NUMERICAL METHODS AND PROGRAMMING **2024/2025**

Storage of numbers and algorithms **(PROBLEMS 1)**

1.— Write in fixed point and floating point, and in the denary, binary, octal and hexadecimal number bases the following numbers:

a) 5.3125
b) 0,1
c) 1/3
d) π
e) e
f) 52745916

Solution 1.a) We firstly change to binary base to make the conversion to octal and hexadecimal easier. The integer part and the decimal part are translated separately. For the integer part:

$$\begin{array}{rcl} 5 : 2 = 2 + \underbrace{1}\ / \ 2 & & \\ 2 : 2 = 1 + \underbrace{0}\ / \ 2 & \Rightarrow & 5 = (101)_2 \\ 1 : 2 = 0 + \underbrace{1}\ / \ 2 & & \end{array}$$

To change to octal we take the digits in sets of 3 ($8{=}2^3$) and for hexadecimal in sets of 4 ($16{=}2^4$), es decir:

$$5 = (\underbrace{101})_2 \quad = \quad (5)_8$$

$$5 = (\underbrace{0101})_2 \quad = \quad (5)_{16}$$

For the decimal part:

$$\begin{array}{rcl} 0{,}3125 \times 2 = \underbrace{0}\,{,}6250 & & \\ 0{,}6250 \times 2 = \underbrace{1}\,{,}2500 & \Rightarrow & 0{,}3125 = (0{,}0101)_2 \\ 0{,}2500 \times 2 = \underbrace{0}\,{,}5000 & & \\ 0{,}5000 \times 2 = \underbrace{1}\,{,}0000 & & \end{array}$$

The same way for the octal and hexadecimal bases:

$$0{,}3125 = (0.\underbrace{010}\,\underbrace{100})_2 = (0{,}24)_8$$

$$0{,}3125 = (0.\underbrace{0101})_2 = (0{,}5)_{16}$$

Thus:

$$\boxed{\begin{array}{ccccc} 5{,}3125 & = & (0{,}53125)\,10^1 & = \\ (101{,}0101)_2 & = & (0{,}1010101)_2\,2^3 & = \\ (5{,}24)_8 & = & (0{,}524)_8\,8^1 & = \\ (5{,}5)_{16} & = & (0{,}55)_{16}\,16^1 & \end{array}}$$

Solution 1.b)

$$\begin{array}{rcrcl} 0{,}1 & \times & 2 & = & \underbrace{0}\,{,}2 \\ 0{,}2 & \times & 2 & = & \underbrace{0}\,{,}4 \\ 0{,}4 & \times & 2 & = & \underbrace{0}\,{,}8 \\ 0{,}8 & \times & 2 & = & \underbrace{1}\,{,}6 \\ 0{,}6 & \times & 2 & = & \underbrace{1}\,{,}2 \\ 0{,}2 & \times & 2 & = & \underbrace{0}\,{,}4 \end{array} \quad \Rightarrow \quad 0{,}1 = (0{,}0\,\overline{0011})_2$$

$$0{,}1 = (0{,}0\,\overline{0011})_2 = (0.\underbrace{000}\,\underbrace{\overline{110}\,\overline{011}\,\overline{001}\,\overline{100}})_2 = (0{,}0\overline{6314})_8$$

$$0{,}1 = (0.\underbrace{0001}\,\underbrace{\overline{1001}})_2 = (0{,}1\overline{9})_{16}$$

Then:

$$\boxed{\begin{array}{ccccc} 0{,}1 & = & (0{,}1)\,10^0 & = \\ (0{,}0\overline{0011})_2 & = & (0.\overline{1100})_2\,2^{-3} & = \\ (0{,}0\overline{6314})_8 & = & (0.\overline{6314})_8\,8^{-1} & = \\ (0{,}1\overline{9})_{16} & = & (0{,}1\overline{9})_{16}\,16^0 & \end{array}}$$

Solution 1.c)

$$1/3 = 0.\overline{3}$$

$$\begin{array}{rcrcl} 0.\overline{3} & \times & 2 & = & \underbrace{0}\,.\overline{6} \\ 0.\overline{6} & \times & 2 & = & \underbrace{1}\,.\overline{3} \\ 0.\overline{3} & \times & 2 & = & \underbrace{0}\,.\overline{6} \end{array} \quad \Rightarrow \quad 0.\overline{3} = (0.\overline{01})_2$$

$$0.\overline{3} = (0.\underbrace{\overline{010}\,\overline{101}})_2 = (0.\overline{25})_8$$

$$0.\overline{3} = (0.\underbrace{\overline{0101}})_2 = (0.\overline{5})_{16}$$

Thus:

$$\boxed{\begin{array}{ccccc} 0.\overline{3} & = & (0.\overline{3})\,10^0 & = \\ (0.\overline{01})_2 & = & (0.\overline{10})_2\,2^{-1} & = \\ (0.\overline{25})_8 & = & (0.\overline{25})_8\,8^0 & = \\ (0.\overline{5})_{16} & = & (0.\overline{5})_{16}\,16^0 & \end{array}}$$

Solution 1.d)

$$\pi \approx 3{,}141592653589793$$

$$\begin{array}{rcrcrcrcr} 3 & : & 2 & = & 1 & + & \underbrace{1} & / & 2 \\ 1 & : & 2 & = & 0 & + & \underbrace{1} & / & 2 \end{array} \qquad \Rightarrow \qquad 3 = (11)_2$$

$$3 = (\underbrace{011})_2 = (3)_8$$

$$3 = (\underbrace{0011})_2 = (3)_{16}$$

The conversion of the decimal part is shown in table 1.

$$0{,}141592653589793... = (0{,}00100100001111110110101010100010001...)_2 \; =$$

$$(0.\underbrace{001}\,\underbrace{001}\,\underbrace{000}\,\underbrace{011}\,\underbrace{111}\,\underbrace{101}\,\underbrace{101}\,\underbrace{010}\,\underbrace{100}\,\underbrace{010}\,\underbrace{001}...)_2 = (0{,}11037552421...)_8 =$$

$$(0.\underbrace{0010}\,\underbrace{0100}\,\underbrace{0011}\,\underbrace{1111}\,\underbrace{0110}\,\underbrace{1010}\,\underbrace{1000}\,\underbrace{1000}\,1...)_2 = (0{,}243F6A88...)_{16}$$

Then:

$3{,}141592653589793...$	$=$	$(0{,}3141592653589793...)\ 10^1$
$(11{,}00100100001111110110101010100010001...)_2$	$=$	
$= (0{,}11001001000011111101101010100010001...)_2\ 2^2$		
$(3{,}11037552421...)_8$	$=$	$(0{,}311037552421...)_8\ 8^1$
$(3{,}243F6A99...)_{16}$	$=$	$(0{,}3243F6A99...)_{16}\ 16^1$

Solution 1.e)

$$e = 2{,}718281828459045$$

$$\begin{array}{rcrcrcrcr} 2 & : & 2 & = & 1 & + & \underbrace{0} & / & 2 \\ 1 & : & 2 & = & 0 & + & \underbrace{1} & / & 2 \end{array} \qquad \Rightarrow \qquad 2 = (10)_2$$

$$2 = (\underbrace{010})_2 = (2)_8$$

$$2 = (\underbrace{0010})_2 = (2)_{16}$$

The conversion of the decimal part is shown in table 2.

$$0{,}718281828459045... = (0{,}10110111111000010101000101100010 1...)_2 \; =$$

$$(0.\underbrace{101}\,\underbrace{101}\,\underbrace{111}\,\underbrace{110}\,\underbrace{000}\,\underbrace{101}\,\underbrace{010}\,\underbrace{001}\,\underbrace{011}\,\underbrace{000}\,\underbrace{101}...)_2 = (0{,}55760521305...)_8 =$$

$$(0.\underbrace{1011}\,\underbrace{0111}\,\underbrace{1110}\,\underbrace{0001}\,\underbrace{0101}\,\underbrace{0001}\,\underbrace{0110}\,\underbrace{0010}\,1...)_2 = (0.B7E15162...)_{16}$$

Tabla 1: Decimal part conversion of π.

1	0.141592653589793	0.283185307179586	0
2	0.283185307179586	0.566370614359172	0
3	0.566370614359172	1.132741228718340	1
4	0.132741228718344	0.265482457436688	0
5	0.265482457436688	0.530964914873376	0
6	0.530964914873376	1.061929829746750	1
7	0.061929829746752	0.123859659493505	0
8	0.123859659493505	0.247719318987009	0
9	0.247719318987009	0.495438637974019	0
10	0.495438637974019	0.990877275948037	0
11	0.990877275948037	1.981754551896070	1
12	0.981754551896074	1.963509103792150	1
13	0.963509103792148	1.927018207584300	1
14	0.927018207584297	1.854036415168590	1
15	0.854036415168594	1.708072830337190	1
16	0.708072830337187	1.416145660674370	1
17	0.416145660674374	0.832291321348748	0
18	0.832291321348748	1.664582642697500	1
19	0.664582642697496	1.329165285394990	1
20	0.329165285394993	0.658330570789985	0
21	0.658330570789985	1.316661141579970	1
22	0.316661141579971	0.633322283159941	0
23	0.633322283159941	1.266644566319880	1
24	0.266644566319883	0.533289132639765	0
25	0.533289132639765	1.066578265279530	1
26	0.066578265279532	0.133156530559063	0
27	0.133156530559063	0.266313061118126	0
28	0.266313061118126	0.532626122236251	0
29	0.532626122236251	1.065252244472500	1
30	0.065252244472504	0.130504488945007	0
31	0.130504488945007	0.261008977890015	0
32	0.261008977890015	0.522017955780029	0
33	0.522017955780029	1.044035911560050	1
34	0.044035911560059	0.088071823120117	0
35	0.088071823120117	0.176143646240234	0
36	0.176143646240234	0.352287292480469	0
37	0.352287292480469	0.704574584960937	0
38	0.704574584960937	1.409149169921870	1
39	0.409149169921875	0.818298339843750	0
40	0.818298339843750	1.636596679687500	1

Tabla 2: Decimal part conversion of e.

1	0.718281828459045	1.436563656918090	1
2	0.436563656918090	0.873127313836180	0
3	0.873127313836180	1.746254627672360	1
4	0.746254627672360	1.492509255344720	1
5	0.492509255344720	0.985018510689439	0
6	0.985018510689439	1.970037021378880	1
7	0.970037021378879	1.940074042757760	1
8	0.940074042757757	1.880148085515510	1
9	0.880148085515515	1.760296171031030	1
10	0.760296171031030	1.520592342062060	1
11	0.520592342062059	1.041184684124120	1
12	0.041184684124119	0.082369368248237	0
13	0.082369368248237	0.164738736496474	0
14	0.164738736496474	0.329477472992949	0
15	0.329477472992949	0.658954945985897	0
16	0.658954945985897	1.317909891971790	1
17	0.317909891971794	0.635819783943589	0
18	0.635819783943589	1.271639567887180	1
19	0.271639567887178	0.543279135774355	0
20	0.543279135774355	1.086558271548710	1
21	0.086558271548711	0.173116543097422	0
22	0.173116543097422	0.346233086194843	0
23	0.346233086194843	0.692466172389686	0
24	0.692466172389686	1.384932344779370	1
25	0.384932344779372	0.769864689558744	0
26	0.769864689558744	1.539729379117480	1
27	0.539729379117488	1.079458758234970	1
28	0.079458758234978	0.158917516469955	0
29	0.158917516469955	0.317835032939911	0
30	0.317835032939911	0.635670065879821	0
31	0.635670065879821	1.271340131759640	1
32	0.271340131759644	0.542680263519287	0
33	0.542680263519287	1.085360527038570	1
34	0.085360527038574	0.170721054077148	0
35	0.170721054077148	0.341442108154297	0
36	0.341442108154297	0.682884216308593	0
37	0.682884216308593	1.365768432617180	1
38	0.365768432617187	0.731536865234375	0
39	0.731536865234375	1.463073730468750	1
40	0.463073730468750	0.926147460937500	0

Thus:

$2{,}718281828459045...$	$=$	$(0{,}2718281828459045...)\ 10^1$
$(10{,}10110111111000010101000101100010 1...)_2$	$=$	
$= (0{,}10101101111111000010101000101100010 1...)_2\ 2^2$		
$(2{,}55760521305...)_8$	$=$	$(0{,}255760521305...)_8\ 8^1$
$(2.B7E15162...)_{16}$	$=$	$(0{,}2B7E15162...)_{16}\ 16^1$

Solution 1.f) In this particular case is easier to convert the number to hexadecimal and then to octal and binary:

$$
\begin{array}{rcrcrcrcl}
52745916 & : & 16 & = & 3296619 & + & \underbrace{12} & / & 16 \\
3296619 & : & 16 & = & 206038 & + & \underbrace{11} & / & 16 \\
206038 & : & 16 & = & 12877 & + & \underbrace{6} & / & 16 \\
12877 & : & 16 & = & 804 & + & \underbrace{13} & / & 16 \\
804 & : & 16 & = & 50 & + & \underbrace{4} & / & 16 \\
50 & : & 16 & = & 3 & + & \underbrace{2} & / & 16 \\
3 & : & 16 & = & 0 & + & \underbrace{3} & / & 16
\end{array}
\qquad \Rightarrow \qquad 52745916 = (324D6BC)_{16}
$$

Converting now to binary and octal:

$$(324D6BC)_{16} = (\underbrace{0011}\,\underbrace{0010}\,\underbrace{0100}\,\underbrace{1101}\,\underbrace{0110}\,\underbrace{1011}\,\underbrace{1100})_2$$

$$(\underbrace{011}\,\underbrace{001}\,\underbrace{001}\,\underbrace{001}\,\underbrace{101}\,\underbrace{011}\,\underbrace{010}\,\underbrace{111}\,\underbrace{100})_2 = (311153274)_8$$

Therefore:

52745916	$=$	$(0{,}52745916)\ 10^8$
$(11001001001101011010111100)_2$	$=$	
$= (0{,}11001001001101011010111100)_2\ 2^{26}$		
$(311153274)_8$	$=$	$(0{,}311153274)_8\ 8^9$
$(324D6BC)_{16}$	$=$	$(0{,}324D6BC)_{16}\ 16^7$

2.— The above values are to be stored in the variables `A, UD, UT, PI, E` and `GR` of a FORTRAN program:

```
REAL*4 A,UD,UT,PI,E,GR
A = 5.3125
UD = 0.1
UT = 0.333333
PI = 3.14159
E = 2.71828
GR = 52745916.
...
```

If for the mantissa of the numbers variables of type `REAL*4` are used, using 3 bytes (including the sign)

a) Obtain the values that are actually stored in each case.

b) Compare the stored values with the corresponding exact values, indicating the sources of error.

c) Compare the rounding error produced in each case with its upper bound.

d) Estimate how many significant figures should have been used to write the numbers 1/3, π and e. Repeat the estimation for the case in which the variables are to be stored as `REAL*8`.

Solution 2.a) We convert the constants to binary, in this case with the decimal digits the statement indicates. We express the numbers in floating point and group them in sets of 1 byte (8 digits):

$$\begin{array}{lcl}
5{,}3125 & = & (0{,}10101010|00000000|00000000|0)_2\ 2^3 \\
0{,}1 & = & (0{,}11001100|11001100|11001100|1...)_2\ 2^{-3} \\
0{,}333333 & = & (0{,}10101010|10101010|10011111|1...)_2\ 2^{-1} \\
3{,}14159 & = & (0{,}11001001|00001111|11001111|1...)_2\ 2^2 \\
2{,}71828 & = & (0{,}10101101|11111000|01001100|1...)_2\ 2^2 \\
52745916, & = & (0{,}11001001|00110101|10101111|0...)_2\ 2^{26}
\end{array}$$

Assuming 23 bits for the mantissa, with 1 for the sign and approximation by rounding, the actually stored values would be:

$$\begin{array}{lcl}
\texttt{A} & \rightarrow & (0{,}10101010|00000000|0000000)_2\ 2^3 \\
\texttt{UD} & \rightarrow & (0{,}11001100|11001100|1100110)_2\ 2^{-3} \\
\texttt{UT} & \rightarrow & (0{,}10101010|10101010|10\underline{10000})_2\ 2^{-1} \\
\texttt{PI} & \rightarrow & (0{,}11001001|00001111|110\underline{1000})_2\ 2^2 \\
\texttt{E} & \rightarrow & (0{,}10101101|11111000|0100110)_2\ 2^2 \\
\texttt{GR} & \rightarrow & (0{,}11001001|00110101|101\underline{1000})_2\ 2^26
\end{array}$$

NOTE: Actually, since the first bit is always a 1 it is not stored, we will not take this into account for the sake of simplicity. The underlined part is the one affected by rounding.

If we convert again to decimal we have the values actually stored:

`A`	$\equiv$	5,3125	$=$	5,3125
`UD`	$\equiv$	0,0999999940395	$\neq$	0,1
`UT`	$\equiv$	0,333333015442	$\neq$	0,333333
`PI`	$\equiv$	3,14159011841	$\neq$	3,14159
`E`	$\equiv$	2,71827983856	$\neq$	2,71828
`GR`	$\equiv$	52745920.	$\neq$	52745916.

Solution 2.b)

* 5.3125 is exactly stored in `A` since its mantissa occupies less than 23 bits.
* 0.1 and 52745916 are stored in `UD` and `GR` respectively with a rounding error due to their mantissas occupying more than 23 significant digits. Rounding with 23 bits losses information.

* 0.333333, 3.14159 and 2.71828 are stored in `UT`, `PI` and `E` respectively with a rounding error, again because their mantissa requires more than 23 significant digits. Also this values have inherent errors since we are actually trying to use the numbers 1/3, π and e.

Solution 2.c) Let $\overline{x}$ be the real number we want to represent, $\widehat{x}$ the value we save and x the value actually stored ($\overline{x} = \pi, \hat{x} = 3{,}14159, x = 3{,}14159011841$). We define:

$$\begin{cases} \text{error de redondeo} & = & r_x^A & = & \frac{\widehat{x}-x}{\widehat{x}} \\ \text{error inherente} & = & r_x^I & = & \frac{\overline{x}-\widehat{x}}{\overline{x}} \\ \\ \text{error total} & = & r_x & = & \frac{\overline{x}-x}{x} = \left(\frac{\overline{x}-\widehat{x}}{\overline{x}}\right) + \left(\frac{\widehat{x}-x}{\widehat{x}}\right)\cancel{\left(\frac{\widehat{x}}{\overline{x}}\right)}^{\approx 1} \approx r_x^I + r_x^A \end{cases}$$

The upper bound for the storing error is the Machine Epsilon:

$$\left|r_x^A\right| \leq \frac{1}{2}\, 2^{-m+2} \Rightarrow \left|r_x^A\right| \leq r_M = 2^{-23} \approx 1{,}1909\ 10^{-7}$$

Rounding errors:

$$\begin{array}{rcccrcl} r_{\mathtt{A}}^A & = & \frac{5{,}3125-5{,}3125}{5{,}3125} & = & 0 & & \\ r_{\mathtt{UD}}^A & = & \frac{0{,}1-0{,}0999999940395}{0{,}1} & \approx & 0{,}596050\ 10^{-7} & \rightarrow & \left|r_{\mathtt{UD}}^A\right| \approx 0{,}50\ r_M \\ r_{\mathtt{UT}}^A & = & \frac{0{,}333333-0{,}333333015442}{0{,}333333} & \approx & -0{,}463260\ 10^{-7} & \rightarrow & \left|r_{\mathtt{UT}}^A\right| \approx 0{,}39\ r_M \\ r_{\mathtt{PI}}^A & = & \frac{3{,}14159-3{,}14159011841}{3{,}14159} & \approx & -0{,}376911\ 10^{-7} & \rightarrow & \left|r_{\mathtt{PI}}^A\right| \approx 0{,}32\ r_M \\ r_{\mathtt{E}}^A & = & \frac{2{,}71828-2{,}71827983856}{2{,}71828} & \approx & 0{,}593905\ 10^{-7} & \rightarrow & \left|r_{\mathtt{E}}^A\right| \approx 0{,}50\ r_M \\ r_{\mathtt{GR}}^A & = & \frac{52745916.-52745920.}{52745916.} & \approx & -0{,}758353\ 10^{-7} & \rightarrow & \left|r_{\mathtt{GR}}^A\right| \approx 0{,}64\ r_M \end{array}$$

Inherent errors:

$$\begin{array}{rcccrcl} r_{\mathtt{A}}^I & = & r_{\mathtt{UD}}^I = r_{\mathtt{GR}}^I & = & 0 & & \\ r_{\mathtt{UT}}^I & = & \frac{1/3-0{,}333333}{1/3} & \approx & 1{,}000000\ 10^{-6} & \rightarrow & \left|r_{\mathtt{UT}}^I\right| \approx 8{,}4\ r_M \\ r_{\mathtt{PI}}^I & = & \frac{\pi-3{,}14159}{\pi} & \approx & 0{,}844664\ 10^{-6} & \rightarrow & \left|r_{\mathtt{PI}}^I\right| \approx 7{,}1\ r_M \\ r_{\mathtt{E}}^I & = & \frac{e-2{,}71828}{e} & \approx & 0{,}672653\ 10^{-6} & \rightarrow & \left|r_{\mathtt{E}}^I\right| \approx 5{,}6\ r_M \end{array}$$

Global errors:

$$\begin{array}{rcccrcl} r_{\mathtt{A}} & = & r_{\mathtt{A}}^A + r_{\mathtt{A}}^I & = & 0 & & \\ r_{\mathtt{UD}} & = & r_{\mathtt{UD}}^A + r_{\mathtt{UD}}^I & \approx & 0{,}596050\ 10^{-7} & \rightarrow & |r_{\mathtt{UD}}| \approx 0{,}50\ r_M \\ r_{\mathtt{UT}} & = & r_{\mathtt{UT}}^A + r_{\mathtt{UT}}^I & \approx & 9{,}53674\ 10^{-7} & \rightarrow & |r_{\mathtt{UT}}| \approx 8\ r_M \\ r_{\mathtt{PI}} & = & r_{\mathtt{PI}}^A + r_{\mathtt{PI}}^I & \approx & 8{,}069729\ 10^{-7} & \rightarrow & |r_{\mathtt{PI}}| \approx 6{,}8\ r_M \\ r_{\mathtt{E}} & = & r_{\mathtt{E}}^A + r_{\mathtt{E}}^I & \approx & 7{,}320435\ 10^{-7} & \rightarrow & |r_{\mathtt{E}}| \approx 6{,}1\ r_M \\ r_{\mathtt{GR}} & = & r_{\mathtt{GR}}^A + r_{\mathtt{GR}}^I & \approx & -0{,}758353\ 10^{-7} & \rightarrow & |r_{\mathtt{GR}}| \approx 0{,}64\ r_M \end{array}$$

Solution 2.d) We can see that the inherent errors of `UT, PI` and `E` are very high. Thus, we observe that $1/3$, π and e were introduced with high inherent errors in comparison with the Machine Epsilon r_M.

We determine the number of significant digits s we should have used in base 10 in order for the error to be the same order of magnitude of that of the Machine Epsilon:

$$\frac{1}{2}\,10^{-(s+1)+2} \approx \frac{1}{2}\,2^{-24+2} \Leftrightarrow 10^{-s+1} \approx 2^{-22} \Leftrightarrow s \geq 1 + 22\log_{10} 2 = 7{,}62266 \Rightarrow \boxed{s \geq 8}$$

Thus, we should have written:

$$\begin{aligned} \texttt{UT} &= 0{,}33333333 \\ \texttt{PI} &= 3{,}1415927 \\ \texttt{E} &= 2{,}7182818 \end{aligned}$$

In case of using `REAl*8` variables: $r_M \approx (1/2)\ 2^{-53+2}$

$$\frac{1}{2}\,10^{-(s+1)+2} \approx \frac{1}{2}\,2^{-53+2} \Leftrightarrow 10^{-s+1} \approx 2^{-51} \Leftrightarrow s \geq 1 + 51\log_{10} 2 = 16{,}3525 \Rightarrow \boxed{s \geq 17}$$

and we should have introduced:

$$\begin{aligned} \texttt{UT} &= 0{,}33333333333333333 \\ \texttt{PI} &= 3{,}1415926535897932 \\ \texttt{E} &= 2{,}7182818284590452 \end{aligned}$$

3.— Compare the values stored in the variables `GDIF` and `NGDIF` in the following FORTRAN program:

```
REAL*4 GDIF
INTEGER*4 NGDIF
GDIF = 52745916. + 15.
GDIF = GDIF - 52745916.
...
NGDIF = 52745916 + 15
NGDIF = NGDIF - 52745916
...IF(GDIF.EQ.FLOAT(NGDIF)) STOP...
```

If for the mantissa of the variables `REAL*4`, 3 bytes are used (including the sign), will the program stop when executing the instruction `IF`? What conclusions can be drawn from this example?

Solution 3. For each of the stored numbers in floating point, except for 0, the first bit after the decimal point is always a 1, therefore is not necessary to store it, thus we have an extra bit $\Rightarrow$ m=25.

Considering this, the number 52745916. is stored as:

$$\texttt{GR} = (0{,}11001001|00110101|10101111)_2\ 2^{26}$$

and number 15. is stored as:

$$(0{,}11110000|00000000|00000000)_2\ 2^{4}$$

then, the statement `GDIF=52745916.+15.` will cast:

$$\begin{array}{r} 11001001|00110101|10101111|00 \\ +\quad 11|11 \\ \hline 11001001|00110101|10110010|11 \end{array}$$

that will be stored as:

$$(0{,}11001001|00110101|10110011)_2\, 2^{26}$$

Then, the statement `GDIF=GDIF-52745916.` will cast:

$$\begin{array}{r} 11001001|00110101|10110011|00 \\ -11001001|00110101|10101111|00 \\ \hline 00000000|00000000|00000100|00 \end{array}$$

which will be stored as:

$$(0{,}10000000|00000000|00000000)_2\, 2^5 = (16.)_{10}$$

However, the operations with integer values will be more accurate because they can store up to 31 binary digits.

Therefore, the program will not stop since `GDIF` will contain the value 16. and `FLOAT(NGDIF)` will give the value 15.

Conclusion: In a computer is not the same to operate with integer or real variables.

4.— A given computer takes a tenth of a second (at most) to:

- **a)** Obtain, through the equations of minimum least squares, the regression line that best approximates 100 data points.
- **b)** Calculate the modulus of a vector of 500 elements.
- **c)** Sort from highest to lowest a list of 50 numbers by the *"bubble sort"* method.
- **d)** Obtain the product of a square matrix of size 20 times a vector.
- **e)** Multiply two square matrices of size 8.
- **f)** Determine (by algebraic definition) the determinant of a square matrix of size 5.
- **g)** Find by sequential search a telephone number in a list (alphabetically sorted) of 100 subscribers.
- **h)** Find by bisection a telephone number in a list (alphabetically sorted) of 100 subscribers.

What is going to be the computational time if the size of the problem (size of the matrix, number of data, for each case) is ten, a hundred, a thousand, ten thousand, a hundred thousand or a million times larger?

Solution 4. We have to determine the computational complexity of each one of the items to analyze their dependency on the size of the problem.

Solution 4.a) Given $\{x_k, y_k\}_{k=1,\dots,m}$, we want to approximate $y(x)$ by a first order polynomial, $P_1(x) = ax + b$, so that $Q = \sum_{k=1}^{m} (y_k - P_1(x_k))^2$ verifies that $\frac{\partial Q}{\partial a} = 0$ y $\frac{\partial Q}{\partial b} = 0$.

Thus:

$$\left\{ \begin{array}{l} \dfrac{\partial Q}{\partial a} = -2 \displaystyle\sum_{k=1}^{m} (y_k - P_1(x_k)) x_k \\ \\ \dfrac{\partial Q}{\partial b} = -2 \displaystyle\sum_{k=1}^{m} (y_k - P_1(x_k)) \end{array} \right\} \Rightarrow$$

$$\Rightarrow \left\{ \begin{array}{l} -2 \displaystyle\sum_{k=1}^{m} (y_k - (a\, x_k + b)) x_k = 0 \\ \\ -2 \displaystyle\sum_{k=1}^{m} (y_k - (a\, x_k + b))) = 0 \end{array} \right\} \Rightarrow \left[\begin{array}{cc} m & \displaystyle\sum_{k=1}^{m} x_k \\ \\ SIM & \displaystyle\sum_{k=1}^{m} x_k^2 \end{array} \right] \left\{ \begin{array}{c} b \\ \\ a \end{array} \right\} = \left\{ \begin{array}{c} \displaystyle\sum_{k=1}^{m} y_k \\ \\ \displaystyle\sum_{k=1}^{m} x_k\, y_k \end{array} \right\}$$

Then:

$$D = det \left[\begin{array}{cc} m & \displaystyle\sum_{k=1}^{m} x_k \\ \\ SIM & \displaystyle\sum_{k=1}^{m} x_k^2 \end{array} \right] = m \sum_{k=1}^{m} x_k^2 - \left(\sum_{k=1}^{m} x_k \right)^2$$

$$b = det \left[\begin{array}{cc} \displaystyle\sum_{k=1}^{m} y_k & \displaystyle\sum_{k=1}^{m} x_k \\ \\ \displaystyle\sum_{k=1}^{m} x_k\, y_k & \displaystyle\sum_{k=1}^{m} x_k^2 \end{array} \right] / D = \left(\sum_{k=1}^{m} y_k \sum_{k=1}^{m} x_k^2 - \sum_{k=1}^{m} x_k\, y_k \sum_{k=1}^{m} x_k \right) / D$$

$$a = det \left[\begin{array}{cc} m & \displaystyle\sum_{k=1}^{m} y_k \\ \\ \displaystyle\sum_{k=1}^{m} x_k & \displaystyle\sum_{k=1}^{m} x_k\, y_k \end{array} \right] / D = \left(m \sum_{k=1}^{m} x_k\, y_k - \sum_{k=1}^{m} x_k \sum_{k=1}^{m} y_k \right) / D$$

If we operate as follows:

$$\begin{cases} \mu_x = \left(\sum_{k=1}^{m} x_k\right)/m & \rightarrow \ (m-1) \text{ sums}, 1 \text{ division} \\ \mu_y = \left(\sum_{k=1}^{m} y_k\right)/m & \rightarrow \ (m-1) \text{ sums}, 1 \text{ division} \\ \sigma_x^2 = \left(\sum_{k=1}^{m} (x_k-\mu_x)^2\right)/m & \rightarrow \ (m) \text{ subtractions}, (m) \text{ products}, (m-1) \text{ sums}, 1 \text{ division} \\ \sigma_{xy} = \left(\sum_{k=1}^{m} (x_k-\mu_x)(y_k-\mu_y)\right)/m & \rightarrow \ (2m) \text{ subtractions}, (m) \text{ products}, (m-1) \text{ sums}, 1 \text{ division} \\ \hline P_1(x) = \mu_y + \dfrac{\sigma_{xy}}{\sigma_x^2}(x-\mu_x) & \text{TOTAL} = (9m) \text{ operations} \end{cases}$$

Therefore the computational cost will be proportial to $\boxed{\mathcal{O}(m)}$

Solution 4.b) Given a vector $\boldsymbol{v} = \{v_i\}_{i=1,\ldots,m}$:

$$|\boldsymbol{v}| = \left(\sum_{i=1}^{m} v_i^2\right)^{1/2} \Rightarrow \left\{ \begin{array}{l} m \text{ products} \\ (m-1) \text{ sums} \\ 1 \text{ square root} \end{array} \right\} \Rightarrow \text{ TOTAL} = 2m \text{ operations} \Rightarrow \boxed{\mathcal{O}(m)}$$

Solution 4.c)

```
DO I = 1, M − 1
  DO J = M, I + 1, −1
    IF(L(I).LT.L(J))THEN
      LL = l(I)
      L(I) = L(J)
      L(J) = LL
    ENDIF
  ENDDO
ENDDO
```

$$\Rightarrow \underbrace{[(m-1)+(m-2)+\ldots+1]}_{\frac{m(m-1)}{2}} \text{ comparisons}$$

Not every operation requires a change because it depends on the initial order of the data. If they are already sorted they need 0 changes. If every single one would be necessary it would take $\frac{m(m-1)}{2}$ changes.

In any way, the computational complexity would be proportional to $\frac{m(m-1)}{2}$ in general, thus $\boxed{\mathcal{O}(m^2)}$.

Solution 4.d)

$$\boldsymbol{w} = \boldsymbol{A}\boldsymbol{v} \ \Rightarrow \quad w_i = \sum_{j=1}^{m} a_{i,j} v_j \quad ; \quad i = 1, \ldots, m \quad \Rightarrow \quad m \text{ veces} \left\{ \begin{array}{l} m \text{ productos} \\ (m-1) \text{ sumas} \end{array} \right.$$

$$\text{TOTAL} = m(2m-1) \text{ operations} \qquad \Rightarrow \qquad \boxed{\mathcal{O}(m^2)}$$

Solution 4.e)

$$\boldsymbol{C} = \boldsymbol{AB} \Rightarrow \qquad c_{i,j} = \sum_{k=1}^{m} a_{i,k} b_{k,j} \qquad ; \qquad \begin{matrix} i = 1, \ldots, m \\ j = 1, \ldots, m \end{matrix} \qquad \Rightarrow \qquad m^2 \text{ times} \left\{ \begin{matrix} m \text{ products} \\ (m-1) \text{ sums} \end{matrix} \right.$$

$$\text{TOTAL} = m^2(2m-1) \text{ operations} \qquad \Rightarrow \qquad \boxed{\mathcal{O}(m^3)}$$

Solution 4.f)

$$det(\boldsymbol{A}) \; = \; \sum_{j=1}^{m} a_{i,k} \boldsymbol{A}_{k,j} \qquad \Rightarrow \qquad \begin{matrix} 1 \text{ determinant} \\ \text{of order m} \end{matrix} \Rightarrow \left\{ \begin{matrix} m \text{ products} \\ (m-1) \text{ sums} \\ m \text{ determinants of order (m-1)} \end{matrix} \right.$$

Let C_m be the number operations needed to obtain the determinant of order m. Then we know that:

$$C_m = m + (m-1) + m\, C_{m-1} = -1 + m(2 + C_{m-1}) \qquad \text{con} \qquad C_1 = 0$$

Therefore:

$$\begin{array}{lcll}
C_1 & = & -1+1 & \\
C_1 & = & -1+2(2+C_1) & = -1+2(1+1) \\
C_3 & = & -1+3(2+C_2) & = -1+3(1+2(1+1)) \\
C_4 & = & -1+4(2+C_3) & = -1+4(1+3(1+2(1+1))) \\
\ldots & & \ldots & \ldots \\
C_m & = & -1+m(2+C_{m-1}) & = -1+m(1+(m-1)(\ldots(1+4(1+3(1+2(1+1)))))) \\
 & = & -1+m+m(m-1)+\ldots+m(m-1)..,4+m(m-1)..,4\,3+m(m-1)..,4\,3\,2\,2 & \\
 & = & -1+\dfrac{m!}{(m-1)!}+\dfrac{m!}{(m-2)!}+\ldots+\dfrac{m!}{3!}+\dfrac{m!}{2!}+\dfrac{m!}{1!}2 & \\
 & = & -1+m!\underbrace{\left(1+\dfrac{1}{1!}+\dfrac{1}{2!}+\ldots+\dfrac{1}{(m-1)!}\right)}_{\to e \text{ when } m \to \infty} & \\
 & \approx & -1+e\,m! &
\end{array}$$

Thus, the determinant of a matrix of order m has a cost of $(-1 + e\,m!)$ operations, that's is $\boxed{\mathcal{O}(m!)}$

Solution 4.g) We define:

`A(M)` as the array containing the surnames.

`T(M)` as the array containing the corresponding phone numbers.

`AA` as the surname whose phone number we are searching for.

Then:

```
DO I = 1, M
  IF(A(I).EQ.AA)THEN
    WRITE(6,*)'THE PHONE NUMBER IS', T(I)
    STOP
  ENDIF
ENDDO
WRITE(6,*)'NON-EXISTENT PHONE NUMBER'
```

The number of comparisons depends on the position of the number we are searching for in the file. Best case scenario only one comparison would be needed. Worst case, m comparisons. In the average case $m/2$ comparisons.

Thus, the computational cost will be $\boxed{=(m)}$

Solution 4.h) The program will be:

```
IF(A(1).EQ.AA)THEN
    WRITE(6,*)'THE PHONE NUMBER IS', T(1)
    STOP
ELSE IF(A(M).EQ.AA)THEN
    WRITE(6,*)'THE PHONE NUMBER IS', T(M)
    STOP
ELSE IF(A(1).LT.AA).AND.(AA.LT.A(M))THEN
    IMIN = 1
    IMAX = M
    DO WHILE((IMAX - IMIN).GT.1)
        IMED = (IMIN + IMAX)/2
        IF(A(IMED).EQ.AA)THEN
            WRITE(6,*)'THE PHONE NUMBER IS', T(IMED)
            STOP
        ELSE IF(A(IMED).LT.AA)THEN
            IMIN = IMED
        ELSE
            IMAX = IMED
        ENDIF
    ENDDO
ENDIF
WRITE(6,*)'NON-EXISTENT PHONE NUMBER'
```

The number of comparisons to make will be, at maximum, approximately $\log_2 m$, thus the cost is $\boxed{\mathcal{O}(\log_2 m)}$

With every computational cost determined and the data of the statement::

$$\begin{cases} T_a \approx K_a\, m_a & \rightarrow \quad K_a \approx 0{,}1s/100 \\ T_b \approx K_b\, m_b & \rightarrow \quad K_b \approx 0{,}1s/500 \\ T_c \approx K_c\, m_c^2 & \rightarrow \quad K_c \approx 0{,}1s/50^2 \\ T_d \approx K_d\, m_d^2 & \rightarrow \quad K_d \approx 0{,}1s/20^2 \\ T_e \approx K_e\, m_e^3 & \rightarrow \quad K_e \approx 0{,}1s/8^3 \\ T_f \approx K_f\, m_f! & \rightarrow \quad K_f \approx 0{,}1s/5! \\ T_g \approx K_g\, m_g & \rightarrow \quad K_g \approx 0{,}1s/100 \\ T_h \approx K_h\, \log_2 m_h & \rightarrow \quad K_h \approx 0{,}1s/\log_2 100 \end{cases}$$

If the problem is n times larger:

$$\begin{cases} m_a = 100\, n & \rightarrow \quad T_a \approx n\, 0{,}1s \\ m_b = 500\, n & \rightarrow \quad T_b \approx n\, 0{,}1s \\ m_c = 50\, n & \rightarrow \quad T_c \approx n^2\, 0{,}1s \\ m_d = 20\, n & \rightarrow \quad T_d \approx n^2\, 0{,}1s \\ m_e = 8\, n & \rightarrow \quad T_e \approx n^3\, 0{,}1s \\ m_f = 5\, n & \rightarrow \quad T_f \approx \dfrac{(5n)!}{5!}\, 0{,}1s \\ m_g = 100\, n & \rightarrow \quad T_g \approx n\, 0{,}1s \\ m_h = 100\, n & \rightarrow \quad T_h \approx \left(1 + \dfrac{\log_2 n}{\log_2 100}\right)\, 0{,}1s \end{cases}$$

For n=10,100,1000,10000,100000,1000000

n	$\frac{m_a}{T_a}$	$\frac{m_a}{T_a}$	$\frac{m_a}{T_a}$	$\frac{m_a}{T_a}$	$\frac{m_a}{T_a}$	$\frac{m_a}{T_a}$	$\frac{m_a}{T_a}$	$\frac{m_a}{T_a}$
1	$\frac{10^2}{0{,}1s}$	$\frac{5\,10^2}{0{,}1s}$	$\frac{5\,10^1}{0{,}1s}$	$\frac{2\,10^1}{0{,}1s}$	$\frac{8}{0{,}1s}$	$\frac{5}{0{,}1s}$	$\frac{10^2}{0{,}1s}$	$\frac{10^2}{0{,}1}$
10	$\frac{10^3}{1s}$	$\frac{5\,10^3}{1s}$	$\frac{5\,10^2}{10s}$	$\frac{2\,10^2}{10s}$	$\frac{8\,10^1}{1{,}67min}$	$\frac{5\,10^1}{8\,10^{53}\text{years}}$	$\frac{10^3}{1s}$	$\frac{10^2}{0{,}15s}$
10^2	$\frac{10^4}{10s}$	$\frac{5\,10^4}{10s}$	$\frac{5\,10^3}{16{,}7min}$	$\frac{2\,10^3}{16{,}7min}$	$\frac{8\,10^2}{1{,}16\text{days}}$	$--$	$\frac{10^4}{10s}$	$\frac{10^3}{10s}$
10^3	$\frac{10^5}{1{,}67min}$	$\frac{5\,10^5}{1{,}67min}$	$\frac{5\,10^4}{1{,}16\text{días}}$	$\frac{2\,10^4}{1{,}16\text{days}}$	$\frac{8\,10^3}{3{,}17\text{years}}$	$--$	$\frac{10^5}{1{,}67min}$	$\frac{10^4}{0{,}25s}$
10^4	$\frac{10^6}{16{,}7min}$	$\frac{5\,10^6}{16{,}7min}$	$\frac{5\,10^5}{0{,}317\text{years}}$	$\frac{2\,10^5}{0{,}317\text{years}}$	$\frac{8\,10^4}{3170\text{years}}$	$--$	$\frac{10^6}{16{,}7min}$	$\frac{10^5}{0{,}3s}$
10^5	$\frac{10^7}{2{,}78h}$	$\frac{5\,10^7}{2{,}78h}$	$\frac{5\,10^6}{31{,}7\text{years}}$	$\frac{2\,10^6}{31{,}7\text{years}}$	$\frac{8\,10^5}{3{,}17\,10^6\text{years}}$	$--$	$\frac{10^7}{2{,}78h}$	$\frac{10^6}{0{,}35s}$
10^6	$\frac{10^8}{1{,}16\text{days}}$	$\frac{5\,10^8}{1{,}16\text{days}}$	$\frac{5\,10^7}{3170\text{years}}$	$\frac{2\,10^7}{3170\text{years}}$	$\frac{8\,10^6}{3{,}17\,10^9\text{years}}$	$--$	$\frac{10^8}{1{,}16\text{days}}$	$\frac{10^7}{0{,}4s}$

5.— In order to obtain numerically the value of $\alpha = \sqrt{2}$ two iterative algorithms are proposed:

$$\textbf{1)}\ x_{n+1} = \frac{2 + x_n(10 - x_n)}{10}; \qquad \textbf{2)}\ x_{n+1} = \frac{x_n}{2} + \frac{1}{x_n}$$

For both of them:

- **a)** Analyze the evolution of the absolute truncation error between two consecutive iterations.
- **b)** Determine for which initial values x_0 does the algorithm converge and for which ones it does not. Provide examples.
- **c)** Simplificar el estudio anterior suponiendo que la aproximación inicla x_0 es "suficientemente buena". Obtener el orden de convergencia.
- **d)** Sabiendo que el valor $\sqrt{2}$ se encuentra comprendido entre 1.41 y 1.42, estimar cuantas iteraciones hay que realizar partiendo de la aproximación inicial x_0=1.415 para obtener una mejor apxomación con 5, 10, 15, 20 y 100 cifras significativas exactas. Realizar algunas iteraciones para verificar el resultado.

Solution 5.a)

1) $$x_{n+1} = \frac{2 + x_n(10 - x_n)}{10}$$

Let e_n be the error at iteration n, we can write:

$$\left.\begin{array}{l} e_n = \alpha - x_n \rightarrow x_n = \alpha - e_n \\ e_{n+1} = \alpha - x_{n+1} \rightarrow x_{n+1} = \alpha - e_{n+1} \end{array}\right\}$$

Taking it to the expression of the method:

$$\alpha - e_{n+1} = \frac{2 + (\alpha - e_n)(10 - (\alpha - e_n))}{10} = \frac{2 + 10(\alpha - e_n) - (\alpha - e_n)^2}{10} \Rightarrow$$

$$\Rightarrow \cancel{10\alpha} - 10\, e_{n+1} = 2 + \cancel{10\alpha} - 10\, e_n - \alpha^2 - e_n^2 + 2\,\alpha\, e_n$$

We also know that $\alpha^2 = 2$, thus:

$$e_{n+1} = \left(1 - \frac{2\alpha}{10} + \frac{e_n}{10}\right) e_n = \underbrace{\boxed{\left(1 - \frac{\sqrt{2}}{5}\right) e_n + \frac{1}{10} e_n^2}}_{\text{Asintotically linear}}$$

2) $$x_{n+1} = \frac{x_n}{2} + \frac{1}{x_n}$$

Using again the expressions:

$$\left.\begin{array}{l} e_n = \alpha - x_n \rightarrow x_n = \alpha - e_n \\ e_{n+1} = \alpha - x_{n+1} \rightarrow x_{n+1} = \alpha - e_{n+1} \end{array}\right\}$$

and taking them to the expression of the method:

$$\alpha - e_{n+1} = \frac{\alpha - e_n}{2} + \frac{1}{\alpha - e_n} \Rightarrow e_{n+1} = \alpha - \frac{\alpha}{2} + \frac{e_n}{2} - \frac{1}{\alpha - e_n} =$$

$$= \frac{\alpha + e_n}{2} - \frac{1}{\alpha - e_n} = \frac{(\alpha^2 - e_n^2) - 2}{2(\alpha - e_n)}$$

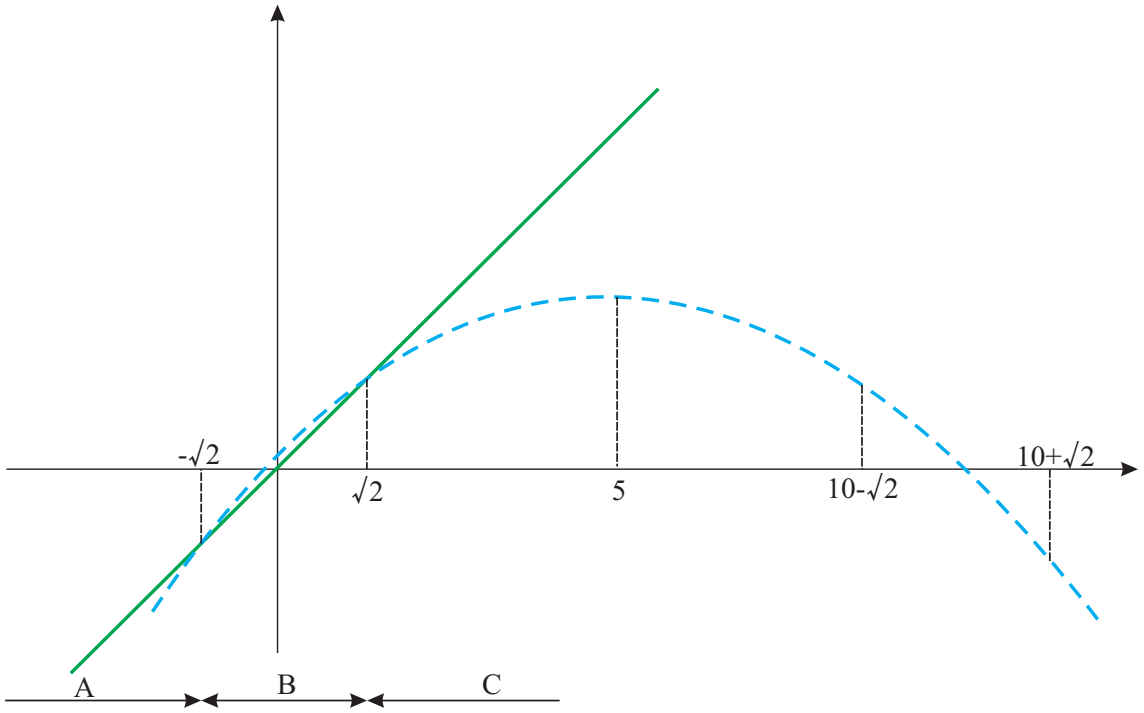
If we already know that $\alpha^2 = 2$:

$$e_{n+1} = -\frac{1}{2(\alpha - e_n)} e_n^2 = \underbrace{\boxed{-\frac{1}{2(\sqrt{2} - e_n)} e_n^2}}_{\text{Asintotically quadratic}}$$

Solution 5.b)

1) $x_{n+1} = F(x_n)$, $F(x) = \dfrac{2 + x(10 - x)}{10}$

Fixed points: $x = F(x) \Leftrightarrow 10x = 2 + 10x - x^2 \Leftrightarrow x = \pm\sqrt{2}$

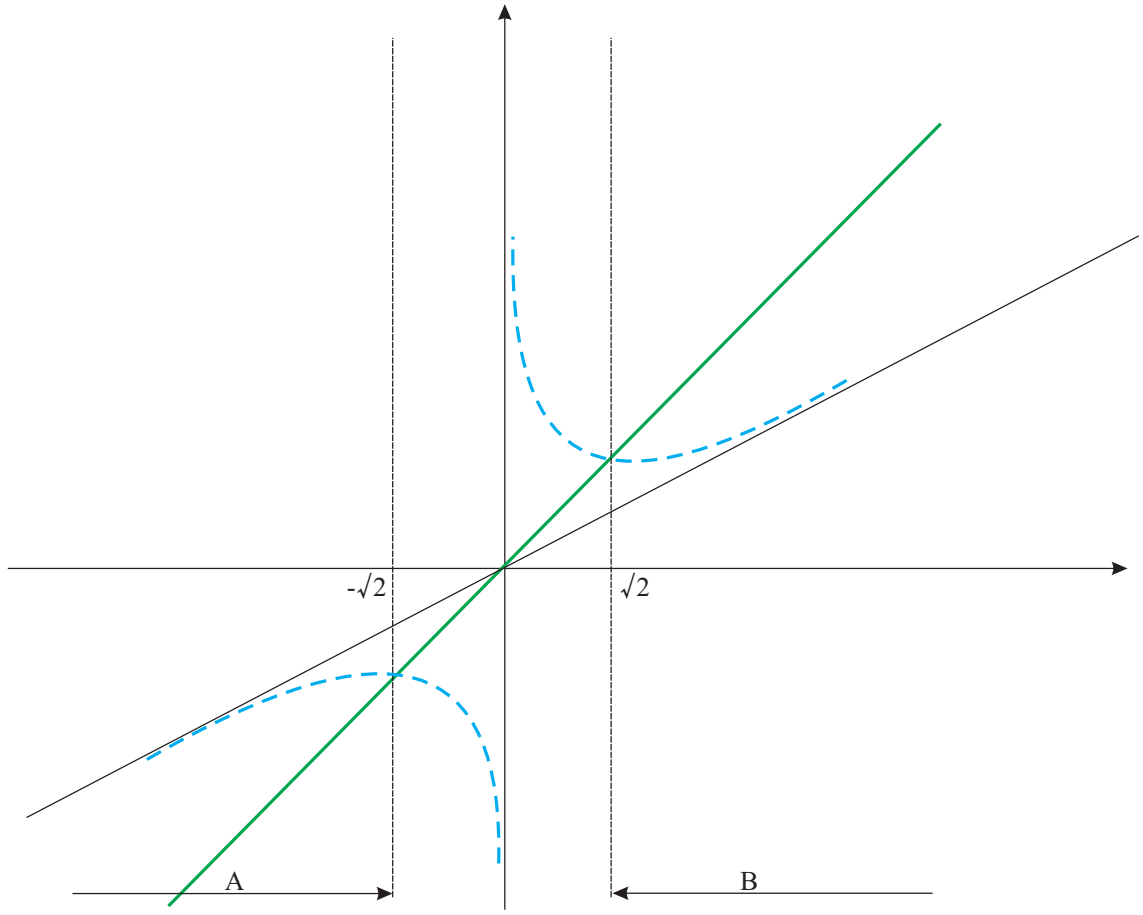


$$
\begin{cases}
\text{Area A} & : & x_0 < -\sqrt{2} & \rightarrow & \text{diverges to } -\infty \\
--- & : & x_0 = -\sqrt{2} & \rightarrow & x_n = -\sqrt{2} \quad \forall n \\
\text{Area B} & : & x_0 \in (-\sqrt{2}, \sqrt{2}) & \rightarrow & \text{converges to} + \sqrt{2} \text{ (increasing),F.A.C} = \left|1 - \frac{\sqrt{2}}{5}\right| \approx 0{,}717 \\
--- & : & x_0 = \sqrt{2} & \rightarrow & x_n = \sqrt{2} \quad \forall n \\
\text{Area C} & : & x_0 \in (\sqrt{2}, 5) & \rightarrow & \text{converges to} + \sqrt{2} \text{ (decreasing),F.A.C} = \left|1 - \frac{\sqrt{2}}{5}\right| \approx 0{,}717 \\
--- & : & x_0 \in [5, 10 - \sqrt{2}) & \rightarrow & x_1 \in \text{ Area C} \\
--- & : & x_0 = 10 - \sqrt{2} & \rightarrow & x_n = \sqrt{2} \quad \forall n \geq 1 \\
--- & : & x_0 \in (10 - \sqrt{2}, 10 + \sqrt{2}) & \rightarrow & x_1 \in \text{ Area B} \\
--- & : & x_0 = 10 + \sqrt{2} & \rightarrow & x_n = -\sqrt{2} \quad \forall n \geq 1 \\
--- & : & x_0 > 10 + \sqrt{2} & \rightarrow & x_1 \in \text{ Area A}
\end{cases}
$$

Thus, it converges to $\sqrt{2}$ for $x_0 \in (-\sqrt{2}, 10 + \sqrt{2})$

2) $\quad x_{n+1} = F(x_n) \quad , \quad F(x) = \frac{x}{2} + \frac{1}{x}$

Fixed points: $x = F(x) \Leftrightarrow x = \frac{x}{2} + \frac{1}{x} \Leftrightarrow x = \pm\sqrt{2}$



$$\begin{cases} \text{Area A} & : & x_0 < -\sqrt{2} & \rightarrow & \text{converges to } -\sqrt{2} \text{ (increasing), quadratic} \\ - - - & : & x_0 = -\sqrt{2} & \rightarrow & x_n = -\sqrt{2} \quad \forall n \\ - - - & : & x_0 \in (-\sqrt{2}, 0) & \rightarrow & x_1 \in \text{Area A} \\ - - - & : & x_0 = 0 & \rightarrow & x_1 \text{ undefined} \\ - - - & : & x_0 \in (0, \sqrt{2}) & \rightarrow & x_1 \in \text{Area B} \\ - - - & : & x_0 = \sqrt{2} & \rightarrow & x_n = \sqrt{2} \quad \forall n \\ \text{Area B} & : & x_0 > \sqrt{2} & \rightarrow & \text{converges to } \sqrt{2} \text{ (increasing), quadratic} \end{cases}$$

Thus, converges to $-\sqrt{2}$ for $x_0 \in (-\infty, 0)$ and to $\sqrt{2}$ for $x_0 \in (0, +\infty)$.

Solution 5.c) If the initial guess is "good enough':

$$|e_0| \leq \delta << 1 \qquad \Rightarrow \qquad |e_n| \leq \delta << 1 \quad \forall n$$

$$\textbf{1)} \quad e_{n+1} = \left(1 - \frac{\sqrt{2}}{5}\right) e_n + \frac{1}{10} e_n^2$$

$$|e_{n+1}| \leq \left[\left|1 - \frac{\sqrt{2}}{5}\right| + \frac{1}{10}|e_n|\right] |e_n|$$

$$|e_{n+1}| \leq \left[\left|1 - \frac{\sqrt{2}}{5}\right| + \delta\right] |e_n|$$

Since we know that $|e_n| = |x_n - \alpha|$:

$$|x_{n+1} - \alpha| \leq \underbrace{\left[\left|1 - \frac{\sqrt{2}}{5}\right| + \delta\right]}_{\lambda_1 \leq 0{,}717 + \delta < 1} |x_n - \alpha| \Rightarrow$$

$\Rightarrow$ Linear convergence (at least)

$$\textbf{2)} \quad e_{n+1} = -\frac{1}{2\left(\sqrt{2} - e_n\right)} e_n^2$$

$$|e_{n+1}| \leq \left|\frac{1}{2\left(\sqrt{2} - \delta\right)}\right| |e_n|^2$$

Applying again $|e_n| = |x_n - \alpha|$:

$$|x_{n+1} - \alpha| \leq \underbrace{\left|\frac{1}{2\left(\sqrt{2} - \delta\right)}\right|}_{\lambda_2} |x_n - \alpha|^2 \Rightarrow$$

$\Rightarrow$ quadratic convergence subject to $\lambda_2 |x_0 - \alpha|^{2-1} < 1 \Leftrightarrow |x_0 - \alpha| < 2(\sqrt{2} - \delta)$, which is verified.

Solution 5.d)

$$\left.\begin{array}{l}\alpha \in [1{,}41, 1{,}42] \\ x_0 = 1{,}415\end{array}\right\} \;\Rightarrow\; |x_0 - \alpha| \leq 0{,}005 \qquad \text{con } \alpha = \sqrt{2}$$

We want $x_n \approx \alpha$ with d exact digits, not including sign, thus:

$$\left|\frac{\alpha - x_n}{\alpha}\right| \leq \frac{1}{2}\, 10^{-(d+1)+2} = \frac{1}{2}\, 10^{-d+1}$$

$$\textbf{1)}\quad e_{n+1} \approx \left(1 - \frac{\sqrt{2}}{5}\right) e_n \;\Rightarrow\; e_n \approx \left(1 - \frac{\sqrt{2}}{5}\right)^n e_0$$

$$\Rightarrow\; |\alpha - x_n| \approx \left(1 - \frac{\sqrt{2}}{5}\right)^n |\alpha - x_0| \leq 0{,}005 \left(1 - \frac{\sqrt{2}}{5}\right)^n$$

$$\Rightarrow\; \left|\frac{\alpha - x_n}{\alpha}\right| \leq \frac{0{,}005}{\sqrt{2}} \left(1 - \frac{\sqrt{2}}{5}\right)^n$$

If $\dfrac{0{,}005}{\sqrt{2}} \left(1 - \dfrac{\sqrt{2}}{5}\right)^n \leq \dfrac{1}{2}\, 10^{-d+1}$ then $\left|\dfrac{\alpha - x_n}{\alpha}\right| \leq \dfrac{1}{2}\, 10^{-d+1}$

Thus:

$$\left(1 - \frac{\sqrt{2}}{5}\right)^n \leq \frac{\sqrt{2}}{0{,}005}\frac{1}{2}\, 10^{-d+1} \Leftrightarrow n \underbrace{\log_{10}\left(1 - \frac{\sqrt{2}}{5}\right)}_{<0} \leq \log_{10}\left(\frac{\sqrt{2}}{0{,}005}\frac{1}{2}\right) + (-d+1) \Leftrightarrow$$

$$\Leftrightarrow d - 1 - \log_{10}\left(\frac{\sqrt{2}}{0{,}005}\frac{1}{2}\right) \leq n \underbrace{\left[-\log_{10}\left(1 - \frac{\sqrt{2}}{5}\right)\right]}_{>0} \Leftrightarrow$$

$$\Leftrightarrow n \geq \frac{1}{-\log_{10}\left(1 - \frac{\sqrt{2}}{5}\right)}\, d + \frac{-1 - \log_{10}\left(\frac{\sqrt{2}}{0{,}005}\frac{1}{2}\right)}{-\log_{10}\left(1 - \frac{\sqrt{2}}{5}\right)} \approx 6{,}926\, d - 21{,}820 \Leftrightarrow$$

$$\Leftrightarrow n_1 \geq 6{,}926\, d - 21{,}820$$

$$\textbf{2)}\quad e_{n+1} \approx \left(-\frac{1}{2\sqrt{2}}\right) e_n^2 \;\Rightarrow\; e_n \approx -\left(\frac{1}{2\sqrt{2}}\right)^{1+2+2^2+\ldots+2^{n-1}} e_0^{2^n} = -\left(\frac{1}{2\sqrt{2}}\right)^{2^n-1} e_0^{(2^n)}$$

$$\Rightarrow\; |\alpha - x_n| \approx \left(\frac{1}{2\sqrt{2}}\right)^{2^n-1} 0{,}005^{(2^n)}$$

$$\Rightarrow\; \left|\frac{\alpha - x_n}{\alpha}\right| \leq \frac{2\cancel{\sqrt{2}}}{\cancel{\sqrt{2}}} \left(\frac{0{,}005}{2\sqrt{2}}\right)^{2^n}$$

If $2\left(\frac{0{,}005}{2\sqrt{2}}\right)^{2^n} \leq \frac{1}{2}\,10^{-d+1}$ then $\left|\frac{\alpha - x_n}{\alpha}\right| \leq \frac{1}{2}\,10^{-d+1}$

Thus:

$$\left(\frac{0{,}005}{2\sqrt{2}}\right)^{2^n} \leq \frac{1}{4}\,10^{-d+1} \Leftrightarrow 2^n \underbrace{\log_{10}\left(\frac{0{,}005}{2\sqrt{2}}\right)}_{<0} \leq \log_{10}\left(\frac{1}{4}\right) + (-d+1) \Leftrightarrow$$

$$\Leftrightarrow d - 1 - \log_{10}\left(\frac{1}{4}\right) \leq 2^n \underbrace{\left[-\log_{10}\left(\frac{0{,}005}{2\sqrt{2}}\right)\right]}_{>0} \Leftrightarrow$$

$$\Leftrightarrow 2^n \geq \frac{d}{-\log_{10}\left(\frac{0{,}005}{2\sqrt{2}}\right)} + \frac{-1 - \log_{10}\left(\frac{1}{4}\right)}{-\log_{10}\left(\frac{0{,}005}{2\sqrt{2}}\right)} \approx 0{,}363d - 0{,}145 \Leftrightarrow$$

$$\Leftrightarrow n_2 \geq \frac{\log_{10}(0{,}363d - 0{,}145)}{\log_{10}(2)}$$

The table shows an estimation of the number of iterations needed for each method to reach the desired significant digits:

d	n_1	n_2
5	13	1
10	48	2
15	83	3
20	117	3
100	671	6

Example:

i	$x_i(1)$	$x_i(2)$
0	1.415	1.415
1	1.4147775	1.41421378092
2	1.41461796255	1.41421356237
3	1.41450356455	1.41421356237
4	1.41442153114	
5	1.41436270436	
6	1.41432051841	
7	1.41429026553	
8	1.41426857001	
9	1.4142530112	
10	1.41424185323	
11	1.41423385129	
12	1.41422811268	
13	1.41422398721	
14	1.41422104578	
15	1.41421892915	

6.— An engineer working on road layout is creating a FORTRAN program. One part of the program has to print a list of the cut and fill areas along the road axis. By default the listing starts at kilometer point (PK) 0,000 and a data is printed every 0,1 Km. The program works apparently well, but users complain that sometimes strange values appear in the PK column, such as 70,999 instead of 71,000.

a) Make a FORTRAN program that repeatedly adds the value 0,1 in a variable of type `REAL*4` (initialized to zero) and prints the results. Verify that after a certain number of operations the printed result is not a multiple of $0,1$. Repeat the calculations with variables of type `REAL*8`, and check that the effect persists, although it takes longer to occur.

b) Check that the same effect occurs in a spreadsheet. To do this, start a column with the value zero in the first row and generate the values of each of the following rows by adding the value 0,1 to the result of the previous row.

c) Propose an alternative to solve the problem satisfactorily.

Solution 6.a) The program with variables `REAL*4` is:

```
c ========================================== Program Iterativesum_R4
c ============================================================
c This program sums iteratively in single precission the number a = 0.1
c until the rounding error affects the third decimal.
c ============================================================

      implicit real*4(a-h,o-z)
      character*20 spk
      logical next

      npk = 0
      dpk = 0.1
      xpk = 0.0
      next = .TRUE.

      do while(next)
          npk = npk + 1
          xpk = xpk + dpk
          write(spk,'(f20.3)') xpk
      write(6,100) xpk
          next = (spk(20:20).eq.'0')
      enddo

      write(6,101) npk,spk
  100 format('P.K. ',f20.3)
  101 format(' Problem in  term ',i20,'--->xpk =',a)

      end
```

The program with variables `REAL*8` is:

```
c ========================================= Programa Iterativesum_R8
c ============================================================
c This program sums iteratively in double precission the numbero a = 0.1
c until the rounding error affects the third decimal.
c ============================================================

      implicit real*8(a-h,o-z)
      character*20 spk
      logical next

      npk = 0
      dpk = 0.1d+00
      xpk = 0.0d+00
      next = .TRUE.

      do while(seguir)
          npk = npk + 1
          xpk = xpk + dpk
          write(spk,'(f20.3)') xpk
      write(6,100) xpk
          next = (spk(20:20).eq.'0')
      enddo

      write(6,100) npk,spk
  100 format('P.K. ',f20.3)
  101 format(' Problem in  term ',i20,'--->xpk =',a)

      end
```

We can see that the operation `xpk=xpk+dpk` creates accumulation of storing errors.

The same effect will remain if the write a loop in which the index variable is of type real.

```
do x = 0.,100.,0.1
  write(6,*) x
enddo
```

Solution 6.b) The effect also appears when solving the problem in a spreadsheet since the operations being performed are exactly the same.

Solution 6.b) To solve the problem it would only be necessary to rewrite the loop changing the sum operation by a product of two real numbers, which has less propagation of errors. It means, changing `xpk=xpk+dpk` by `xpk=0.1*FLOAT(npk)`