REPÚBLICA BOLIVARIANA DE VENEZUELA

MINISTERIO DEL PODER POPULAR PARA LA EDUCACIÓN UNIVERSITARIA

UNIVERSIDAD NACIONAL EXPERIMENTAL MARÍTIMA DEL CARIBE

SISTEMAS OPERATIVOS I

CAL-604

SECCIÓN: A

FUNDAMENTOS NUMÉRICOS EN INGENIERÍA

Integrantes:

Alarcón Campuzano, Jonathan Alexander

García Savoca, Anthony Alessandro

Profesor:

Román Ramos

Catia La Mar, junio de 2025

Sección 1: Sistemas Numéricos y Conversión

1. Definición y Ejemplos:

- Defina brevemente los sistemas de numeración decimal, binario, octal y

hexadecimal, explicando su base y los dígitos utilizados.

- Convierta los siguientes números a las bases indicadas, mostrando claramente

los pasos de conversión:

▪ (175)10 a binario y hexadecimal.

▪ (1101011)2 a decimal y octal.

▪ (3F.C)16 a decimal y binario.

2. Aplicación en Computación:

- Explique por qué las computadoras utilizan el sistema binario para representar la información.
- Investigue y describa brevemente un caso de uso práctico en ingeniería donde la comprensión de la conversión entre sistemas numéricos sea crucial (ej. comunicaciones digitales, programación de bajo nivel, etc.).

1.-

a) Definición de los sistemas numéricos:

Sistema	Base	Dígitos utilizados	Descripción
Decimal	10	0-9	Es el sistema que usamos cotidianamente. Cada posición tiene un valor de 10^n, según la posición del dígito.
Binario	2	0, 1	Usado por las computadoras. Cada posición representa una potencia de 2 (2^n).
Octal	8	0-7	Agrupa bits de 3 en 3. Cada posición es una potencia de 8 (8^n).
Hexadecimal	16	0-9 y A-F	Muy usado en programación y electrónica. Cada posición es una potencia de 16 (16^n).

b) Conversión de los números:

$(175)_{10}$ a binario y hexadecimal

A binario:

Dividimos sucesivamente por 2:

División	Cociente	Residuo
175 ÷ 2	87	1
87 ÷ 2	43	1
43 ÷ 2	21	1
21 ÷ 2	10	1
10 ÷ 2	5	0
5 ÷ 2	2	1
2 ÷ 2	1	0
1 ÷ 2	0	1

Leyendo de abajo hacia arriba:
$(175)_{10} = (10101111)_2$

A hexadecimal:

Primero, podemos agrupar los bits de 4 en 4:
$(10101111)_2 \rightarrow 1010\ 1111$

Ahora convertimos cada grupo:

- $1010 \rightarrow A$
- $1111 \rightarrow F$

Entonces:
$(175)_{10} = (AF)_{16}$

$(1101011)_2$ a decimal y octal

A decimal:

Sumamos las potencias de 2 correspondientes:

$$\overset{6}{1}\times 2+\overset{5}{1}\times 2+\overset{4}{0}\times 2+\overset{3}{1}\times 2+\overset{2}{0}\times 2+\overset{1}{1}\times 2+\overset{0}{1}\times 2 = 64+32+0+8+0+2+1=107$$

Por lo tanto:
$(1101011)_2 = (107)_{10}$

A octal:

Agrupamos los bits de 3 en 3 desde la derecha:

$(1101011)_2 \rightarrow 001\ 101\ 011$

Ahora convertimos cada grupo:

- $001 \rightarrow 1$
- $101 \rightarrow 5$
- $011 \rightarrow 3$

Por lo tanto:
$(1101011)_2 = (153)_8$

$(3F.C)_{16}$ a decimal y binario

A decimal:

Descomponemos:

$$(3F.C)_{16} \times 16+F \times 16+C \times 16^{1\quad 0\quad -1}$$

Recordemos que:

- F = 15
- C = 12

Entonces:

$3 \times 16=48$

$15 \times 1=15$

$12 \times 0.0625=0.75$

Sumamos:

$48+15+0.75=63.75$

Por lo tanto:
$(3F.C)_{16} = (63.75)_{10}$

A binario:

- $3 \rightarrow 0011$
- F (15) $\rightarrow$ 1111
- C (12) $\rightarrow$ 1100 (para parte fraccionaria)

Entonces:
$(3F.C)_{16} = (00111111.1100)_2 \rightarrow (111111.1100)_2$ (quitando ceros a la izquierda)

2. Aplicación en Computación

a) ¿Por qué las computadoras utilizan el sistema binario?

Las computadoras usan el sistema binario porque su hardware solo puede detectar dos estados físicos: **encendido (1)** y **apagado (0)**. Estos estados son fáciles de implementar utilizando transistores, los cuales solo necesitan diferenciar entre paso o no paso de corriente eléctrica. El uso del binario reduce la complejidad del diseño y mejora la confiabilidad de los sistemas digitales.

b) Caso de uso práctico en ingeniería

Caso: Comunicaciones digitales (Protocolos de transmisión)

En comunicaciones digitales, los datos se transmiten como secuencias de bits (binario). Para enviar texto, imágenes o audio, estos datos deben ser convertidos entre distintos sistemas de numeración:

- **Codificación de caracteres**: Los textos se codifican en binario mediante estándares como ASCII o Unicode.
- **Direcciones IP**: Se expresan en decimal para los usuarios, pero internamente se manejan en binario.
- **Protocolos de red**: Como TCP/IP utilizan conversiones binario-decimal-hexadecimal para el control de paquetes de datos.

Por tanto, la comprensión de estas conversiones es esencial para los ingenieros que diseñan redes, sistemas de telecomunicaciones y protocolos de comunicación.

Sección 2: Fuentes y Tipos de Errores

1. **Clasificación de Errores:**

a. Describa en detalle las siguientes fuentes de error en el cálculo numérico, proporcionando un ejemplo conciso para cada una:

- Error de truncamiento
- Error de redondeo
- Error inherente (o de modelo)
- Error numérico

b. Explique la diferencia fundamental entre el error absoluto y el error relativo. ¿Cuándo es más útil uno que el otro?

2. Propagación de Errores:
    - Considerando la función $f(x)=x3-2x+1$. Si x tiene un error de Δx, derive una expresión para la propagación del error en f(x) utilizando el concepto de diferencial.
    - Si $x=2\pm0.01$, estime el error máximo en el cálculo de f(x).

1. Clasificación de Errores

Error de truncamiento

- **Definición:**
  Ocurre cuando se corta (trunca) una serie infinita o un cálculo que debería continuar, pero se interrumpe para simplificar el trabajo.
- **Ejemplo:**

  Al calcular e^x con la serie de Taylor:

$$e=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\cdots$$

Si se detiene en el segundo término:

$$e\approx 1+x$$

Se introduce un error de truncamiento al no considerar los términos siguientes.

Error de redondeo

- **Definición:**
  Se produce al aproximar un número a un número con menos cifras decimales de las que realmente tiene.
- **Ejemplo:**

$\pi\approx 3.141592$ se redondea a 3.14

El error de redondeo es:

$3.141592-3.14=0.001592$

Error inherente (o de modelo)

- **Definición:**
  Surge de la simplificación de la realidad al crear un modelo matemático. Todo modelo es una aproximación de un fenómeno real.
- **Ejemplo:**
  Para calcular la trayectoria de un proyectil, se puede ignorar la resistencia del aire. Esa omisión introduce un error inherente, porque el modelo no representa completamente la realidad.

Error numérico

- **Definición:**
  Son los errores que surgen al realizar operaciones numéricas en una

computadora, debido a limitaciones de representación, almacenamiento o algoritmos.

- **Ejemplo:**
  Al sumar números de órdenes de magnitud muy diferentes:

$$1.0000001 + 10^{-10}$$

$$10^{-10}$$

Puede que el sistema no logre representar el 10 debido a la precisión limitada, perdiéndose el valor.

Diferencia entre error absoluto y error relativo

Concepto	Definición	Fórmula
Error absoluto (EA)	Diferencia entre el valor verdadero (V) y el valor aproximado (A)	(EA =
Error relativo (ER)	Relación entre el error absoluto y el valor verdadero	(ER = \frac{EA}{

¿Cuándo usar cada uno?

- El **error absoluto** es útil cuando la magnitud de los valores es pequeña o el orden de magnitud es fijo.
- El **error relativo** es mejor cuando los valores varían mucho en tamaño, ya que permite comparar el tamaño del error respecto al tamaño del dato.

Ejemplo simple:

- Si medimos una distancia de 100 km con un error de 1 km:

  $EA = 1 \text{ km}$

  $ER = \frac{1}{100} = 0.01 \ (1\%)$

2. Propagación de Errores

Derivación general usando diferenciales

Si simplificamos:

$f(x) = (3x - 2x) + 1 = x + 1$

Por lo tanto, la función realmente es:

$f(x) = x + 1$

Concepto de diferencial aplicado a la propagación de errores

Sabemos que el error en f(x), denotado como Δf, se puede aproximar mediante el diferencial:

$\Delta f \approx df = f'(x) \cdot \Delta x$

Donde:

- f′(x) es la derivada de f(x) respecto a x.
- Δx es el error absoluto en la variable x.

Paso 1: Derivada de la función

Dado que:

$f(x) = x + 1$

La derivada es:

$f'(x) = 1$

Paso 2: Expresión de la propagación del error

Entonces el error en f(x) es:

$\Delta f \approx f'(x) \cdot \Delta x = 1 \cdot \Delta x$

Por lo tanto:

$\Delta f \approx \Delta x$

Interpretación:

- El error en f(x) es igual al error en x.
- Como la función es simplemente una traslación de x, el error se propaga directamente sin amplificación ni atenuación.

Cálculo del error máximo cuando $x = 2 \pm 0.01$

- Tenemos:

  $x = 2, \Delta x = 0.01$

- Calculamos f′(2)

  $f'(2) = 3(2)^2 - 2 = 3(4) - 2 = 12 - 2 = 10$:

- Entonces:

  $df = 10 \cdot 0.01 = 0.1$

- El error máximo en f(x)es aproximadamente **±0.1**.

Sección 3: Representación en Formato IEEE 754

1. Fundamentos del Estándar:
    - Explique qué es el estándar IEEE 754 y por qué es importante en la computación numérica.
    - Describa las tres partes principales de un número en punto flotante según este estándar (signo, exponente y mantisa) y explique su función.
2. Conversión Decimal a Punto Flotante:
    - Utilizando el estándar IEEE 754 de precisión simple (32 bits), realice la conversión del número decimal 0.15625 a su representación en punto flotante. Muestre todos los pasos detallados:
        - ▪ Conversión de la parte entera y fraccionaria a binario.
        - ▪ Normalización del número binario.
        - ▪ Cálculo del exponente sesgado (bias).
        - ▪ Formación del número en formato IEEE 754.
3. Conversión Punto Flotante a Decimal:
    - Dado el siguiente número en formato IEEE 754 de precisión simple:
      0 10000001 01100000000000000000000

      conviértalo a su equivalente decimal. Muestre los pasos.

1. Fundamentos del Estándar

¿Qué es el estándar IEEE 754 y por qué es importante?

- El **IEEE 754** es el estándar internacional para representar números en **punto flotante** en computadoras.
- Fue creado para que diferentes computadoras y lenguajes de programación representen los números de forma uniforme, evitando errores al intercambiar datos.
- Permite representar números **muy grandes**, **muy pequeños** y **fraccionarios**, pero con precisión limitada.
- Es esencial porque:
    - Reduce los errores de interpretación.
    - Permite realizar cálculos numéricos estandarizados.
    - Mejora la compatibilidad entre plataformas.

Las tres partes de un número en punto flotante IEEE 754 (precisión simple: 32 bits)

Parte	Tamaño	Función
Signo (S)	1 bit	Indica si el número es positivo (0) o negativo (1).
Exponente (E)	8 bits	Representa la potencia de 2, desplazada por el **bias** (sesgo) de 127.
Mantisa (M)	23 bits	Representa la fracción significativa del número (con un 1 implícito a la izquierda).

Formato general:

$$(-1)^{S} \times 1.M \times 2^{E-127}$$

2. Conversión Decimal a Punto Flotante (IEEE 754 simple, 32 bits)

Paso 1: Conversión de la parte entera y fraccionaria a binario

El número es **0.15625**

- Parte entera: **0** → $(0)_2$
- Parte fraccionaria: convertimos multiplicando sucesivamente por 2:

Multiplicación	Resultado entero	Fracción siguiente
0.15625 × 2 = 0.3125	0	0.3125
0.3125 × 2 = 0.625	0	0.625
0.625 × 2 = 1.25	1	0.25

Multiplicación	Resultado entero	Fracción siguiente
0.25 × 2 = 0.5	0	0.5
0.5 × 2 = 1.0	1	0.0

Por lo tanto:

$0.15625 = (0.00101)_2$

Paso 2: Normalización

Normalizamos el número binario para obtener la forma:

$1.M \times 2^{-3}$

Porque:

$0.001012 = 1.012 \times 2^{-3}$

- La mantisa es: **01**
- El exponente es: **-3**

Paso 3: Calcular el exponente sesgado (bias)

- Bias de 127:

$E = -3 + 127 = 124$

- En binario:

$12410 = 011111002$

Paso 4: Formación del número IEEE 754

Parte	Valor
Signo	0 (positivo)
Exponente	01111100
Mantisa	01000000000000000000000

La mantisa tiene los bits de "01" después del punto, completados con ceros hasta los 23 bits.

Resultado final:

0 01111100 01000000000000000000000

Este es el número **0.15625** en IEEE 754 de 32 bits.

3. Conversión de Punto Flotante a Decimal

Dado:

0 10000001 01100000000000000000000

Paso 1: Identificar las partes

Parte	Valor binario	Valor decimal
Signo (S)	0	positivo
Exponente (E)	10000001	129
Mantisa (M)	011000...	0.011

Paso 2: Calcular el exponente real

Exponente real = $E-127=129-127=2$

Paso 3: Reconstruir el número

La mantisa completa es:

$$1.011=1+\frac{0}{2}+\frac{1}{4}+\frac{1}{8}=1+0.25+0.125=1.375$$

Entonces el número es:

$$(+1)\times 1.375\times 2^{2}=1.375\times 4=5.5$$

Resultado final:

5.5

Sección 4: Aspectos Introductorios Adicionales de Interés

1. Número de Máquina y Épsilon de Máquina:

- Defina qué es el "número de máquina" (o MN) y el "épsilon de máquina" (ϵmach). Explique su relevancia en el cálculo numérico.
- Describa cómo el épsilon de máquina se relaciona con la precisión de la representación de números en punto flotante y la aparición de errores de redondeo.

2. Estabilidad y Convergencia:
- Defina claramente los conceptos de "estabilidad" y "convergencia" en el contexto de los métodos numéricos.
- Proporcione un ejemplo intuitivo (no es necesario un algoritmo complejo) donde un método numérico pueda ser:
  ▪ Estable pero no convergente (si es posible, o explique por qué no lo sería).
  ▪ Convergente pero inestable.
  ▪ Idealmente, estable y convergente.
- Explique por qué ambos conceptos son cruciales para la fiabilidad de las soluciones obtenidas mediante métodos numéricos.

1. Número de Máquina y Épsilon de Máquina

¿Qué es el Número de Máquina?

- Son todos los números que la computadora puede representar exactamente en su sistema de punto flotante.
- Debido a la cantidad limitada de bits, solo se pueden representar ciertos valores; los demás se aproximan.
- Por ejemplo, en IEEE 754 de 32 bits, solo hay un número finito de representaciones posibles.

Relevancia:

- Toda operación numérica que involucre números no representables debe redondearse al número de máquina más cercano.
- Esto causa pequeños errores que se acumulan en algunos cálculos.

¿Qué es el Épsilon de Máquina (ϵmach)?

- Es el **menor número positivo** que, sumado a 1, da un resultado distinto de 1 en el sistema de representación de la computadora.
- Representa el límite inferior de precisión del sistema.

$$1+\epsilon_{mach}>1$$

- Mide la precisión relativa del sistema de punto flotante.

Relevancia:

- Nos indica cuán precisos son los cálculos numéricos.
- Es fundamental en el control de errores de redondeo.
- En IEEE 754 de precisión simple (32 bits), típicamente:

$$\epsilon_{mach} \approx 2^{-23} \approx 1.19 \times 10^{-7}$$

- En doble precisión (64 bits),:

$$\epsilon_{mach} \approx 2^{-52} \approx 2.22 \times 10^{-16}$$

Relación del ϵ_{mach} con la precisión y el redondeo:

- Cuanto menor es el épsilon, **mayor es la precisión** del sistema.
- Si el sistema tiene pocas cifras significativas, el épsilon será mayor, lo que indica mayor propensión al error de redondeo.
- En operaciones sucesivas (por ejemplo, en métodos iterativos), estos pequeños errores pueden acumularse.

2. Estabilidad y Convergencia

Concepto	Definición
Convergencia	Un método numérico es convergente si, al aumentar el número de pasos o iteraciones, el resultado se aproxima al valor verdadero.
Estabilidad	Un método es estable si los pequeños errores (por redondeo, truncamiento, etc.) **no se amplifican significativamente** durante el cálculo.

Ejemplos intuitivos

¿Puede un método ser estable pero no convergente?

- Rara vez ocurre.
- La estabilidad ayuda a controlar los errores, pero no garantiza que el método llegue a la solución verdadera.
- Ejemplo teórico:
  Un método que siempre devuelva el mismo valor fijo, por ejemplo:

$$x_{n+1} = x_0$$

  - No genera error acumulado (estable).
  - Pero no se aproxima a la solución (no convergente).

¿Puede un método ser convergente pero inestable?

- Sí

- El método puede tender al valor correcto, pero los errores numéricos se amplifican.
- Ejemplo:
  Sumar una serie alternante con números muy pequeños y muy grandes.
    - La suma tiende al valor correcto (convergencia).
    - Pero si se acumulan los errores de redondeo por mala organización de los sumandos, puede volverse inestable.

Ideal: Estable y convergente

- Ejemplo: Método de bisección para resolver f(x)=0.
    - Siempre converge al resultado correcto.
    - Los errores numéricos no se amplifican.

¿Por qué son importantes ambos conceptos?

- **Convergencia** garantiza que el método nos da el resultado correcto si hacemos suficientes pasos.
- **Estabilidad** garantiza que los errores numéricos pequeños no se vuelven incontrolables.
- **Ambos son necesarios** en métodos numéricos para garantizar resultados confiables en computación científica e ingeniería.

REFERENCIAS

Burden, R. L., & Faires, J. D.
Análisis Numérico (9ª Edición). Brooks/Cole. Cengage Learning, 2011.

Chapra, S. C., & Canale, R. P.
Métodos Numéricos para Ingenieros (7ª Edición). McGraw-Hill, 2015.

Goldberg, D.
Lo que todo científico de la computación debe saber sobre aritmética de punto flotante. ACM Computing Surveys, Vol. 23, No. 1 (marzo de 1991), págs. 5–48.

Heath, M. T.
Computación Científica: Una Introducción (2ª Edición). McGraw-Hill, 2002.

IEEE Computer Society
Norma IEEE para Aritmética de Punto Flotante (IEEE 754-2008). IEEE, 2008 (actualizada en 2019).

Kincaid, D., & Cheney, W.
Análisis Numérico: Matemáticas de la Computación Científica (3ª Edición) Brooks/Cole, 2002.