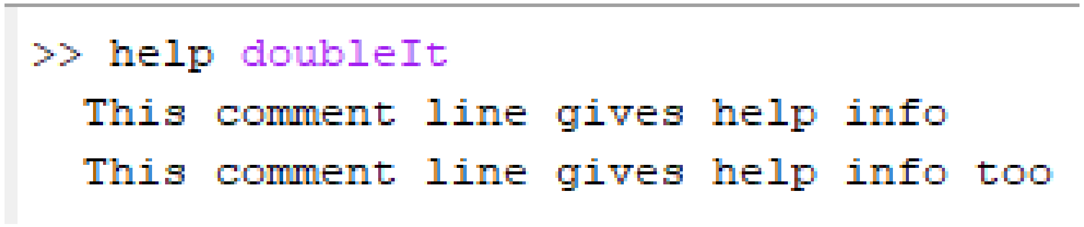
```
function result=doubleIt(a)
% This comment line gives help info
% This comment line gives help info too
  result = 2*a;
end
```

- Choose a function name which does not exist yet as built-in function.
- Help text (starting with a `%`) should be placed directly after the header; The help text will be returned, if you call `help` followed by the corresponding function name

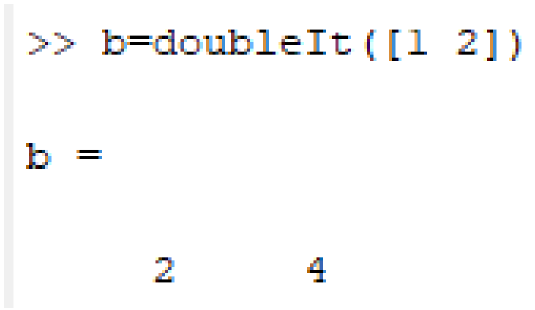
University of Applied Sciences SAXION

## A simple function (3)

```
function result=doubleIt(a)
% This comment line gives help info
% This comment line gives help info too
  result = 2*a;
end
```



```
>> help doubleIt
  This comment line gives help info
  This comment line gives help info too
```



```
>> b=doubleIt([1 2])

b =

     2     4
```

University of Applied Sciences SAXION

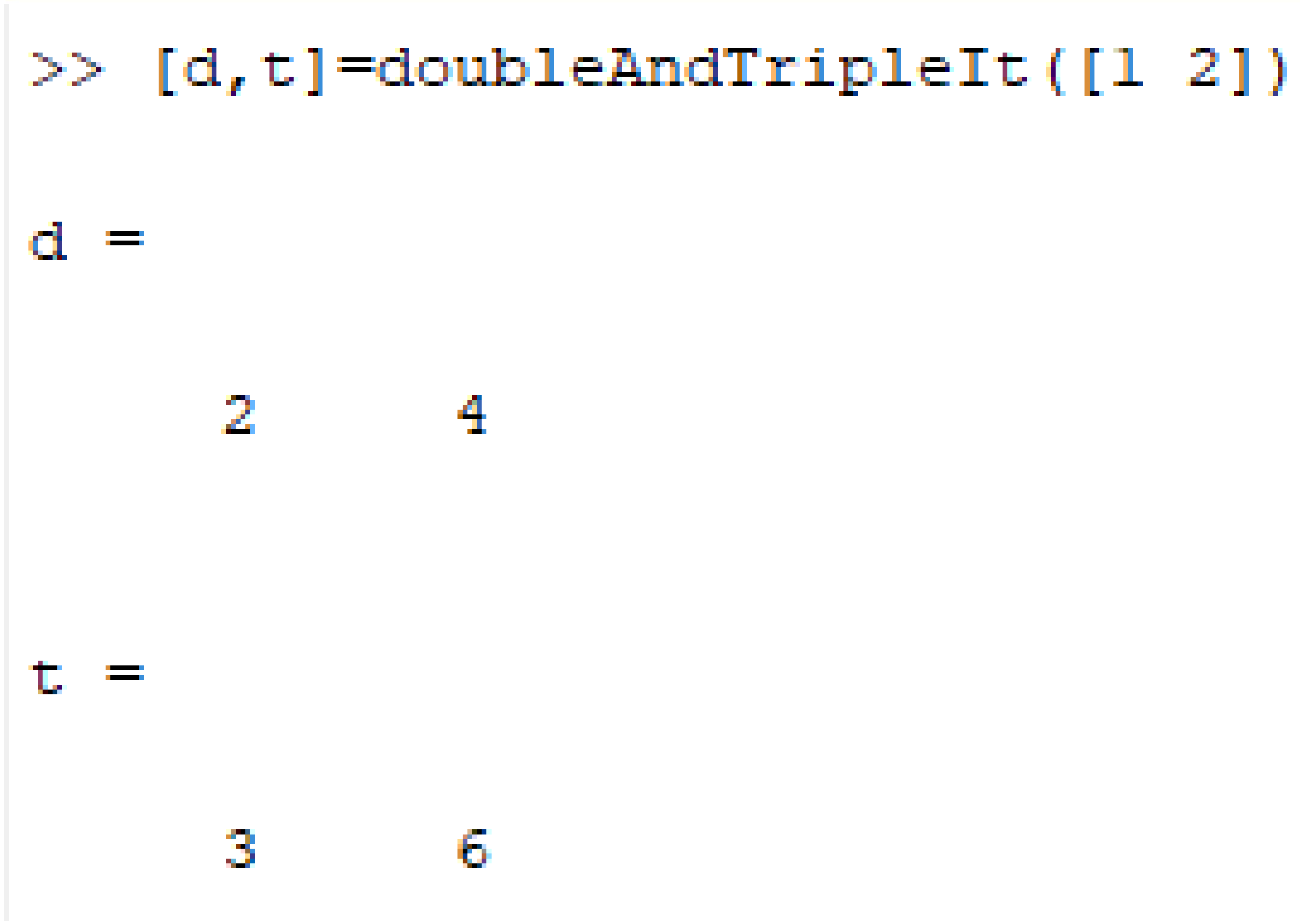
## Another simple function (1)

```
function [result2,result3]=doubleAndTripleIt(a)
% This comment line gives help info
% This comment line gives help info too

  % result2 is twice times a
  result2 = 2*a;
  % result3 is triple times a
  result3 = 3*a;
end
```

- Multiple return values are possible

University of Applied Sciences SAXION

## Another simple function (2)



```
>> [d,t]=doubleAndTripleIt([1 2])

d =

     2     4


t =

     3     6
```

## function end

In the current version of Matlab, a function will always finish with an `end` statement.

In previous versions, it was not needed (Matlab knew that the function ended at the end of the m-file).

To support backwards compatibility, it is not strictly necessary (unless you have multiple functions at the end of a file, i.s.o. in one separate file per function). However, it is highly recommended to do so.

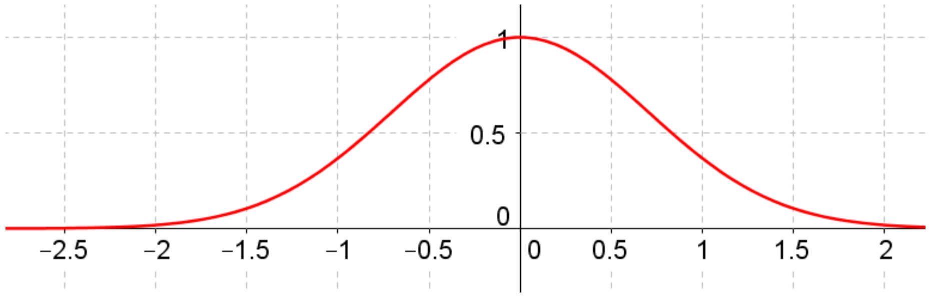
## Outline today

- Functions in MATLAB
- Numerical Integration
- Execution time
  - Matrix style versus standard programming style
  - Measuring execution time with tic and toc
  - Measuring execution time with by using the profiler

University of Applied Sciences SAXION

## If integration doesn't succeed

- There are (mathematical) functions of which the primitive (function) can not be found, or at least, not easily be found.
- Example: $f(x) = e^{-x^2}$
- How can we nevertheless calculate an area



  e. g. between $x = -1$ and $x = 1$?
- This can be done by numerical integration: **approximation of the definite integral:**
  - Divide the interval in little pieces $\Delta x$
  - Make rectangles with width $\Delta x$ and height $f(x)$
  - Add the areas of the rectangles.

University of Applied Sciences SAXION

## Approximation in Matlab

- `dx=0.1; % width of each rectangle`
- `x=[-1:dx:1-dx]; % top left corners`
- `f=exp(-x.^2);`
- `rectangles=f.*dx;`
- `area=sum(rectangles)`

```
area = 1.4924
```

University of Applied Sciences SAXION

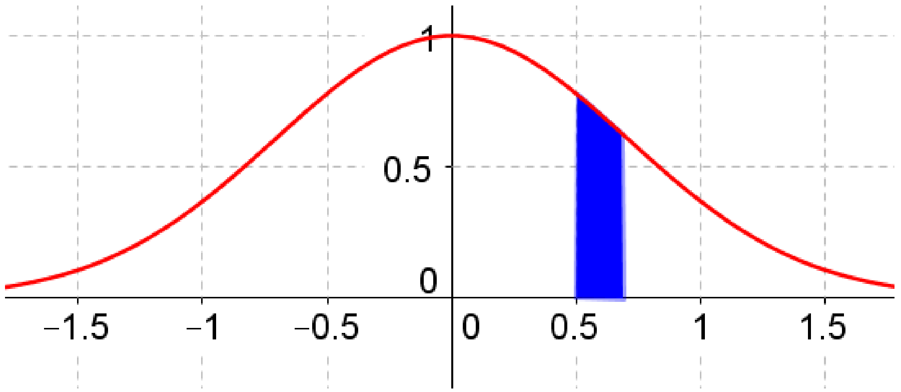
## Better approximation in Matlab

```
› dx=0.01; % width of each rectangle
› x=[-1:dx:1-dx]; % top left corners
› f=exp(-x.^2);
› rectangles=f.*dx;
› area=sum(rectangles)

area = 1.4936
(which is more precise than 1.4924)
```

University of Applied Sciences SAXION

## Trapezium

- A trapezium (trapezoid in American English) is a quadrangle of which two sides are parallel.
- Instead of placing rectangles under the graph, we can better use ‘slant cut off rectangles’:
- So that will be a trapezium (trapezoid)
- The area of a  trapezium (trapezoid) is composed of the area of a rectangle and a triangle
- Matlab provides us with a standard command:
- `trapz(x,y)`



University of Applied Sciences SAXION

## Example with trapz

```
› dx=0.01;
› x=-1:dx:1; % all top corners
› f=exp(-x.^2);
› area=trapz(x,f)

area = 1.4936
```

University of Applied Sciences SAXION

## Integration with `integral` (1)

If we use an *inline function* (also called anonymous function) in Matlab, integration can be done very easily and very precisely by using the built-in `integral` command.

Example: the inline function of $f(x) = e^{-x^2}$

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

University of Applied Sciences SAXION

## Integration with `integral` (2)

We would like to approximate the following integral numerically:

$$\int_{-1}^{1} e^{-x^2} dx$$

```
>> f = @(x) exp(-x.^2);
>> area = integral(f,-1,1)
area = 1.4936
```

## Outline today

- Functions in MATLAB
- Numerical Integration
- Execution time
  - Matrix style versus standard programming style
  - Measuring execution time with tic and toc
  - Measuring execution time with by using the profiler

## sumpos using matrices

```
function y = sumpos(x)
% This function calculates the sum of
% all positive elements of an input vector
  help1 = x>0;
  help2 = x.*help1;
  y = sum(help2);
end
```

## sumpos using for–loop

```
function y = sumpos2(x)
% This function calculates the sum of
% all positive elements of an input vector
  auxSum=0;
  for i = 1:length(x)
    if x(i)>0
        auxSum=auxSum+x(i);
    end
  end
  y = auxSum;
end
```

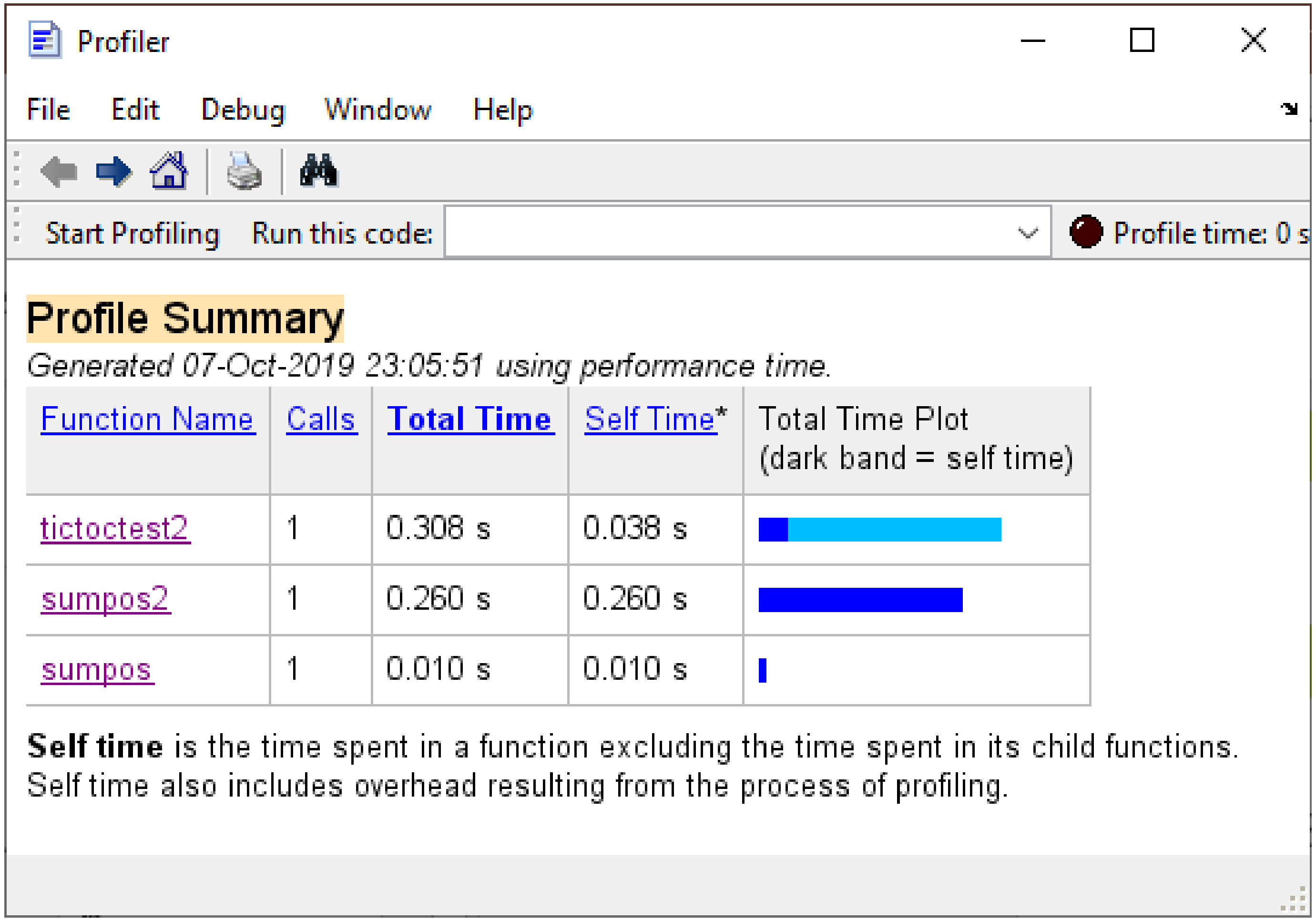
## A script with tic and toc

```
%% tic toc script
clc
v=randi([-255 255],[1 1e6]);
tic
sumpos(v)
toc
tic
sumpos2(v)
toc
```

## Output from toc

```
ans =

    63860606

Elapsed time is 0.011974 seconds.

ans =

    63860606

Elapsed time is 0.261144 seconds.
```

## Output from the profiler



**Profile Summary**

*Generated 07-Oct-2019 23:05:51 using performance time.*

| Function Name | Calls | Total Time | Self Time* | Total Time Plot (dark band = self time) |
|---|---|---|---|---|
| tictoctest2 | 1 | 0.308 s | 0.038 s | |
| sumpos2 | 1 | 0.260 s | 0.260 s | |
| sumpos | 1 | 0.010 s | 0.010 s | |

**Self time** is the time spent in a function excluding the time spent in its child functions. Self time also includes overhead resulting from the process of profiling.

## Questions