ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS
INGENIERÍA EN COMPUTACIÓN

PERÍODO ACADÉMICO: 2025-A
ASIGNATURA: ICCD412 Métodos Numéricos **GRUPO:** GR2
TIPO DE INSTRUMENTO: Tarea 1
FECHA DE ENTREGA LÍMITE: 04/05/2025
ALUMNO: Murillo Tobar Juan

TEMA

Tipos de errores

OBJETIVOS

- Conocer los límites de nuestros sistemas para evitar errores en la codificación de programas relacionados con la materia de métodos numéricos.
- Reproducir desbordamientos de memoria para el futuro reconocimiento y manejo de estos posibles errores.

MARCO TEÓRICO

No solicitado

DESARROLLO

Para esta tarea se requería de identificar los límites de datos en nuestros computadores y ver que sucede al desbordar esos límites en PYTHON. Para la primera parte utilizamos la librería sys como se puede ver en la siguiente captura.

```
C:\Users\Juan>python
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import sys
>>> x = sys.info_float.max
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: module 'sys' has no attribute 'info_float'
>>> import sys
>>> x = sys.float_info.max
>>> y = sys.float_info.min
>>> z = sys.maxsize
>>> print(f"Juan Murillo, max float = {x}, min float = {y}, max entero = {z}")
Juan Murillo, max float = 1.7976931348623157e+308, min float = 2.2250738585072014e-308, max entero = 9223372036854775807
```

Figura 1: Captura de Python

En la segunda parte debíamos desbordar dichos valores para obtener nan en los resultados.

```
>>> get_Overflow2 = (1/y*10**16) - (1/y*10**16)  #Dividimos en este caso
>>> print(get_Overflow2)
nan
```

Figura 2: Desbordamiento del valor float max positivo

```
>>> get_Overflow = x*10**15 - x*10**15
>>> print(get_Overflow)
nan
```

Figura 3: Desbordamiento del valor float min positivo

En el caso de los enteros surge un inconveniente. Como se menciona en [1], en enteros no puede darse el caso de Overflow, por lo que en este caso unicamente buscamos un MemoryError que en algunos sistemas es el congelamiento de la terminal.

```
>>> try:
...     x = z ** z ** z
... except MemoryError:
...     print("Surgio un error de memoria")
...
```

Figura 4: Congelamiento de la terminal

REFERENCIAS

[1] Python Software Foundation, “Built-in exceptions,” 2025, accessed 2025. [Online]. Available: https://docs.python.org/3/library/exceptions