

# ANÁLISIS NUMÉRICO I/ANÁLISIS NUMÉRICO – 2019

## Trabajo de Laboratorio N<sup>o</sup> 1

1. Comprobar que el  $\epsilon$ -máquina es  $2^{-52} = 2.2204 \times 10^{-16}$ , escribiendo en la línea de comandos:

```
>> a = 1 + 2^(-53); b = a-1
```

y comparando con

```
>> a = 1 + 2^(-52); b = a-1
```

2. Obtener el mayor y menor número positivo en punto flotante (*overflow* y *underflow*). Para obtener el mayor número de overflow escribir un ciclo que vaya calculando las sucesivas potencias de 2 y que finalice cuando se produce overflow. Se recomienda utilizar el comando `isfinite` para detectar cuando se produce el overflow. Otra instrucción que puede resultar útil es `break` para interrumpir el ciclo cuando se produce el overflow. El número de underflow se puede obtener dividiendo por 2 repetidamente hasta obtener un número indistinguible del cero en punto flotante.
3. Escribir la siguiente secuencia de comandos en un archivo con extensión `.jl` y ejecutarlo en Julia.

```
x = 0;
while x!=10
    x = x + 0.1
end
```

Para interrumpir la ejecución, pulsar `<CTRL>+<c>`. ¿Qué ocurre si en lugar de incrementarse la variable en 0.1 lo hace en 0.5? ¿Por qué?

4.
  - Escriba un programa que calcule el factorial de 6.
  - Utilice el comando para buscar comandos relacionados con la expresión `fact`.
  - Escriba una función que calcule el factorial de un número  $n$  dado.
5. Escribir un programa que pida dos números reales e imprima en la pantalla el mayor de ellos. El programa debe indicar si los números son iguales.
6. Escribir una función que calcule la potencia  $n$ -ésima de un número, es decir que devuelva  $x^n$  para  $x$  real y  $n$  entero. Realice un programa que utilice la función e imprima en pantalla las primeras 5 potencias naturales de un número ingresado.
7. Escribir dos funciones en Julia para la resolución de ecuaciones de segundo grado  $ax^2 + bx + c = 0$ . Una de ellas, que llamaremos de **mala**, implementando la tradicional fórmula de Baskhara y la otra, que llamaremos de **buena**, usando una manera eficiente para evitar cancelación de dígitos significativos. La sintaxis de la llamada a las funciones debería ser:

```
[x_1, x_2] = buena(a,b,c)
```

y análogamente para **mala**.

8. Escribir una función que implemente el algoritmo de Horner (**horn**) para la evaluación de polinomios. La sintaxis de llamada a la rutina debería ser:

```
> p = horn(coefs,x)
```

donde **p** es el valor del polinomio, **coefs** es un vector con los coeficientes del polinomio, de mayor a menor grado y **x** es el valor de la variable independiente. Es decir que si, por ejemplo, hacemos:

```
> p = horn([1 -5 6 2], 2)
```

entonces la variable **p** almacenará el valor  $p(2)$  donde  $p(x) = x^3 - 5x^2 + 6x + 2$ .