

Mathematical and computational methods for economists PART I

Prof. Mauro Maria Baldi

Department of Economics and Law,
University of Macerata,
mauromaria.baldi@unimc.it

AGENDA

- ❖ Overview of Mathematical Finance
- ❖ A brief introduction to Matlab
- ❖ Variables
- ❖ Solving some Mathematical Finance problems with Matlab
- ❖ The **disp**, **fprintf** and **sprintf** commands
- ❖ Strings
- ❖ Arrays

ACKNOWLEDGEMENT

This material was prepared also taking inspiration from some slides by Professor Elisabetta Michetti, to whom my thanks go.

RULE OF SIMPLE INTEREST

Def.

Consider an amount of money at time $t=0$, $w(0)$, called **PRINCIPAL**, and consider a rule such that the FUTURE VALUE is given by the principal plus **an interest that is attracted only by the principal**. Let $t=0,1,\dots,n$, then:

$$w(1) = w(0) + iw(0) = w(0)(1 + i)$$

$$w(2) = w(1) + iw(0) = w(0) + iw(0) + iw(0) = w(0)(1 + 2i) \dots$$

$$\dots w(n) = w(0)(1 + in), n \in \mathbb{N}$$

Where $i > 0$ is the interest rate related to one period while n is the number of periods.

If t is a real number denoting the length of the operation, then we obtain the following formula (*)

$$w(t) = w(0)(1 + it)$$

RULE OF COMPOUNDING INTEREST (OR EXPONENTIAL INTEREST)

Normally the interest already earned can be reinvested to attract even more interest.

Def.

Consider an amount of money at time $t=0$, $w(0)$, and assume that **the interest earned will be added to the principal periodically** (hence interests will be attracted not just by the original amount but also by all the interest earned so far) as follows:

$$w(1) = w(0) + iw(0) = w(0)(1 + i)$$

$$w(2) = w(1) + iw(1) = w(1)(1 + i) = w(0)(1 + i)^2 \dots$$

$$\dots w(n) = w(0)(1 + i)^n, n \in \mathbb{N}$$

This rule is called **compounding interest method** or **exponential rule** and the final formula is (**)

$$w(t) = w(0)(1 + i)^t$$

where t is a real number representing the length of the operation while i is the interest rate.

Again variables t and i must refer to the same time-unit.



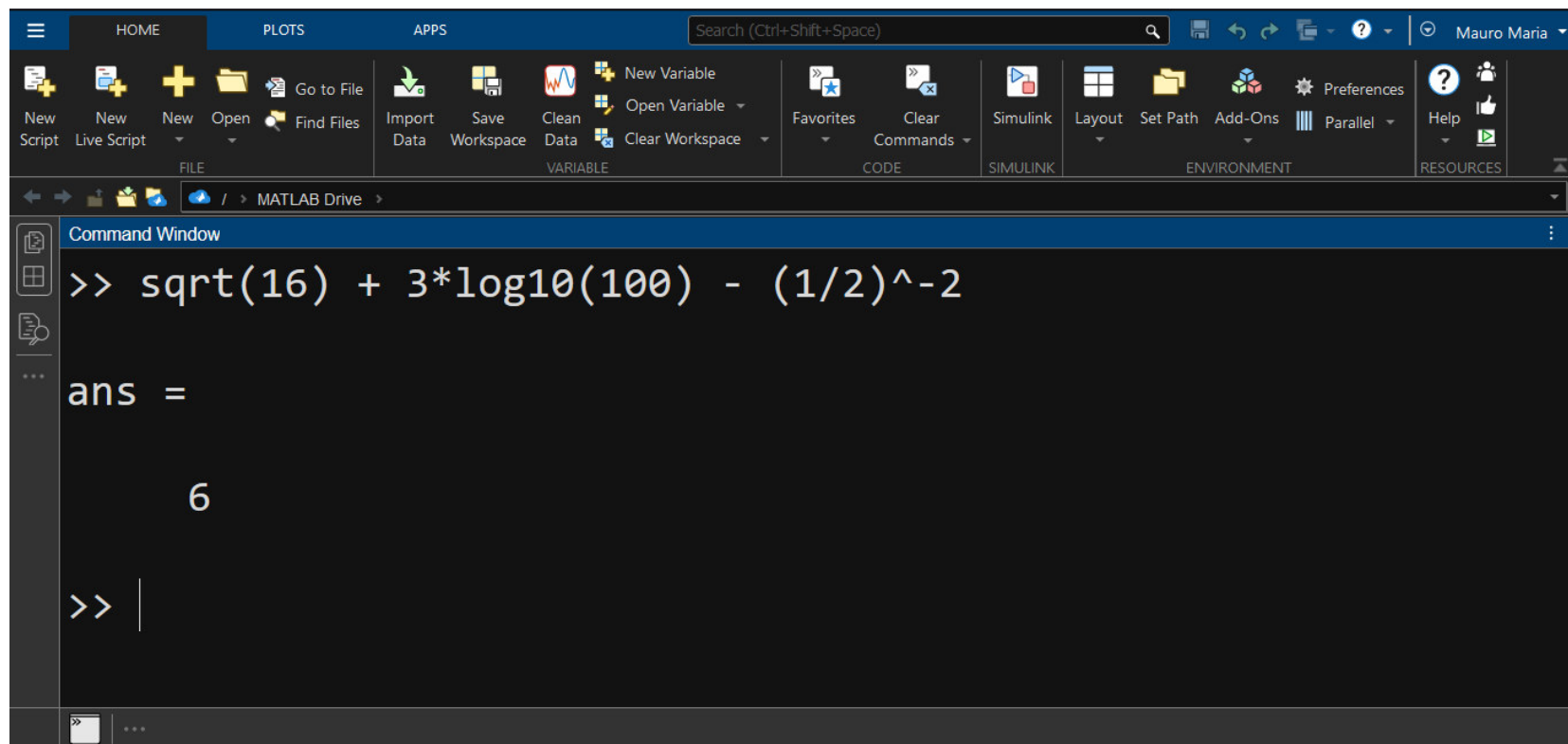
MatLab = MATrix LABoratory (environment for the scientific calculation and numerical simulations)

MatLab Desktop :

- **Command Window:** (to insert commands and instructions, `>>` is the **prompt**)
- **Command History:** chronological sequence of the executed instructions
- **Workspace:** operative memory containing the variables, **array**
- **Current Directory:** containing all the files and folders

Matlab can be used as a calculator in its command window. Here is an example to compute the value of the expression

$$\sqrt{16} + 3 \log_{10} 100 - \left(\frac{1}{2}\right)^{-2}$$



The screenshot shows the MATLAB interface with the Command Window open. The command window displays the following text:

```
>> sqrt(16) + 3*log10(100) - (1/2)^-2  
  
ans =  
  
      6  
  
>> |
```

SAVE A REAL NUMBER

To assign a name to a number you must use the symbol =

Define variable a with value 25

```
>> a=25
```

Such variable will be saved in the Workspace!

Notice: MatLab is *case sensitive*

Notice: ; after the command: the result of the instruction will not appear in the command window

EXERCISE

Save in MatLab the following real numbers:

$$A = \frac{12}{5}; B = 3^6; C = 2.1 \cdot 7$$

Notice: the point separates decimals

And calculate:

$$a = A + \frac{B}{C}; b = \frac{C - a}{B}; c = bB^A$$

SOLUTION

```
>> A = 12/5; B = 3^6; C = 2.1*7;
```

```
>> a = A + B/C
```

```
a =
```

```
51.9918
```

```
>> b = (C - a)/B
```

```
b =
```

```
-0.0512
```

```
>> c = b*B^A
```

```
c =
```

```
-3.7969e+05
```

GUIDING EXERCISE

Consider the rule of simple interest. If today we invest 2000 euro at an interest rate (annual) of 10%, what will we receive in 2 years? What in 3 and 4 years?

Warning: this exercise is going to be a guiding exercise throughout the course. Please keep it in mind because we are going to revise it a number of times

SOLUTION

```
>> w0 = 2000;  
>> ii = 10/100;  
>> w3 = w0*(1 + ii*3)
```

w3 =

2600

```
>> w4 = w0*(1 + ii*4)
```

w4 =

2800

```
>> w5 = w0*(1 + ii*5)
```

w5 =

3000

CAN WE DO BETTER?

The proposed solution is correct but it presents a number of drawbacks:

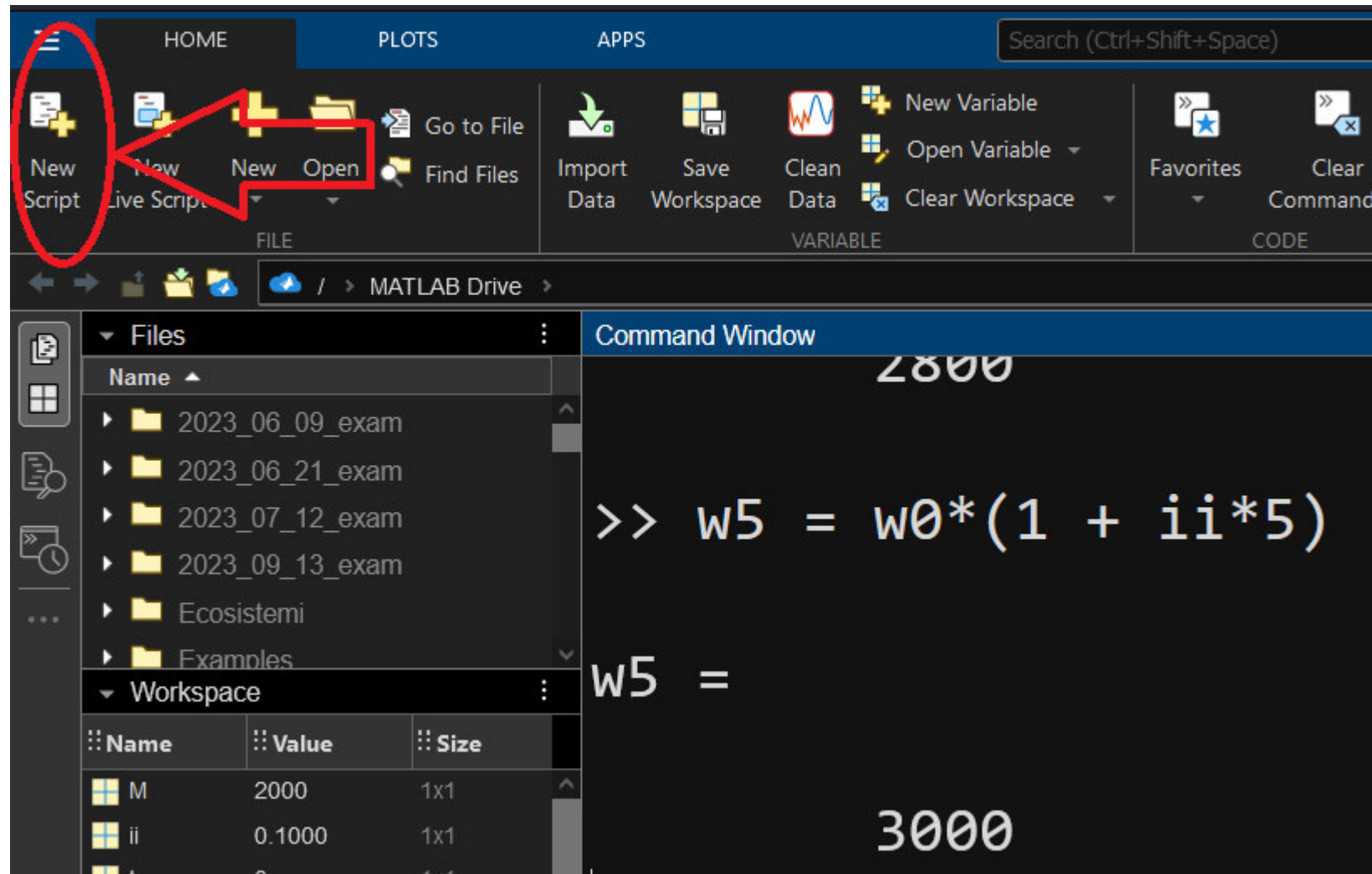
- ❖ You have to type everything from the very beginning if you make a mistake
- ❖ You may need to repeat the same computations with different values of the variables. In this case, you will have to write everything again from the beginning
- ❖ All your instructions are lost when you switch off your computer

CAN WE DO BETTER?

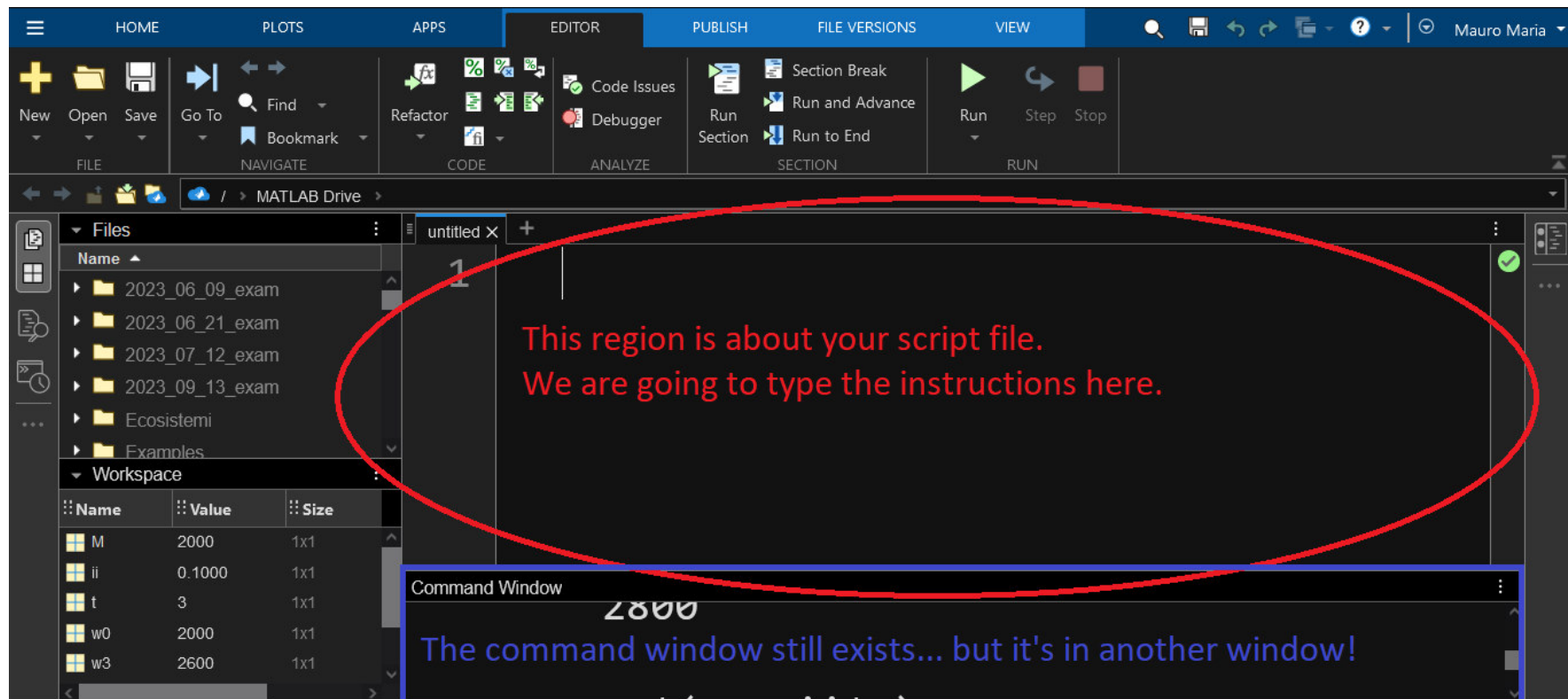
- ❖ Scripts are files containing instructions that can be executed as many times as we want
- ❖ The file is not lost when you switch off your computer
- ❖ We can modify the variable values as many times as we want (we should not forget to save the new version of our script!)
- ❖ Matlab scripts have extension .m

In the following slide, we are going to see how to store the instructions of our guiding example inside a script called **simple_interest_1.m**

STEP 1: open a script file in Matlab

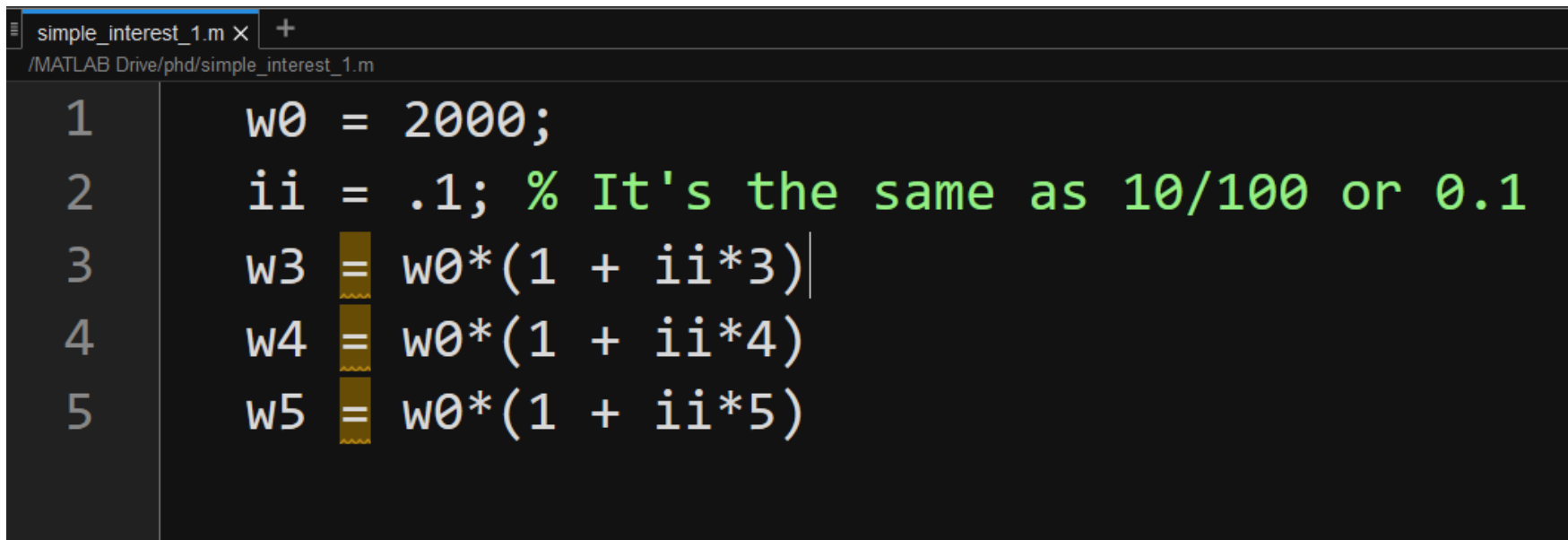


STEP 2: you will see something like this



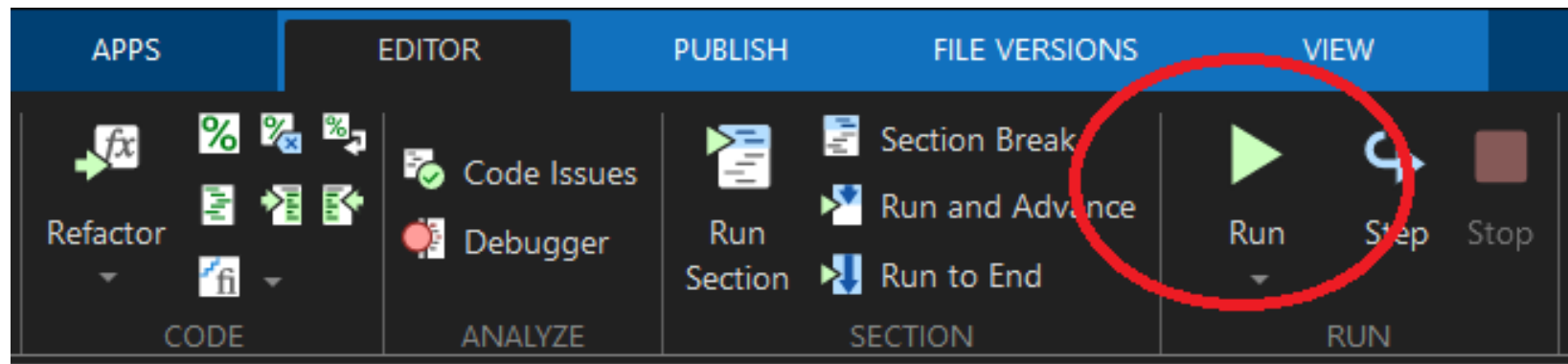
STEP 3: write your instruction inside the script file

Very highly recommended suggestion: save your script as soon as possible and also save it very frequently when you add new instructions!

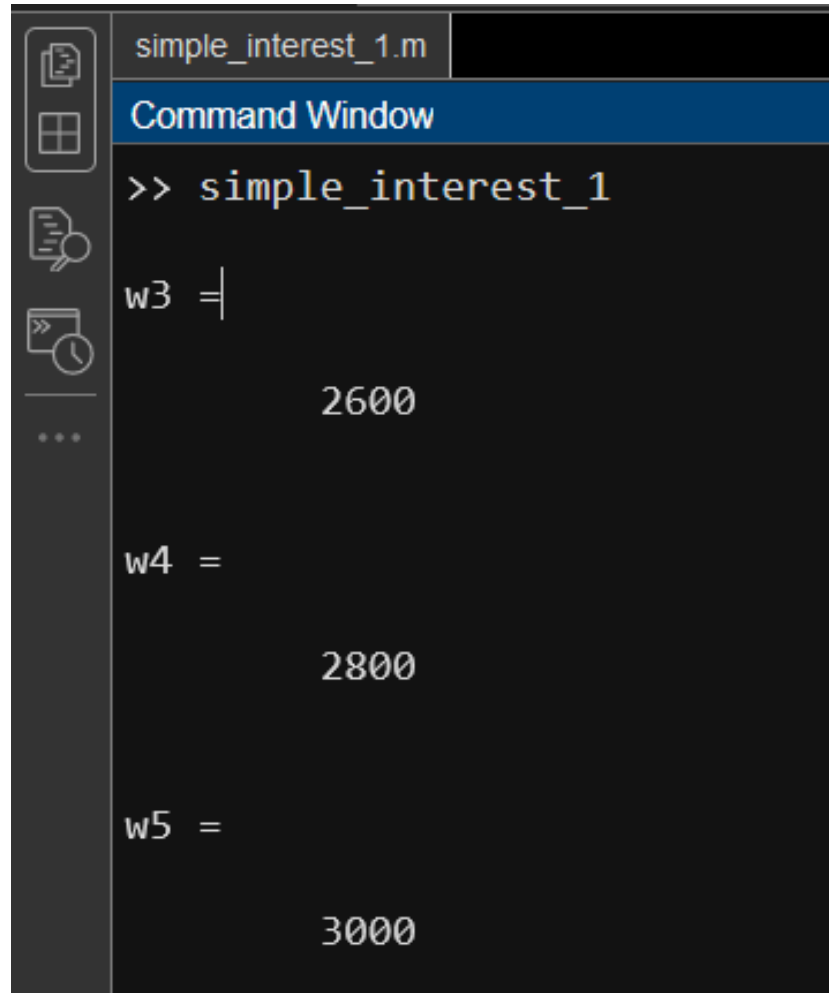


```
simple_interest_1.m × +  
/MATLAB Drive/phd/simple_interest_1.m  
1      w0 = 2000;  
2      ii = .1; % It's the same as 10/100 or 0.1  
3      w3 = w0*(1 + ii*3)  
4      w4 = w0*(1 + ii*4)  
5      w5 = w0*(1 + ii*5)
```

STEP 4: run your code from the editor menu



STEP 5: get the results in the command window



The image shows a MATLAB Command Window with a dark background. The title bar at the top reads "simple_interest_1.m". Below the title bar, the text "Command Window" is displayed in a blue header. The command prompt ">>" is followed by the function name "simple_interest_1". The results of the function are displayed in three lines: "w3 =" followed by "2600", "w4 =" followed by "2800", and "w5 =" followed by "3000".

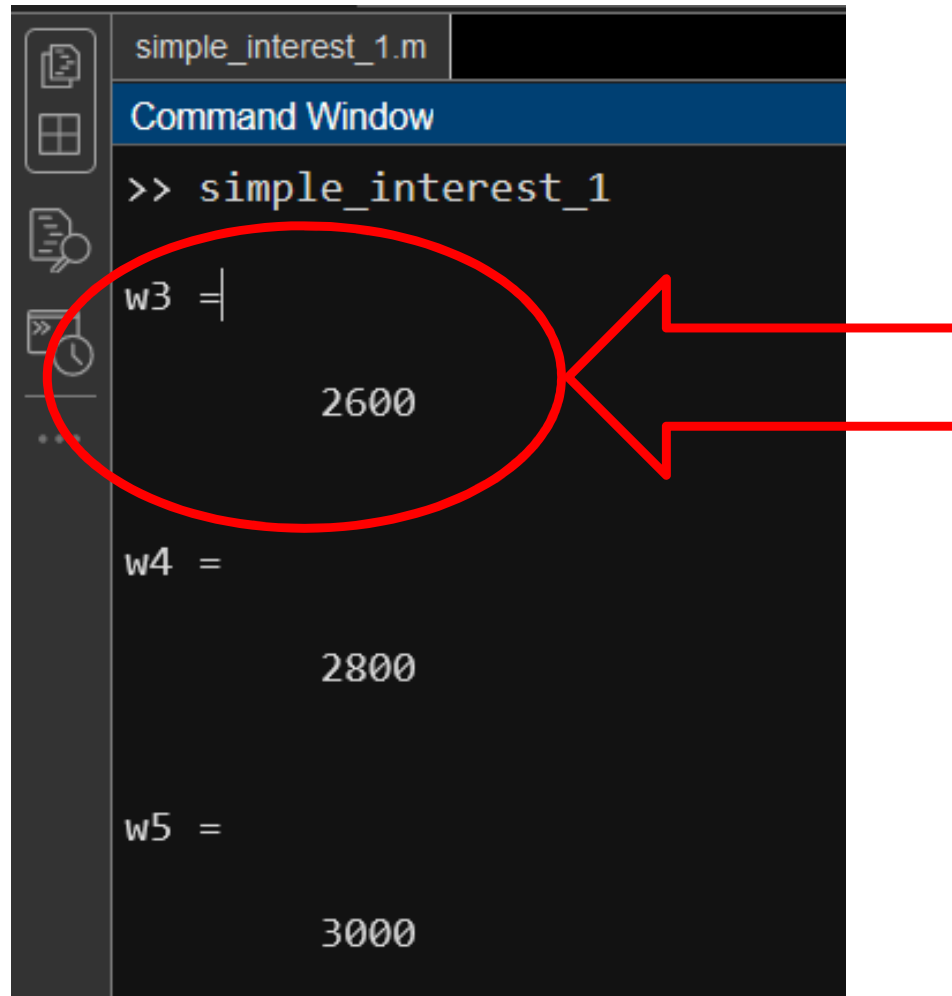
```
simple_interest_1.m
Command Window
>> simple_interest_1

w3 =
    2600

w4 =
    2800

w5 =
    3000
```

CAN WE DO BETTER?



A screenshot of the MATLAB Command Window. The title bar shows 'simple_interest_1.m'. The window title is 'Command Window'. The command prompt shows '>> simple_interest_1'. Below this, the variable 'w3' is assigned the value '2600'. This assignment is circled in red, and a red arrow points from the text '2600 ?' in the adjacent text block to this circle. Below 'w3', the variable 'w4' is assigned the value '2800'. Below 'w4', the variable 'w5' is assigned the value '3000'.

```
>> simple_interest_1  
w3 =  
    2600  
  
w4 =  
    2800  
  
w5 =  
    3000
```

How about if the program wrote

`w3 = 2600`

instead of

`w3 =`

`2600 ?`

The appearance would be much better!

We can achieve this by slightly modifying our script. We can call it, for instance, **simple_interest_2.m**

```
disp("This is the second version of the guiding example")
w0 = 2000;
ii = .1; % It's the same as 10/100 or 0.1
w3 = w0*(1 + ii*3);
fprintf("w3 = %d\n", w3)|
w4 = w0*(1 + ii*4);
fprintf("w4 = %d\n", w4)
w5 = w0*(1 + ii*5);
fprintf("w5 = %d\n", w5)
```

Command Window

```
>> simple_interest_2
This is the second version of the guiding example
w3 = 2600
w4 = 2800
w5 = 3000
>> |
```

AN INSIGHT INTO THE SCRIPT

- ❖ In the previous script, we have encountered some new commands (**disp** and **fprintf**) and some awkward terminology (**%d** and **\n**).
- ❖ What does all of this mean?
- ❖ To answer this question, we need to introduce the concept of **string**

WHAT IS A STRING

A string is a sequence of characters and/or numbers in quotation marks or apostrophes

```
>> s1 = "I am a string!«
```

```
s1 =
```

```
"I am a string!"
```

```
>> s2 = 'I am string number 2'
```

```
s2 =
```

```
'I am string number 2'
```

STRING CONCATENATION - I

A very interesting property of Matlab is that the + operator can also be extended to strings to concatenate them. Here is an example:

```
>> s3 = s1 + s2
```

```
s3 =
```

```
"I am a string!I am string number 2"
```

Another possibility is to use the **strcat** function. For example:

```
>> s3 = strcat(s1, s2)
```

```
s3 =
```

```
"I am a string!I am string number 2"
```


STRING CONCATENATION - II

Both approaches work also with more than two strings. Here is another example:

```
>> s4 =
```

```
"I am a string! ...and I am string number 2"
```

```
>> s4 = strcat(s1, "...and ", s2)
```

```
s4 =
```

```
"I am a string!...and I am string number 2"
```

THE disp FUNCTION - I

I can display the content of a string (or of a variable) in the command window using the **disp** function. To see this, let's create another short and simple script: **disp_example.m**

```
s1 = "Number of hours of this course:";  
n = 12;  
disp(s1)  
disp(n)
```

The result in the command window is:

```
Number of hours of this course:  
12
```

THE disp FUNCTION - II

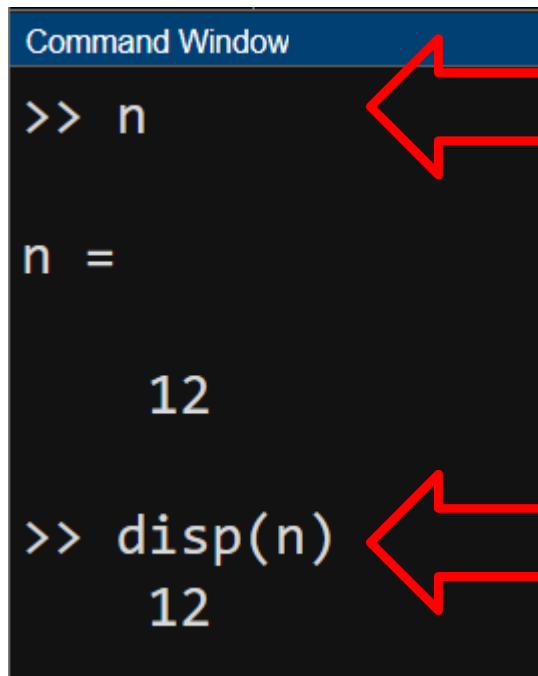
The **disp** function can also be called from the command line itself

Command Window

```
>> s1 = "Number of hours of this course";  
>> n = 12;  
>> disp(s1)  
Number of hours of this course  
>> disp(n)  
12
```

THE disp FUNCTION - III

There is a subtle difference between writing `disp(n)` or just `n` in the command window



```
Command Window
>> n
n =
    12

>> disp(n)
12
```

Here the result is:

n =
12

But here the result is just 12, without writing

n =

CAN WE DO BETTER?

The **disp_example.m** script provided us with the following output:

```
Number of hours of this course:  
12
```

This solution is still not satisfactory. In fact, we want our script to display:

```
Number of hours of this course: 12
```

Matlab offers a number of solutions to our purpose. They will be shown in the next slides.

SOLUTION I: THE `sprintf` FUNCTION

Consider a variant of the **disp_example.m** script. Let's name it **disp_example_2.m**

```
disp("This is an improvement of the disp_example.m script")
s1 = "Number of hours of this course:";
n = 12;
auxString = sprintf("%s %d", s1, n);
disp(auxString)
disp('...isn't this better?')
```

In the command line we read:

```
This is an improvement of the disp_example.m script
Number of hours of this course: 12
...isn't this better?
```

AN INSIGHT INTO THE `sprintf` FUNCTION

Let's have a closer look at the **`sprintf`** function

The result is a string, in this case it is stored in the `auxString` variable

```
auxString = sprintf("%s %d", s1, n);
```

The diagram illustrates the `sprintf` function call. A green arrow points from the text 'The result is a string, in this case it is stored in the auxString variable' to the `auxString` variable. Two green circles highlight the format string `"%s %d"` and the arguments `s1` and `n`. Two green arrows point from the bottom text boxes to these circles: one from the box explaining the format specifiers to the `"%s %d"` circle, and another from the box explaining the arguments to the `s1, n` circle.

Here we state that the output consists of a string (`%s`) and an integer (`%d`)

Here we state that the string is `s1` and the integer is `n`

CONTINUATION

After the previous instruction, the auxiliary variable auxString contains the string:

"Number of hours of this course: 12"

This string is then displayed in the command window through the instruction:

```
disp(auxString)
```

Please also note that the command

```
disp(...isn't this better? ")
```

has been written as:

```
disp('...isn't this better?')
```

Here you have to put two apostrophes not to interrupt the string because in this case the string is surrounded by apostrophes instead of quote marks.

SOLUTION II: THE fprintf FUNCTION

- ❖ We can directly display the result of the **sprintf** function in the command line avoiding to save it to an auxiliary string.
- ❖ To this purpose, we use the **fprintf** function instead of the **sprintf** function.
- ❖ Consider a variant of the **disp_example_2.m** script. Let's name it **disp_example_3.m**

```
disp("There is a drawback in this example ")  
s1 = "Number of hours of this course:";  
n = 12;  
fprintf("%s %d", s1, n);  
disp("...this isn't better at all!")
```

However, the result is disappointing:

```
There is a drawback in this example  
Number of hours of this course: 12...this isn't better at all!
```

This sentence should appear in the next line!

HOW DO WE FIX THIS ISSUE?

We just write the escape character `\n` to go to a new line.
The correct code is in the script **disp_example_4.m**

```
disp("This is a correct example of the fprintf function")  
s1 = "Number of hours of this course:";  
n = 12;  
fprintf("%s %d\n", s1, n);  
disp("...isn't it better?")
```

The result is:

```
This is a correct example of the fprintf function  
Number of hours of this course: 12  
...isn't it better?
```

FORMATTING TEXT - I

Now our script works well, but it contains a hidden trick. To see it, let's modify our guiding example, this time with $w_0 = 2$. The proposed code is in script **simple_interest_3.m**

```
w0 = 2;  
ii = .1; % It's the same as 10/100 or 0.1  
w3 = w0*(1 + ii*3);  
fprintf("w3 = %d\n", w3)  
w4 = w0*(1 + ii*4);  
fprintf("w4 = %d\n", w4)  
w5 = w0*(1 + ii*5);  
fprintf("w5 = %d\n", w5)
```

The result is:

```
w3 = 2.600000e+00  
w4 = 2.800000e+00  
w5 = 3
```

This is the exponential notation that can also be obtained with the %d formatting operator, i.e.,

```
fprintf("w3 = %e\n", w3)  
% ... etcetera  
fprintf("w4 = %d\n", w4)
```

FORMATTING TEXT - II

To display the result as 2.6, use the %f formatting operator, where f stands for fixed point notation. Let's save the new code in script **simple_interest_4.m**

```
w0 = 2;  
ii = .1; % It's the same as 10/100 or 0.1  
w3 = w0*(1 + ii*3);  
fprintf("w3 = %f\n", w3)  
w4 = w0*(1 + ii*4);  
fprintf("w4 = %f\n", w4)  
w5 = w0*(1 + ii*5);  
fprintf("w5 = %f\n", w5)
```

The result in the command window is:

```
w3 = 2.600000  
w4 = 2.800000  
w5 = 3.000000
```

CAN WE DO BETTER?

The previous result is an improvement of the exponential notation. But how about if we want to show the result with just two decimal digits?

The solution is in script **simple_interest_5.m**

```
w0 = 2;  
ii = .1; % It's the same as 10/100 or 0.1  
w3 = w0*(1 + ii*3);  
fprintf("w3 = %.2f\n", w3)  
w4 = w0*(1 + ii*4);  
fprintf("w4 = %.2f\n", w4)  
w5 = w0*(1 + ii*5);  
fprintf("w5 = %.2f\n", w5)
```

This script produces the following results:

```
w3 = 2.60  
w4 = 2.80  
w5 = 3.00
```

OTHER USEFUL COMMANDS

>>**clc** cleans command window

>>**clear** cleans workspace

>> ↑ recall the last executed command

>>**doc "argument"** consults MatLab help on the «argument»

>>**format long** activates 14 decimals format

>>**format short** activates 4 default decimal format

>>**doc elfun** to consult MatLab help on elementary functions.

TRY TO

Save in MatLab the following real numbers:

And calculate: $x = \ln(4); y = \frac{1}{8}; z = -6$

$$p = e^x - \ln 2; q = \sqrt{yz^3}; r = |-10 + 2z|$$

SOLUTION

```
>> x = log(4); y = 1/8; z = -6;
```

```
>> p = exp(x) - log(2)
```

```
p =
```

```
3.3069
```

```
>> q = sqrt(y)*z^3
```

```
q =
```

```
-76.3675
```

```
>> r = abs(-10 + 2*z)
```

```
r =
```

```
22
```


ANOTHER EXAMPLE

Consider 2200 euro disposable in 15 months and the rule of exponential interest with $i=11\%$. Determine its present value

First we choose to determine the monthly interest rate: $i_{12} = (1.11)^{1/12} - 1$

With MatLab

```
>> i12=(1.11)^(1/12)-1
```

```
i12 =
```

```
0.0087
```

Then compute the present value: $PV = 2200(1 + i_{12})^{-15}$

With MatLab

```
>> pv=2200*(1+i12)^-15
```

```
pv =
```

```
1.9309e+03
```

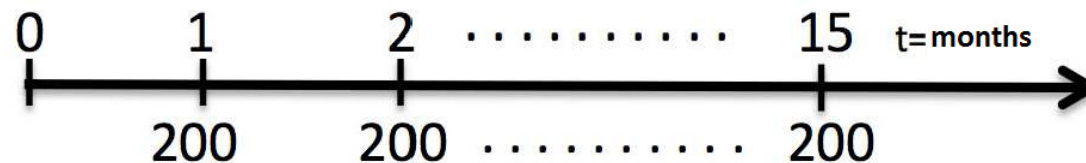
HENCE the present value is: 1930.9

ANNUITY

Def. Annuity: An annuity is a sequence of **finitely many payments** of a fixed amount (R) **due at equal time intervals** for n periods. This is a complex financial operation.

Def.: An annuity is called **ORDINARY-ANNUITY** or **annuity-immediate** if the payments are made at the end of the payment periods, while it is called **ANNUITY-DUE** if the payments are made at the beginning of the payment periods

EX: Annuity that starts today of amount 200 paid montly for 15 months: this is an annuity-immediate



MAIN GOAL: we want to know the value of the annuity at time 0, called **present value**, or the accumulated value of the annuity at time n , called the **future value**, by considering the compounding interest rule where **i is the rate of interest per period** (for instance for a quarterly annuity we need to use i_4 , for a semi-annual annuity the rate i_2 must be used...)

ORDINARY ANNUITY

Consider an ordinary annuity, then its **present value** is given by the sum of the present values of each payment.

Denote with $w(\underline{R}, 0)$ the present value of an ordinary annuity of amount R . If $R=1$ we have a unitary ordinary annuity and we define

$$v = (1 + i)^{-1}$$

then

$$\begin{aligned} W(\underline{1}, 0) = a_{\overline{n}|i} &= v + v^2 + v^3 + \dots + v^n \\ &= v \times \left[\frac{1 - v^n}{1 - v} \right] \\ &= \frac{1 - v^n}{i} \\ &= \frac{1 - (1 + i)^{-n}}{i}. \end{aligned}$$

Called «a angle n at i»

And consequently,

$$w(\underline{R}, 0) = R \cdot a_{\overline{n}|i}$$

is the present value of an ordinary annuity of amount R

From the present value of an annuity it is immediate to obtain the **future value** (or the value at any given time):

$$w(\underline{R}, n) = w(\underline{R}, 0)(1 + i)^n = Ra_{\overline{n}|i}(1 + i)^n = \\ R \frac{(1 + i)^n - 1}{i} = R s_{\overline{n}|i}$$

EX14: Calculate the present value of an ordinary annuity of amount \$100 paid annually for 5 years at the rate of interest of 9% . Also calculate its future value at time 5.

$$\text{Present value: } 100 a_{\overline{5}|0.09} = 100 \times \left[\frac{1 - (1.09)^{-5}}{0.09} \right] = \$388.97$$

$$\text{Future value: } (1.09)^5 \times (100 a_{\overline{5}|0.09}) = (1.09)^5 \times 388.97 = \$598.47.$$

$$\text{or alternatively: } 100 s_{\overline{5}|0.09} = 100 \times \left[\frac{(1.09)^5 - 1}{0.09} \right] = \$598.47.$$

ANOTHER EXAMPLE

Consider an annuity-due of 625 euro, payable every year , ending in 10 years, monthly interest rate 0.5%. Find the present value.

First determine the annual interest rate: $i = (1.005)^{12} - 1$

With MatLab

```
>> i=(1.005)^12-1
```

i =

0.0617

Then compute the present value:

$$PV = 625 \frac{1 - (1 + i)^{-10}}{i} (1 + i)$$

With MatLab

```
>> PV=625*(1-(1+i)^-10)/i*(1+i)
```

PV =

4.8452e+03

HENCE the present value is: 4845.2

PROPOSED EXERCISES

- 1.** Calculate the present value of an ordinary annuity of amount 100 euro payable quarterly for 10 years at the annual rate of interest of 9%. Also calculate its future value at the end of 10 years.
- 2.** Find the present value of an annuity-due of 200 euro per semester for 2 years, if the semi-annual interest rate is 5%.
- 3.** Consider an immediate-annuity of 300 euro, payable every 6 month for 30 months, interest rate 7%. Determine its present value and final value.
- 4.** Find the value after 5 years of an annuity-due of 400 euro payable every 2 months for 17 years given $i_2=0.5$.
- 5.** Consider a yearly annuity-immediate of 500 euro starting at 5 year with 11 payments. Find the present value and the final value given $i_{1/2}=18\%$.
- 6.** Consider an annuity-due of 160 euro payable every two-months, starting at 2 years and ending at 4 years. Find the present value and the final value (at the end of the annuity, i.e. after 4 years) given $i_{12}=2\%$.

Def. Perpetuity: A perpetuity is an annuity with no termination date,

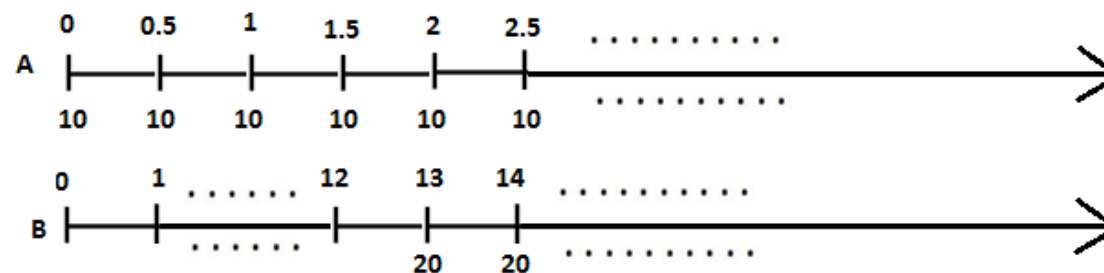
i.e., $n \rightarrow \infty$.

Similar definitions to those related to an annuity can be given.

Def.: A perpetuity is called **ORDINARY-PERPETUITY** or **perpetuity-immediate** if the payments are made at the end of the payment periods, while it is called **PERPETUITY-DUE** if the payments are made at the beginning of the payment periods

Def. Deferred perpetuity: A deferred perpetuity is one for which the first payment starts some time in the future.

EX19:



A is a semi-annual perpetuity-due, **B** is an annual perpetuity-immediate deferred 12 years (or perpetuity-due deferred 13 years)

MAIN GOAL: Also for a perpetuity it is of interest to determine its **present value**, while obviously the final value cannot be calculated.

PRESENT VALUE OF A PERPETUITY

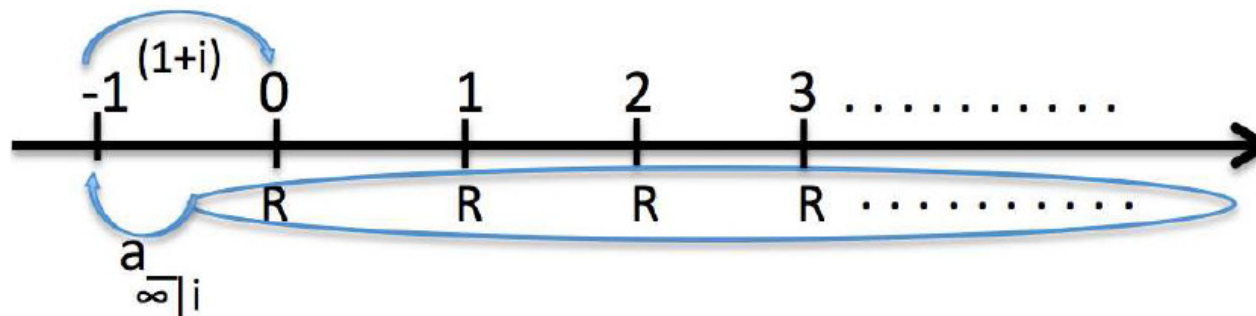
To calculate the present value of a perpetuity, we note that, as $v < 1, v^n \rightarrow 0$ as $n \rightarrow \infty$. Thus, from the formula of the present value of an annuity

we obtain

$$\lim_{n \rightarrow +\infty} \frac{1 - (1 + i)^{-n}}{i} = \frac{1}{i} = a_{\infty|i}.$$

Called «a angle infinity at i»

while for the case when the first payment is made immediately, we have



$$Ra_{\infty|i}(1 + i) = R\ddot{a}_{\infty|i}$$

and similarly the formula for the deferred case can be obtained.

PROPOSED EXERCISES

1. Calculate the present value of an ordinary perpetuity of amount 100 euro payable quarterly at the annual rate of interest of 9%.
2. Find the present value of a perpetuity-due of 200 euro per semester starting after 3 semesters, if the monthly interest rate is 1%.
3. Consider an immediate-perpetuity of 300 euro, payable every 6 months starting after 4 years, interest rate 7%. Determine its value after 10 years.
4. Consider a semi-annual perpetuity due of amount 50 being $i_2=0.05$. Determine the PV.
5. Consider a perpetuity of amount 10, payment every year, first payment at time 3. Determine the PV and the value at time 10 being $i_2=0.04$.

INTERNAL RATE OF RETURN

Consider a project represented in terms of flows-deadlines as follows:

$$OF = \{(x_0, x_1, x_2, \dots, x_n); (t_0, t_1, t_2, \dots, t_n)\}$$

Where $x_j > 0$ if at time t_j there will be an entry while $x_j < 0$ if we have an output at time t_j .

We want to establish which is the interest rate associated to OF so we define the following.

Def: Internal rate of return (IRR) or yeald rate. The IRR is the rate of interest (if it exists unique) such that the present value of the financial project is equated to zero.

Notice that: 1. at the IRR the present values of the entries is equal to the present value associated to the exits; 2. normally it is required to calculate the IRR referred to one year.

Given the following OF

$$OF = \{(x_0, x_1, x_2, \dots, x_n); (0, 1, 2, \dots, n)\}$$

Then the IRR exists iff the following polynomial admits a unique zero

$$P_n(v) = x_0 + x_1 v + x_2 v^2 + \dots + x_n v^n, \quad v = \frac{1}{1+i} \in (0, 1)$$

NOTICE THAT: There is generally no analytic solution for i to the equation $PV=0$ when $n > 2$, and numerical methods have to be used. MatLab can be used to compute the answer easily.

You can use MatLab to calculate IRR of a cash flow

VECTORS

A **vector** composed by n elements is given by n ordered real numbers

Row vector Ex: $\underline{x}=(1, 3, -4, 11)$ composed by 4 elements (dimension 4)

Column vector Ex:

$$\underline{y} = \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix} \text{ composed by 3 elements (dimension 3)}$$

A real number $z=(133)$ is a row or column vector having one element

In Computer Science (and thus in Matlab), a vector is also named **array**.

SAVE A VECTOR

ROW VECTOR: list the elements separated by a **space** or a **comma** , inside square brackets

Save the row vector (10,-1,5)

```
>> A=[10 -1 5]
```

COLUMN VECTOR: list the elements separated by a **semicolon** ; inside square brackets (or write a row vector and put **apostrophe** ' at the end)

Save the column vector with elements 3, 7, 9

```
>> B=[3;7;9]
```

WORK WITH ARRAYS - I

Notice: each element of a vector (or array) is identified by an **index** (which defines its position)

Hence: to **access to an element** of the vector the index must be specified between rounded bracket

EX: to select the second element of vector B

```
>> b=B (2)
```

Notice: in such a way you can modify the value of an element of a vector

EX: change the value of the first element of vector A into 7

```
>> A (1) =7
```

WORK WITH ARRAYS - II

Notice: it is also possible to **delete an element from a vector**
By substituting the selected element with an empty vector **[]**

Es: delete the third element from vector MM

```
>> MM(3) = []
```

EXERCISE

Save the following vectors:

$$x = (-2, \sqrt{3}, e, 10); y = \begin{pmatrix} 0 \\ -1 \end{pmatrix}; z = \begin{pmatrix} 3/5 \\ 7 \\ -4 \end{pmatrix}$$

Which is their dimension?

Transform the vector x in a column vector

Change the second element of vector y into 5

Delete the first element of z

SOLUTION - I

```
>> x = [-2 sqrt(3) exp(1) 10]
```

```
x =
```

```
-2.0000 1.7321 2.7183 10.0000
```

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

```
y =
```

```
0
```

```
-1
```

```
>> z = [3/5, 7, -4]'
```

```
z =
```

```
0.6000
```

```
7.0000
```

```
-4.0000
```

Part I: we declare all the arrays

SOLUTION - II

Part II: we find out their dimensions with the **size** function

```
>> size(x), size(y)
```

```
ans =
```

```
1 4
```

```
ans =
```

```
2 1
```

```
>> size(z)
```

```
ans =
```

```
3 1
```

We can write multiple instruction in the same command line. They can be separates by commas (the result is printed) or semicolons (the result is not printed).

The array z has three rows and one column.

SOLUTION - III

Part III: the remaining part of the exercise.

```
>> x = x'
```

```
x =
```

```
-2.0000
```

```
1.7321
```

```
2.7183
```

```
10.0000
```

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

```
y =
```

```
0
```

```
5
```

```
>> z(1) = []
```

```
z =
```

```
7
```

```
-4
```

THE irr FUNCTION

The command **irr** calculates the internal rate of return for a series of **periodic** cash flows. The obtained IRR is related to the period of the payments.

Syntax: Return = irr(CashFlow)

Where **CashFlow** is a **row vector** containing a stream of periodic cash flows. The first entry in CashFlow is the initial amount (at time t), while the following entries are the payment at time $t+1, t+2, \dots, t+n$.

EX: Find the internal rate of return. The initial investment is 100000 and the following cash flows represent the yearly income from the investment. Year 1:10000, Year 2: 20000, Year 3:30000, Year 4: 40000, Year 5 : 50000.

Since this flow is periodic we can use the command irr and since the periodicity is yearly then we will obtain the annual IRR.

First of all we define the cash-flow as a vector:

```
>> CF=[-100000 10000 20000 30000 40000 50000]
```

```
CF =
```

```
    -100000    10000    20000    30000    40000    50000
```

Then we give the instruction to obtain i (the annual IRR):

```
>> i=irr(CF)
```

```
i =
```

```
    0.1201
```

Consider the following cash flow $\{(-30,10,25);(1,2,2.5)\}$ that is **not periodic**, then we need to use a different command and to specify to MatLab the dates.

You can use MatLab to calculate IRR of a non periodic cash flow: the command **xirr** calculates the internal rate of return for a series of non-**periodic** cash flows. The given IRR is annual and in default it is computed on actual basis (actual number of days)

Syntax: `Return = xirr(CashFlow, CashFlowDates)`

The additional input **CashFlowDates** is a **column vector** of serial date numbers of the same size as CashFlow.

Each element of CashFlowDate represents the dates of the corresponding column of CashFlow. It must be defined as a string **'mm/gg/aaaa'** and MatLab will convert it in a number.

Furthermore, recall that to write a column vector it must be used square brackets with elements separated by semicolon.

For the initial case: dates (1,2,2.5) can be defined as:

```
CFD=['01/01/2000';'01/01/2001';'07/01/2001']
```

Where we fixed a starting date and we opportunely determined the following dates

Notice that: number of days in both a period and a year is the actual number of days.

EX: Find the internal rate of return. The project is as follows where the time is in terms of months. $OF = \{(-30, -20, 28, 32); (0, 2, 2.5, 3)\}$

We first define the row vector of the cash flow:

```
>> cf = [-30 -20 28 32]
```

```
cf =
```

```
   -30   -20    28    32
```

Then we define the column vector of the dates, we consider a start date and we set the following dates:

```
>> cfd = ['01/01/2000'; '03/01/2000'; '03/15/2000'; '04/01/2000']
```

```
cfd =
```

```
01/01/2000
```

```
03/01/2000
```

```
03/15/2000
```

```
04/01/2000
```

Finally we use command `xirr` to obtain the annual IRR:

```
>> xirr(cf, cfd)
```

```
ans =
```

```
2.0271
```

That is $i^* = 202.71\%$

PROPOSED EXERCISES

- 1.** A project requires an initial cash outlay of 2000 euro and is expected to generate 800 euro at the end of year 1 and 1600 at the end of year 2, at which time the project will terminate. Calculate the IRR of the project (analytically).
- 2.** An investor pays 5000 euro today and he will receive 1200 euro at the end of each year for 5 year. Calculate the IRR of his investment (use MatLab).
- 3.** Determine the annual IRR for the following project: $\{(-100, 30, 80); (0, 2, 4)\}$ both analytically and by using MatLab then compare the results (the results can be slightly different since we use commercial year while MatLab uses actual calendar).
- 4.** Determine the annual IRR for the following project using MatLab: $\{(-100, 130, 40, -50); (0, 2, 3, 6)\}$.