MÉTODOS NUMÉRICOS Y PROGRAMACIÓN 2023/2024

Almacenamiento de números y algoritmos (PRÁCTICA 1)

1.— Escribir en coma fija y en coma flotante, y en los sistemas de numeración denario, binario, octal y hexadecimal, los números:

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

Solución 1.a) Pasamos primero a base binaria para hacer más sencilla la conversión a los sistemas octal y hexadecimal. Se cambian de base por separado la parte decimal y la parte entera. Para hallar la parte entera:

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

Para pasar a base octal cogemos las cifras de 3 en 3 ($8{=}2^3$) y para pasar a base hexadecimal de 4 en 4 ($16{=}2^4$), es decir:

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

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

Calculamos ahora la parte decimal:

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

Procedemos de la misma forma que antes para pasar a base octal y hexadecimal:

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

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

Por tanto:

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

Solución 1.b)

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

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

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

Luego:

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

Solución 1.c)

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

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

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

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

Luego:

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

Solución 1.d)

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

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

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

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

La conversión de la parte decimal se muestra en la tabla 1.

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

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

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

Por tanto:

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

Solución 1.e)

$$e = 2{,}718281828459045$$

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

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

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

La conversión de la parte decimal se muestra en la tabla 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: Conversión parte decimal número 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: Conversión parte decimal número 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

Por tanto:

$$\begin{array}{|c|c|c|}\hline 2{,}718281828459045... & = & (0{,}2718281828459045...)\ 10^1 \\ (10{,}10110111111000010101000101100010 1...)_2 & = & \\ = (0{,}10101101111110000101010001011000101...)_2\ 2^2 & & \\ (2{,}55760521305...)_8 & = & (0{,}255760521305...)_8\ 8^1 \\ (2.B7E15162...)_{16} & = & (0{,}2B7E15162...)_{16}\ 16^1 \\ \hline \end{array}$$

Solución 1.f) En este caso es menos farragoso convertir el número a base hexadecimal, y a partir de ahí a base octal y base binaria. El método es idéntico:

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

Pasamos ahora a base binaria y octal:

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

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

Luego:

$$\begin{array}{|c|c|c|}\hline 52745916 & = & (0{,}52745916)\ 10^8 \\ (11001001001101011010111100)_2 & = & \\ = (0{,}11001001001101011010111100)_2\ 2^{26} & & \\ (311153274)_8 & = & (0{,}311153274)_8\ 8^9 \\ (324D6BC)_{16} & = & (0{,}324D6BC)_{16}\ 16^7 \\ \hline \end{array}$$

2.— Se pretende almacenar los valores anteriores en las variables `A, UD, UT, PI, E` y `GR` del programa Fortran:

```
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.
...
```

Si para la mantisa de las variables `REAL*4` se destinan 3 bytes (incluido el signo), se pide:

a) Calcular los valores que efectivamente se almacenan en cada caso.

b) Comparar los valores almacenados con los correspondientes valores exactos, indicando las fuentes de error.

c) Comparar el error de almacenamiento que se produce en cada caso con su cota superior.

d) Estimar con cuántas cifras significativas deberían haberse escrito los números $1/3$, π y e. Repetir la estimación para el caso en que las variables sean del tipo `REAL*8`.

Solución 2.a) Convertimos las constantes a binario, en este caso con las cifras decimales que nos da el enunciado. Las expresamos en coma flotante y agrupamos las cifras en grupos de 1 byte (8 cifras):

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

Suponemos que se almacenan 23 bits de la mantisa, 1 bit se destina al signo y se redondea por aproximación. De este modo los valores almacenados realmente serán:

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

NOTA: En realidad el primer bit que siempre corresponde a un 1 no se almacena pero no tendremos en cuenta este efecto por simplificar. La parte subrayada es la parte afectada por el redondeo.

Si ahora volvemos a pasar estos números a la base decimal tendremos las constantes que se almacenan realmente:

`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.

Solución 2.b)

* 5.3125 se almacena exactamente en `A` dado que su mantisa ocupa menos de 23 bits.
* 0.1 y 52745916 se almacenan en `UD` y `GR` respectivamente con un error de redondeo que se debe a que sus mantisas tienen más de 23 cifras significativas. Al redondear a 23 bits se pierde información.

* 0.333333, 3.14159 y 2.71828 se almacenan en `UT, PI` y `E` respectivamente con un error de redondeo que se debe a que sus mantisas tienen más de 23 cifras significativas, y al redondear de nuevo se pierde información. Además estos datos tienen errores inherentes, ya que se supone que pretendemos utilizar los números $1/3$, π y e.

Solución 2.c) Sea $\overline{x}$ el número real que queremos representar, $\widehat{x}$ el número que almacenamos y x la variable realmente almacenada ($\overline{x} = \pi, \hat{x} = 3{,}14159, x = 3{,}14159011841$). Definimos:

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

El error de almacenamiento está acotado por el error de máquina:

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

Errores de redondeo:

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

Errores inherentes:

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

Errores totales:

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

Solución 2.d) Observamos que los errores inherentes de `UT, PI` y `E` son excesivos. Es decir, observamos que $1/3$, π y e se han escrito con errores inherentes elevados en comparación con el error de máquina r_M.

Calculamos el número de cifras significativas s que tendríamos que haber usado en base 10 para que el error fuera del orden de magnitud del de máquina:

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

Luego tendríamos que haber escrito:

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

para que los errores inherentes fuesen del orden de magnitud de los de almacenamiento.

En el caso de utilizar variables tipo `REAl*8`: $r_M \approx (1/2)\ 2^{-53+2}$

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

y tendríamos que haber escrito:

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

para que los errores inherentes fuesen del orden de magnitud de los de almacenamiento.

3.— Comparar los valores almacenados en las variables `GDIF` y `NGDIF` en el siguiente programa Fortran:

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

Si para la mantisa de las variables `REAL*4` se destinan 3 bytes (incluido el signo), ¿se detendrá el programa al ejecutar la instrucción `IF`? ¿Qué conclusiones pueden extraerse de este ejemplo?

Solución 3. Para todos los números almacenados en coma flotante excepto para el 0, el primer bit después de la coma siempre irá ocupado por un 1, por tanto no es necesario almacenarlo, es decir es como si tuviéramos un bit más $\Rightarrow$ m=25.

Teniendo en cuenta esto, el número 52745916. se almacena como:

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

y el número 15. se almacena como:

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

Así, la instrucción `GDIF=52745916.+15.` dará como resultado:

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

que se almacenará como:

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

Posteriormente, la instrucción `GDIF=GDIF-52745916.` dará como resultado:

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

que se almacena como:

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

Sin embargo, las operaciones con enteros serán más precisas porque caben números hasta 31 dígitos binarios.

Por tanto el programa no se detendrá pues `GDIF` contendrá el valor 16. y `FLOAT(NGDIF)` dará el valor 15.

Conclusión: En un ordenador no es lo mismo operar con variables de tipo enteras que con variables de tipo real.

4.— Cierto ordenador tarda una décima de segundo (como máximo) en:

- **a)** Calcular, mediante las ecuaciones normales de mínimos cuadrados, la recta de regresión que mejor aproxima 100 puntos dato.
- **b)** Calcular el módulo de un vector de 400 componentes.
- **c)** Ordenar de mayor a menor una lista de 50 números mediante el método de la burbuja (*"bubble sort"*)
- **d)** Calcular el producto de una matriz cuadrada de orden 20 por un vector.
- **e)** Multiplicar dos matrices cuadradas de orden 8.
- **f)** Calcular (aplicando su definición algebraica) el determinante de una matriz cuadrada de orden 5.

g) Buscar por inspección secuencial un número de teléfono en un listín telefónico (ordenado alfabéticamente) donde figuran 100 abonados.

h) Buscar por bisección un número de teléfono en un listín telefónico (ordenado alfabéticamente) donde figuran 100 abonados.

¿Cuál será el tiempo de cálculo si el tamaño del problema (orden de la matriz o número de datos, según el caso) es diez, cien mil, diez mil, cien mil o un millón de veces mayor?.

Solución 4. Tenemos que calcular el coste computacional de cada uno de los apartados para analizar su dependencia con el tamaño del problema.

Solución 4.a) Dados $\{x_k, y_k\}_{k=1,\dots,m}$, queremos aproximar $y(x)$ por un polinomio de orden 1, $P_1(x) = ax + b$, de forma que para $Q = \sum_{k=1}^{m} (y_k - P_1(x_k))^2$ se cumpla que $\frac{\partial Q}{\partial a} = 0$ y $\frac{\partial Q}{\partial b} = 0$.

Es decir:

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

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

Luego:

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

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

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

Si realizamos las operaciones de la siguiente forma:

$$\begin{cases} \mu_x = \left(\sum_{k=1}^{m} x_k \right) /m & \rightarrow \ (m-1) \text{ sumas}, 1 \text{ división} \\ \mu_y = \left(\sum_{k=1}^{m} y_k \right) /m & \rightarrow \ (m-1) \text{ sumas}, 1 \text{ división} \\ \sigma_x^2 = \left(\sum_{k=1}^{m} (x_k - \mu_x)^2 \right) /m & \rightarrow \ (m) \text{ restas}, (m) \text{ productos}, (m-1) \text{ sumas}, 1 \text{ división} \\ \sigma_{xy} = \left(\sum_{k=1}^{m} (x_k - \mu_x)(y_k - \mu_y) \right) /m & \rightarrow \ (2m) \text{ restas}, (m) \text{ productos}, (m-1) \text{ sumas}, 1 \text{ división} \\ \hline P_1(x) = \mu_y + \dfrac{\sigma_{xy}}{\sigma_x^2}(x - \mu_x) & \text{TOTAL} = (9m) \text{ operaciones} \end{cases}$$

Por tanto, el tiempo de cálculo es proporcional a $\boxed{T(m)}$

Solución 4.b) Dado un vector $\boldsymbol{v} = \{v_i\}_{i=1,\ldots,m}$:

$$|\boldsymbol{v}| = \left(\sum_{i=1}^{m} v_i^2 \right)^{1/2} \Rightarrow \left\{ \begin{array}{l} m \text{ productos} \\ (m-1) \text{ sumas} \\ 1 \text{ raíz cuadrada} \end{array} \right\} \Rightarrow \text{ TOTAL} = 2m \text{ operaciones} \Rightarrow \boxed{T(m)}$$

Solución 4.c)

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

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

No todas las operaciones requieren un cambio ya que dependen del orden inicial de los datos. Si ya están ordenados requieren 0 cambios. Si todos los cambios fueran necesarios requerirán $\dfrac{m(m-1)}{2}$ cambios.

En todo caso, el tiempo de cálculo será proporcional a $\frac{m(m-1)}{2}$ de forma general, es decir $\boxed{T(m^2)}$.

Solución 4.d)

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

$$\text{TOTAL} = m(2m-1) \text{ operaciones} \quad \Rightarrow \quad \boxed{T(m^2)}$$

Solución 4.e)

$$\boldsymbol{C} = \boldsymbol{A}\boldsymbol{B} \;\Rightarrow\quad c_{i,j} = \sum_{k=1}^{m} a_{i,k} b_{k,j} \quad ; \quad \begin{array}{l} i = 1, \ldots, m \\ j = 1, \ldots, m \end{array} \quad \Rightarrow \quad m^2 \text{ veces} \left\{ \begin{array}{l} m \text{ productos} \\ (m-1) \text{ sumas} \end{array} \right.$$

$$\text{TOTAL} = m^2(2m-1) \text{ operaciones} \quad \Rightarrow \quad \boxed{T(m^3)}$$

Solución 4.f)

$$det(\boldsymbol{A}) \;=\; \sum_{j=1}^{m} a_{i,k} \boldsymbol{A}_{k,j} \quad \Rightarrow \quad \begin{array}{c} 1 \text{ determinante} \\ \text{de orden m} \end{array} \Rightarrow \left\{ \begin{array}{l} m \text{ productos} \\ (m-1) \text{ sumas} \\ m \text{ determinantes de orden (m-1)} \end{array} \right.$$

Sea C_m el número de operaciones que hay que realizar para calcular un determinante de orden m. Dado lo anterior sabemos que:

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

Por tanto:

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

Por tanto, el cálculo del determinante de una matriz de orden m tiene un coste de $(-1 + e\, m!)$ operaciones, es decir $\boxed{T(m!)}$

Solución 4.g) Definimos:

`A(M)` como el vector que contiene los apellidos.

`T(M)` como el vector que contiene los teléfonos correspondientes.

`AA` como el apellido cuyo teléfono buscamos.

De esta forma:

```
DO I = 1, M
  IF(A(I).EQ.AA)THEN
    WRITE(6,*)'EL TELÉFONO ES', T(I)
    STOP
  ENDIF
ENDDO
WRITE(6,*)'TELÉFONO INEXISTENTE'
```

El número de comparaciones depende de la posición del número que buscamos en el archivo. En el mejor de los casos sólo se necesitará una comparacion. En el peor de los casos m comparaciones. En el caso promedio $m/2$ comparaciones.

Por tanto el coste será $\boxed{T(m)}$

Solución 4.h) El programa será el siguiente:

```
IF(A(1).EQ.AA)THEN
    WRITE(6,*)'EL TELÉFONO ES', T(1)
    STOP
ELSE IF(A(M).EQ.AA)THEN
    WRITE(6,*)'EL TELÉFONO ES', 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,*)'EL TELÉFONO ES', T(IMED)
            STOP
        ELSE IF(A(IMED).LT.AA)THEN
            IMIN = IMED
        ELSE
            IMAX = IMED
        ENDIF
    ENDDO
ENDIF
WRITE(6,*)'TELÉFONO INEXISTENTE'
```

El número de comparaciones que hay que realizar será como máximo aproximadamente $\log_2 m$, por lo que el coste será $\boxed{T(\log_2 m)}$

Por tanto, con los costes computacionales calculados y los datos del enunciado:

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

Luego si el problema es n veces mayor:

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

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

5.— Para calcular numéricamente el valor $\alpha = \sqrt{2}$ se plantean los dos algoritmos iterativos siguientes:

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

Para ambos, se pide:

- **a)** Analizar la evolución del error absoluto de truncamiento entre dos iteraciones sucesivas.
- **b)** Estudiar para qué valores iniciales x_0 converge el algoritmo, y para qué valores no converge. Dar algunos ejemplos.
- **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.

Solución 5.a)

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

Llamamos e_n al error en la iteración n, de esta forma podemos escribir:

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

Lo llevamos a la expresión del método:

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

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

Sabemos además que $\alpha^2 = 2$, por tanto:

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

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

Utilizamos de nuevo las expresiones:

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

y lo llevamos a la expresión del método:

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

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

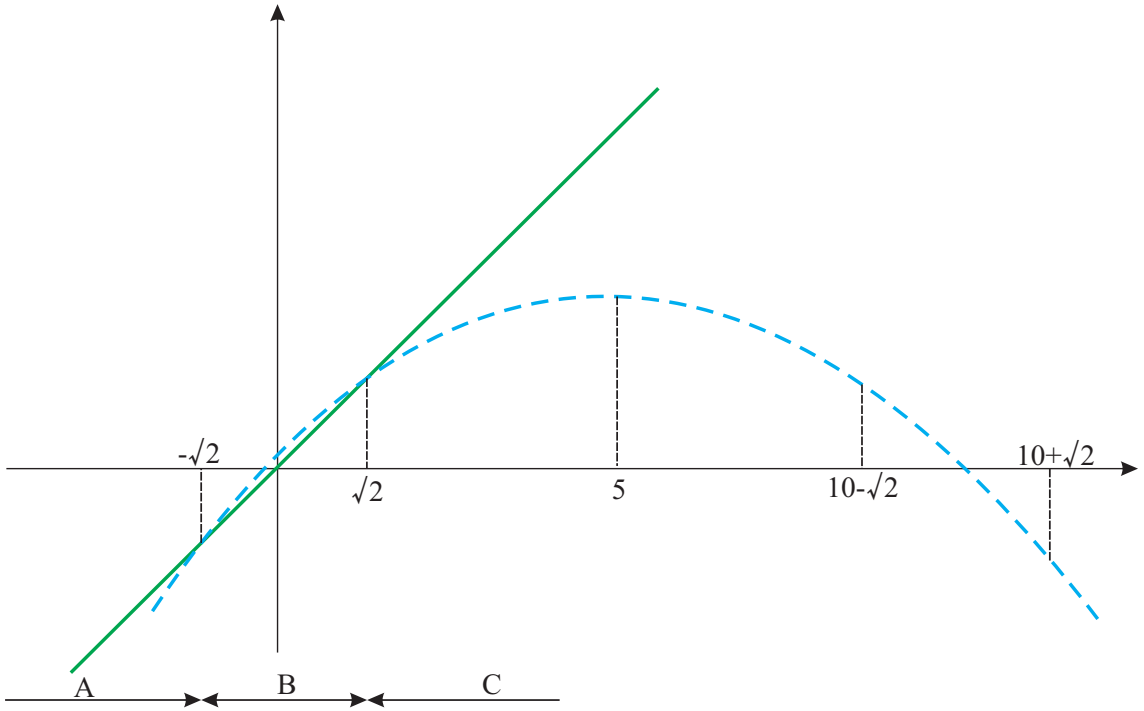
Sabemos además que $\alpha^2 = 2$, por tanto:

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

Solución 5.b)

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

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

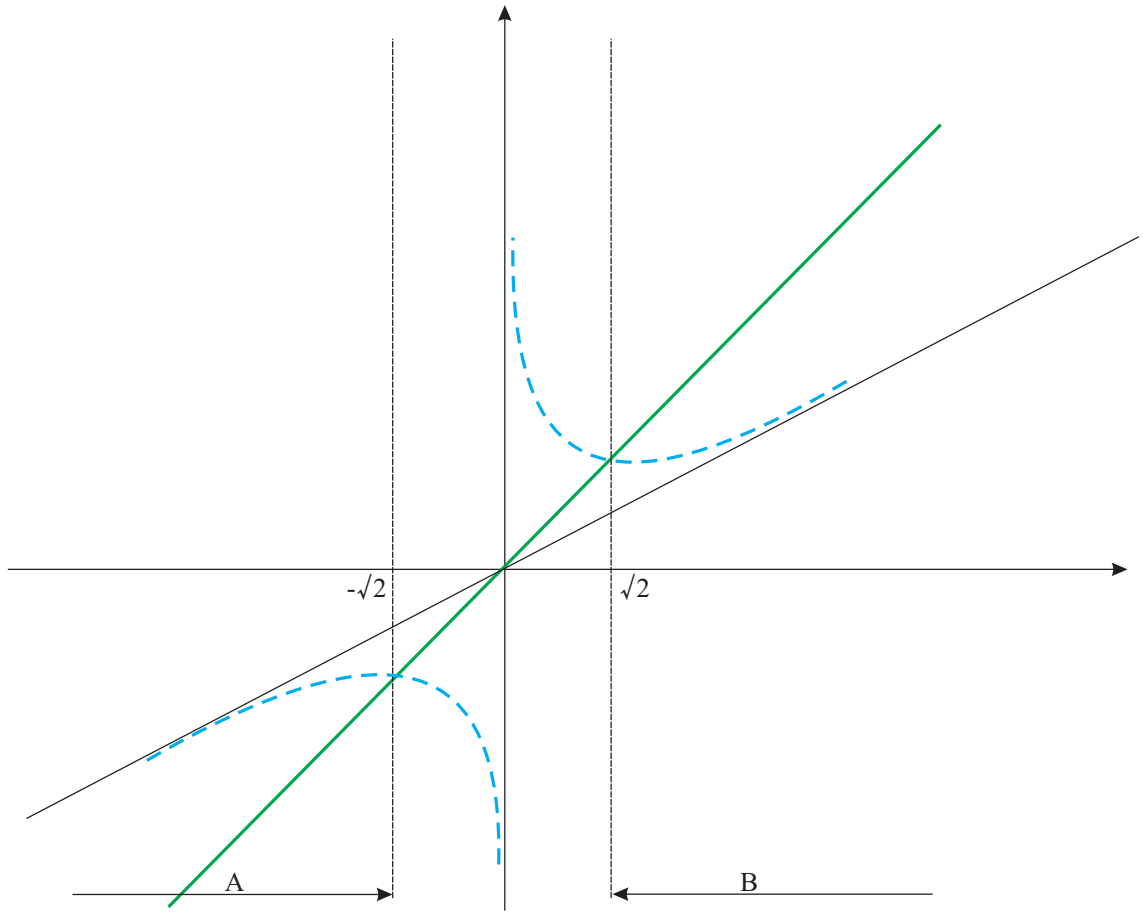


$$
\begin{cases}
\text{Zona A} : x_0 < -\sqrt{2} \rightarrow \text{diverge a } -\infty \\
--- : x_0 = -\sqrt{2} \rightarrow x_n = -\sqrt{2} \quad \forall n \\
\text{Zona B} : x_0 \in (-\sqrt{2}, \sqrt{2}) \rightarrow \text{converge a } +\sqrt{2} \text{ (creciente),F.A.C} = \left|1 - \frac{\sqrt{2}}{5}\right| \approx 0{,}717 \\
--- : x_0 = \sqrt{2} \rightarrow x_n = \sqrt{2} \quad \forall n \\
\text{Zona C} : x_0 \in (\sqrt{2}, 5) \rightarrow \text{converge a } +\sqrt{2} \text{ (decreciente),F.A.C} = \left|1 - \frac{\sqrt{2}}{5}\right| \approx 0{,}717 \\
--- : x_0 \in [5, 10 - \sqrt{2}) \rightarrow x_1 \in \text{ Zona C} \\
--- : x_0 = 10 - \sqrt{2} \rightarrow x_n = \sqrt{2} \quad \forall n \geq 1 \\
--- : x_0 \in (10 - \sqrt{2}, 10 + \sqrt{2}) \rightarrow x_1 \in \text{ Zona B} \\
--- : x_0 = 10 + \sqrt{2} \rightarrow x_n = -\sqrt{2} \quad \forall n \geq 1 \\
--- : x_0 > 10 + \sqrt{2} \rightarrow x_1 \in \text{ Zona A}
\end{cases}
$$

Por tanto, converge a $\sqrt{2}$ para $x_0 \in (-\sqrt{2}, 10 + \sqrt{2})$

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

Puntos fijos: $x = F(x) \Leftrightarrow x = \dfrac{x}{2} + \dfrac{1}{x} \Leftrightarrow x = \pm\sqrt{2}$



$$\begin{cases} \text{Zona A} & : & x_0 < -\sqrt{2} & \rightarrow & \text{converge a } -\sqrt{2} \text{ (creciente), Cuadráticamente} \\ --- & : & x_0 = -\sqrt{2} & \rightarrow & x_n = -\sqrt{2} \quad \forall n \\ --- & : & x_0 \in (-\sqrt{2}, 0) & \rightarrow & x_1 \in \text{Zona A} \\ --- & : & x_0 = 0 & \rightarrow & x_1 \text{ indefinido} \\ --- & : & x_0 \in (0, \sqrt{2}) & \rightarrow & x_1 \in \text{Zona B} \\ --- & : & x_0 = \sqrt{2} & \rightarrow & x_n = \sqrt{2} \quad \forall n \\ \text{Zona B} & : & x_0 > \sqrt{2} & \rightarrow & \text{converge a } \sqrt{2} \text{ (decreciente), Cuadráticamente} \end{cases}$$

Por tanto, converge a $-\sqrt{2}$ para $x_0 \in (-\infty, 0)$ y a $\sqrt{2}$ para $x_0 \in (0, +\infty)$.

Solución 5.c) Si la aproximación inicial es "suficientemente buena':

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

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

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

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

Como sabemos que $|e_n| = |x_n - \alpha|$:

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

$\Rightarrow$ Convergencia lineal (asegurada)

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

Aplicamos de nuevo $|e_n| = |x_n - \alpha|$:

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

$\Rightarrow$ Convergencia cuadrática con la condición $\lambda_2 |x_0-\alpha|^{2-1} < 1 \Leftrightarrow |x_0-\alpha| < 2(\sqrt{2}-\delta)$, que se cumple con seguridad.

Solución 5.d)

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

Queremos que $x_n \approx \alpha$ con d cifras exactas sin contar el signo, lo que implica:

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

1) $$e_{n+1} \approx \left(1-\frac{\sqrt{2}}{5}\right) e_n \Rightarrow e_n \approx \left(1-\frac{\sqrt{2}}{5}\right)^n e_0$$
$$\Rightarrow |\alpha - x_n| \approx \left(1-\frac{\sqrt{2}}{5}\right)^n |\alpha - x_0| \leq 0{,}005 \left(1-\frac{\sqrt{2}}{5}\right)^n$$
$$\Rightarrow \left|\frac{\alpha - x_n}{\alpha}\right| \leq \frac{0{,}005}{\sqrt{2}} \left(1-\frac{\sqrt{2}}{5}\right)^n$$

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

Luego:

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

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

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

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

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

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

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

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

Luego:

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

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

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

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

En la tabla se muestra el número estimado de iteraciones necesarias en cada método para conseguir las cifras indicadas:

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

Ejemplo:

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.— Un ingeniero que trabaja en trazado de carreteras realiza un programa Fortran. Una parte del programa tiene que imprimir un listado de las áreas de desmonte y terraplén a lo largo del eje de la carretera. Por defecto el listado comienza en el punto kilométrico (PK) 0.000 y se imprime un dato cada 0.1 Km. El programa funciona aparentemente bien, pero quienes lo utilizan se quejan de que en la columna correspondiente al PK aparecen en ocasiones valores extraños, como 70.999 en lugar de 71.000.

Se pide:

a) Realizar un programa Fortran que sume repetidamente el valor 0.1 en una variable de tipo `REAL*4` (inicializada a 0) e imprima los resultados. Comprobar que tras un cierto número de operaciones el resultado que se imprime no es múltiplo de 0.1. Repetir los cálculos con variables de tipo `REAL*8`, y comprobar que el efecto persiste, aunque tarda más en producirse.

b) Comprobar que el mismo efecto se produce en una hoja de cálculo. Para ello, se iniciará una columna con el valor cero en la primera fila y se generarán los valores de cada una de las filas siguientes sumando el valor 0.1 al resultado de la fila anterior.

c) Proponer una alternativa que permita resolver el problema satisfactoriamente.

Solución 6.a) El programa con las variables tipo `REAL*4` sería el siguiente:

```
c ========================================= Programa SumaIterada_R4
c ============================================================
c Este programa suma iterativamente en simple precision el numero a = 0.1
c hasta que el error de redondeo afecta al tercer decimal.
c ============================================================

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

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

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

      write(6,101) npk,spk
  100 format('P.K. ',f20.3)
  101 format(' Problema en el termino ',i20,'--->xpk =',a)

      end
```

El programa con las variables tipo `REAL*8` sería el siguiente:

```
c ========================================= Programa SumaIterada_R8
c ============================================================
c Este programa suma iterativamente en doble precision el numero a = 0.1
c hasta que el error de redondeo afecta al tercer decimal.
c ============================================================

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

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

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

      write(6,100) npk,spk
  100 format('P.K. ',f20.3)
  101 format(' Problema en el termino ',i20,'--->xpk =',a)

      end
```

Se puede comprobar fácilmente que la operación `xpk=xpk+dpk` produce la acumulación de errores de almacenamiento.

El mismo efecto se produciría si hacemos un bucle en el que la variable contador sea de tipo real.

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

Solución 6.b) El efecto también es el mismo al resolver el problema en una hoja de cálculo debido a que las operaciones que se están realizando son las mismas.

Solución 6.b) Para solventar el problema bastaría con reescribir el bucle sustituyendo la operación suma por una operación de multiplicación de números reales, que tiene menor propagación de errores. Es decir sustituir la operación `xpk=xpk+dpk` por `xpk=0.1*FLOAT(npk)`