INTRODUCCIÓN A LOS MÉTODOS NUMÉRICOS

PRÁCTICA 1 **Curso 2025-2026**

1. Realizar un programa Fortran que sume repetidamente el valor 0.1 en una variable de tipo REAL*4 (inicializada a cero) e imprima los resultados.

```
implicit integer*4(i-n)
implicit real*4(a-h,o-z)

x=0.0
do i=1,1000
  x=x+0.1
  write(6,*)x
enddo

end
```

   a) 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 repetirse.

   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

```
implicit integer*4(i-n)
implicit real*4(a-h,o-z)

x=0.0
do i=1,1000
  x=0.1*float(i)
  write(6,*)x
enddo

end
```

2. Escribir en coma fija y en coma flotante en los sistemas de numeración denario (base decimal) y binario los siguientes números:

   a) 5.125

   b) 0.2

   c) π

   d) e

   e) 30566232.

Base decimal, (coma fija a la izquierda, coma flotante a la derecha) redondeo por aproximación

Decimal	Coma fija	Coma flotante
5,125	5,125	$0{,}51250000\ 10^{1}$
0,2	0,2	$0{,}20000000\ 10^{0}$
π	3,1415927	$0{,}31415927\ 10^{1}$
e	2,7182818	$0{,}27182818\ 10^{1}$
3056232	3056232	$0{,}30562320\ 10^{7}$

(1)

Base binaria, (coma fija a la izquierda, coma flotante a la derecha) redondeo por aproximación

Binario	Coma fija	Coma flotante
5,125	$(101{,}001)_2$	$(0{,}101001)_2\ 2^{(11)_2}$
0,2	$(0.\widehat{0011})_2$	$(0.\widehat{1100})_2\ 2^{-(10)_2}$
π	$(11{,}001001000)_2$	$(0{,}11001001000)_2\ 2^{(10)_2}$
e	$(10{,}101110000)_2$	$(0{,}10101110000)_2\ 2^{(10)_2}$
3056232	$(1011101010001001101000)_2$	$(0{,}1011101010001001101000)_2\ 2^{(10110)_2}$

(2)

3. Si se almacenan los valores del ejercicio anterior en variables de tipo REAL*4 con 3 bytes para la mantisa (incluido el signo) en un programa Fortran como:

```
REAL*4 A,B,PI,E,G

A=5.125
B=0.2
PI=3.1416
E=2.718
G=30566232.
...
```

Se pide:

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

Binario	Coma flotante (24 dígitos de mantisa)
5,125	$+(0{,}101001000000000000000000)_2\ 2^{(11)_2}$
0,2	$+(0{,}110011001100110011001100)_2\ 2^{-(10)_2}$
π	$+(0{,}110010010000111111101101)_2\ 2^{(10)_2}$
e	$+(0{,}101011011111100001010101)_2\ 2^{(10)_2}$
3056232	$+(0{,}101110101000100110100000)_2\ 2^{(10110)_2}$

(3)

b) Comparar los valores almacenados con los correspondientes valores exactos (o al menos con valores mucho más precisos), indicando las fuentes de error en cada caso.

Binario	Coma flotante
5,125	$+(0{,}10100100000000000000000)_2\ 2^{(11)_2}$
exacto	$+(0{,}10100100000000000000000...)_2\ 2^{(11)_2}$
dif	$+(0{,}00000000000000000000000)_2\ 2^{(11)_2}$
0,2	$+(0{,}11001100110011001100110)_2\ 2^{-(10)_2}$
exacto	$+(0{,}11001100110011001100110011...)_2\ 2^{-(10)_2}$
dif	$+(0{,}00000000000000000000000110011...)_2\ 2^{-(10)_2}$
π	$+(0{,}11001001000011111101101)_2\ 2^{(10)_2}$
exacto	$+(0{,}11001001000011111101101010100 1...)_2\ 2^{(10)_2}$
dif	$+(0{,}00000000000000000000000101001...)_2\ 2^{(10)_2}$
e	$+(0{,}10101101111110000101010)_2\ 2^{(10)_2}$
exacto	$+(0{,}10101101111110000101010001011 0...)_2\ 2^{(10)_2}$
dif	$+(0{,}00000000000000000000000010110...)_2\ 2^{(10)_2}$
3056232	$+(0{,}10111010100010011010000)_2\ 2^{(10110)_2}$
exacto	$+(0{,}10111010100010011010000...)_2\ 2^{(10110)_2}$
dif	$+(0{,}00000000000000000000000...)_2\ 2^{(10110)_2}$

(4)

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

El error de máquina en este caso sería $r_M \approx 1{,}192\ 10^{-7}$

Binario	error absoluto	error relativo
5,125	$+(0{,}0)_2$	0
0,2	$+(0{,}110011...)_2\ 2^{-(10)_2}\ 2^{-(11000)_2}$	$5{,}935...\ 10^{-8}$
π	$+(0{,}101001...)_2\ 2^{(10)_2}\ 2^{-(11000)_2}$	$4{,}861...\ 10^{-8}$
e	$+(0{,}10110...)_2\ 2^{(10)_2}\ 2^{-(11001)_2}$	$3{,}015...\ 10^{-8}$
3056232	$+(0{,}0)_2$	0

(5)

d) Estimar con cuantas cifras significativas deberían haberse indicado los números π y e.

Deberían haberse indicado con al menos 8 cifras significativas.

4. Considérese el vector columna $\boldsymbol{v} \in \mathbb{R}^n$ donde $v_i = i$, la matriz $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ donde $a_{ij} = i + j$ y la matriz $\boldsymbol{B} \in \mathbb{R}^{n \times n}$ donde $b_{ij} = i - j$. Escribir un algoritmo en Fortran para realizar las siguientes operaciones:

   a) Calcular el módulo de $\boldsymbol{v}$.

$$vmodulo = \sqrt{\sum_{i=1}^{n} [v(i)]^2} \tag{6}$$

```
vmodulo=0.d+00
do i=1,n
  vmodulo=vmodulo+dble(i)*dble(i)
enddo
if(vmodulo.ne.(0.d+00)) then
  vmodulo=sqrt(vmodulo)
endif
```

   b) Calcular el producto $\boldsymbol{Av}$.

$$p(i) = \sum_{j=1}^{n} a(i,j)\ v(j) \qquad i = 1, \ldots, n \tag{7}$$

```
do i=1,n
  p(i)=0.d+00
  do j=1,n
    p(i)=p(i)+dble(i+j)*dble(j)
  enddo
enddo
```

   c) Calcular la suma $\boldsymbol{A} + \boldsymbol{B}$.

$$s(i,j) = a(i,j) + b(i,j) \qquad i = 1, \ldots, n,\ j = 1, \ldots, n \tag{8}$$

```
do i=1,n
  do j=1,n
    s(i,j)=dble(i+j)+dble(i-j)
  enddo
enddo
```

   d) Calcular el producto $\boldsymbol{AB}$

$$pm(i,j) = \sum_{k=1}^{n} a(i,k)\ b(k,j) \qquad i = 1, \ldots, n,\ j = 1, \ldots, n \tag{9}$$

```
do i=1,n
  do j=1,n
    pm(i,j)=0.d+00
    do k=1,n
      pm(i,j)=pm(i,j)+dble(i+k)*dble(k-j)
    enddo
  enddo
enddo
```

e) Calcular el producto $\boldsymbol{AB}^T$

$$pm(i,j) = \sum_{k=1}^{n} a(i,k)\ b(j,k) \qquad i = 1,\ldots,n,\ j = 1,\ldots,n \tag{10}$$

```
do i=1,n
  do j=1,n
    pm(i,j)=0.d+00
    do k=1,n
      pm(i,j)=pm(i,j)+dble(i+k)*dble(j-k)
    enddo
  enddo
enddo
```

f) Calcular el valor de $\boldsymbol{v}^T\boldsymbol{Av}$

$$\boldsymbol{v}^T\boldsymbol{Av} = \sum_{i=1}^{n} v(i) \sum_{j=1}^{n} a(i,j)v(j) \tag{11}$$

```
vtav=0.d+00
do i=1,n
  p=0.d+00
  do j=1,n
    p=p+dble(i+j)*dble(j)
  enddo
  vtav=vtav+dble(i)*p
enddo
```

g) Calcular el valor de $\boldsymbol{v}^T\boldsymbol{ABv}$

$$\boldsymbol{v}^T\boldsymbol{Av} = \sum_{i=1}^{n} v(i) \sum_{j=1}^{n} \left(\sum_{k=1}^{n} a(i,k)\ b(k,j) \right) v(j) \tag{12}$$

```
vtabv=0.d+00
do i=1,n
  p=0.d+00
  do j=1,n
    s=0.d+00
    do k=1,n
      s=s+dble(i+k)*dble(k-j)
    enddo
    p=p+s*dble(j)
  enddo
  vtabv=vtabv+dble(i)*p
enddo
```

Estudiar el tiempo de computación de cada uno de los algoritmos anteriores.

a) T(n)
b) T(n^2)

c) $\mathrm{T}(n^2)$

d) $\mathrm{T}(n^3)$

e) $\mathrm{T}(n^3)$

f) $\mathrm{T}(n^2)$

g) $\mathrm{T}(n^3)$

5. Para calcular un valor aproximado de $\alpha = \sqrt{2}$ se proponen los dos algoritmos iterativos siguientes

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

Realizar un programa Fortran para cada uno de ellos y comprobar numéricamente que ambos convergen a $\alpha = \sqrt{2}$ cuando se utiliza la aproximación inicial $x_0 = 1$. A la vista de los resultados obtenidos, ¿qué algoritmo parece más eficiente? Comprobar qué ocurre en el caso de utilizar $x_0 = 0$.

A) (NOTA: Se omiten las tildes en los textos)

```
implicit integer*4(i-n)
implicit real*8(a-h,o-z)

write(6,*)'Indique la aproximacion inicial:'
read(5,*)x
e=1.d-16  ! Cota de error absoluto si la raiz fuese nula.
r=1.d-14  ! Cota de error relativa.
indconv=0
itermax=200
iter=0
do while(indconv.eq.0.and.iter.lt.itermax)
  xn=(2.d+00+x*(10.d+00-x))/10.d+00    !     xn=x/2.d+00+1.d+00/x
  iter=iter+1
  if(abs(x-xn).le.max(e,abs(xn)*r))then
    indconv=1
  endif
  write(6,'(a,i4,a,d25.16,a,d15.6)')' Iter:',iter,
&     ' -> Aprox.:',xn,'     Error rel.:',abs(xn-x)/abs(xn)
  x=xn
enddo

if(indconv.eq.1)then
  write(6,'(a,d25.16)')'La aproximación a sqrt(2) es:',x
else
  write(6,*)'Error: Numero maximo de iteraciones alcanzado'
  write(6,*)'Convergencia no alcanzada'
endif

end
```

	$x_{n+1} = \frac{2+x_n(10-x_n)}{10}$	$x_{n+1} = \frac{x_n}{2} + \frac{1}{x_n}$
0	1	1
1	1.100000000000000	1.500000000000000
2	1.179000000000000	1.416666666666667
3	1.239995900000000	1.414215686274510
4	1.286236916798319	1.414213562374690
5	1.320796376184834	1.414213562373095
⋮	⋮	
90	1.414213562373043	
91	1.414213562373058	
92	1.414213562373068	

Si se utiliza $x_0 = 0$ el algoritmo B) no funciona.

6. Deducir la condición de convergencia de un algoritmo iterativo con orden de convergencia superlineal a partir de la condición general:

$$|x_{k+1} - \alpha| \leq \lambda_p |x_k - \alpha|^p \tag{13}$$

Indicar qué conclusiones se pueden obtener del desarrollo anterior y qué ventajas e inconvenientes tienen estos algoritmos superlineales frente a los lineales.

$$\begin{aligned} |x_{k+1} - \alpha| &\leq \lambda_p |x_k - \alpha|^p \\ &\leq \lambda_p^{1+p} |x_{k-1} - \alpha|^{p^2} \\ &\leq \lambda_p^{1+p+p^2} |x_{k-2} - \alpha|^{p^3} \\ &\vdots \\ &\leq \lambda_p^{1+p+p^2+p^{k-1}} |x_0 - \alpha|^{p^k} \\ &\leq \lambda_p^{\frac{1-p^k}{1-p}} |x_0 - \alpha|^{p^k} \\ &\leq \lambda_p^{\frac{1}{1-p}} \left(\lambda_p^{\frac{1}{p-1}} |x_0 - \alpha| \right)^{p^k} \end{aligned} \tag{14}$$

De modo que la condición de convergencia se reduce a:

$$\lambda_p^{\frac{1}{p-1}} |x_0 - \alpha| < 1 \tag{15}$$

7. En un dispositivo de cálculo se dispone de un tipo de variables de tamaño reducido para almacenar números reales en coma fija en codificación binaria. Las variables ocupan 7 bits, de los cuales 4 se dedican a codificar la mantisa del número (incluido el signo) y 3 se dedican a codificar el exponente (incluido el signo). Deducir a partir de estos datos todos los valores numéricos que se pueden almacenar realmente y cuál sería el error de máquina para este tipo de variables. Considérese para ello que se reserva un bit para el signo tanto del exponente como de la mantisa.

Mant.\ Exp.	$(00)_2$	$\pm(01)_2$	$\pm(10)_2$	$\pm(11)_2$
$\pm(.100)_2$	$(.100)_2\ 2^{(00)_2}$	$\pm(.100)_2\ 2^{(01)_2}$ $\pm(.100)_2\ 2^{-(01)_2}$	$\pm(.100)_2\ 2^{(10)_2}$ $\pm(.100)_2\ 2^{-(10)_2}$	$\pm(.100)_2\ 2^{(11)_2}$ $\pm(.100)_2\ 2^{-(11)_2}$
$\pm(.101)_2$	$(.101)_2\ 2^{(00)_2}$	$\pm(.101)_2\ 2^{(01)_2}$ $\pm(.101)_2\ 2^{-(01)_2}$	$\pm(.101)_2\ 2^{(10)_2}$ $\pm(.101)_2\ 2^{-(10)_2}$	$\pm(.101)_2\ 2^{(11)_2}$ $\pm(.101)_2\ 2^{-(11)_2}$
$\pm(.110)_2$	$(.110)_2\ 2^{(00)_2}$	$\pm(.110)_2\ 2^{(01)_2}$ $\pm(.110)_2\ 2^{-(01)_2}$	$\pm(.110)_2\ 2^{(10)_2}$ $\pm(.110)_2\ 2^{-(10)_2}$	$\pm(.110)_2\ 2^{(11)_2}$ $\pm(.110)_2\ 2^{-(11)_2}$
$\pm(.111)_2$	$(.111)_2\ 2^{(00)_2}$	$\pm(.111)_2\ 2^{(01)_2}$ $\pm(.111)_2\ 2^{-(01)_2}$	$\pm(.111)_2\ 2^{(10)_2}$ $\pm(.111)_2\ 2^{-(10)_2}$	$\pm(.111)_2\ 2^{(11)_2}$ $\pm(.111)_2\ 2^{-(11)_2}$

Mant.\ Exp.	$(00)_2$	$\pm(01)_2$	$\pm(10)_2$	$\pm(11)_2$
$\pm(.100)_2$	+0.5	+1	+2	+4
		-1	-2	-4
	-0.5	+0.5	+0.25	+0.125
		-0.5	-0.25	-0.125
$\pm(.101)_2$	+0.625	+1.25	+2.5	+5
		-1.25	-2.5	-5
	-0.625	+0.3125	+0.15625	+0.078125
		-0.3125	-0.15625	-0.078125
$\pm(.110)_2$	+0.75	+1.5	+3	+6
		-1.5	-3	-6
	-0.75	+0.375	+0.1875	+0.09375
		-0.375	-0.1875	-0.09375
$\pm(.111)_2$	+0.875	+1.75	+3.5	+7
		-1.75	-3.5	-7
	-0.875	+0.4375	+0.21875	+0.109375
		-0.4375	-0.21875	-0.109375