UNIVERSIDAD BANCARIA DE MÉXICO

"Constancia Unidad y trabajo"

INGENIERÍA EN SISTEMAS COMPUTACIONALES

RECONOCIMIENTO DE VALIDEZ OFICIAL DE ESTUDIOS DE LA SECRETARÍA DE EDUCACIÓN PÚBLICA No. 2022241 DE FECHA 13 DE SEPTIEMBRE DE 2002.

NOMBRE DE LA MATERIA:

Métodos numéricos

NOMBRE DEL PROFESOR(A):

Mauricio Gómez

CUATRIMESTRE:

6to Cuatrimestre

TÍTULO DEL TRABAJO O INVESTIGACIÓN:

Programa #3

NOMBRE DE ALUMNO(S):

Francisco de Jesus Pincle Puente

FECHA DE ENTREGA:

22 de mayo 2025

Introducción

El programa es una aplicación web interactiva que calcula el valor de e^x utilizando la serie de Mclaurin, una técnica matemática para aproximar funciones mediante sumas infinitas. La aplicación permite al usuario:

- Ingresar un valor para x (puede ser un número decimal o una fracción)
- Especificar un margen de error deseado
- Visualizar el proceso de aproximación paso a paso
- Comparar la aproximación con el valor real de e^x

La interfaz incluye animaciones, validación de datos y una presentación clara de los resultados en formato tabular.

Fundamento Matemático

La serie de Mclaurin para e^x se define como:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$$

Donde:

- e es la base del logaritmo natural (aproximadamente 2.71828)
- x es el exponente
- n! representa el factorial de n

El programa calcula esta serie término a término hasta alcanzar la precisión especificada por el usuario.

Estructura del Programa

El programa está estructurado en tres partes principales:

1. HTML: Define la estructura de la interfaz de usuario

2. CSS: Proporciona estilos y animaciones

3. JavaScript: Implementa la lógica de cálculo y la interactividad

Interfaz de Usuario

La interfaz de usuario consta de los siguientes elementos:

Formulario de entrada

- Campo para el valor de x
- Campo para el margen de error
- Botón "Calcular"
- Mensajes de error para validación

Indicador de carga

- Muestra un spinner durante el cálculo

Información del resultado

- Valor real de e^x
- Número de términos necesarios para la aproximación

Tabla de resultados

- Número de términos
- Valor aproximado
- Error relativo (%)
- Error absolute

Funciones Principales

A continuación se detallan las funciones JavaScript más importantes del programa:

1. Función `parseFraction`

Esta función convierte fracciones en texto (como "1/2") a valores decimales (0.5).

```javascript
/**
 * Función para analizar fracciones ingresadas como texto
 * Permite al usuario ingresar valores como "1/2" y los convierte a decimales (0.5)
 * @param {string} input - El texto ingresado por el usuario
 * @return {number} - El valor decimal de la fracción o el número ingresado
 */
function parseFraction(input) {
  // Verifica si el input contiene una barra de división (/)
  if (input.includes('/')) {
    // Divide el texto en numerador y denominador
    const parts = input.split('/');
    if (parts.length === 2) {
      const numerator = parseFloat(parts[0]);
      const denominator = parseFloat(parts[1]);

      // Verifica que ambos valores sean números válidos y que el denominador no sea cero
      if (!isNaN(numerator) && !isNaN(denominator) && denominator !== 0) {
        return numerator / denominator; // Retorna la división
      }
    }
    return NaN; // Retorna NaN si la fracción no es válida
  }

  // Si no es una fracción, intenta convertir directamente a número
  return parseFloat(input);
}
```

2. Función `validateInput`

Valida los datos ingresados por el usuario y muestra mensajes de error si son inválidos.

```javascript
/**
 * Valida los datos ingresados por el usuario
 * Muestra mensajes de error si los datos no son válidos
 * @param {string} inputId - ID del elemento de entrada a validar
 * @param {string} errorId - ID del elemento donde mostrar el error
 * @return {number|boolean} - El valor numérico si es válido, o false si no lo es
 */
function validateInput(inputId, errorId) {
  const input = document.getElementById(inputId).value.trim();
  const errorElement = document.getElementById(errorId);

  // Verifica si el campo está vacío
  if (input === '') {
    errorElement.style.display = 'block';
    return false;
  }

  let value;
  // Para el valor de x, permite fracciones
  if (inputId === 'x') {
    value = parseFraction(input);
  } else {
    value = parseFloat(input);
  }

  // Verifica si el valor es un número válido
  if (isNaN(value)) {
    errorElement.style.display = 'block';
    return false;
  }

  // Si todo está bien, oculta el mensaje de error y retorna el valor
  errorElement.style.display = 'none';
  return value;
}
```

3. Función `factorial`

Calcula el factorial de un número, necesario para la serie de Mclaurin.

```javascript
/**
 * Calcula el factorial de un número
 * Necesario para la serie de Mclaurin: e^x = 1 + x/1! + x²/2! + x³/3! +
 * @param {number} num - El número para calcular su factorial
 * @return {number} - El factorial del número
 */
function factorial(num) {
  if (num === 0 || num === 1) return 1; // Caso base: 0! = 1! = 1

  let fact = 1;
  // Multiplica todos los números desde 2 hasta num
  for (let i = 2; i <= num; i++) {
    fact *= i;
  }
  return fact;
}
```

4. Función `calcular`

Función principal que realiza el cálculo de e^x usando la serie de Maclaurin y actualiza la interfaz.

```
/**
 * Función principal que realiza el cálculo de e^x usando la serie de Mclaurin
 * Muestra los resultados en una tabla con la aproximación y los errores
 */
function calcular() {
  // Oculta los mensajes de error previos
  document.getElementById('x-error').style.display = 'none';
  document.getElementById('error-error').style.display = 'none';

  // Valida los datos ingresados
  const x = validateInput('x', 'x-error');
  const margen = validateInput('error', 'error-error');

  // Si algún dato no es válido, detiene la ejecución
  if (x === false || margen === false) {
    return;
  }

  // Muestra el indicador de carga
  document.getElementById('loading').style.display = 'block';

  // Limpia la tabla de resultados anteriores
  const tabla = document.getElementById("tablaResultados").querySelector("tbody");
  tabla.innerHTML = "";

  // Usa setTimeout para no bloquear la interfaz durante los cálculos
  setTimeout(() => {
    let termino = 1; // Primer término de la serie (x^0/0! = 1)
    let suma = 1;    // Suma inicial (comienza con el primer término)
    let anterior = 0;
    let n = 1;       // Contador de términos (comenzamos desde el segundo término)

    // Calcula el valor real de e^x usando la función Math.exp
    const resultadoReal = Math.exp(x);
    let delay = 0;   // Retraso para la animación de las filas
```

```
// Bucle principal para calcular la serie de Mclaurin
while (true) {
  // Calcula el siguiente término: x^n/n!
  let nuevoTermino = Math.pow(x, n) / factorial(n);
  suma += nuevoTermino; // Añade el término a la suma

  // Calcula los errores
  let errorAbsoluto = Math.abs(resultadoReal - suma);
  let errorRelativo = Math.abs((resultadoReal - suma) / resultadoReal) * 100;

  // Crea una nueva fila en la tabla para este término
  const fila = tabla.insertRow();
  fila.style.animationDelay = `${delay}ms`; // Retraso para efecto escalonado
  delay += 50; // Incrementa el retraso para la siguiente fila

  // Llena la fila con los datos calculados
  fila.innerHTML = `
    <td>${n + 1}</td>
    <td>${suma.toFixed(10)}</td>
    <td>${errorRelativo.toFixed(6)}%</td>
    <td>${errorAbsoluto.toExponential(6)}</td>
  `;

  // Si el error absoluto es menor o igual al margen deseado, termina
  if (errorAbsoluto <= margen) break;

  // Incrementa el contador para el siguiente término
  n++;
  // Límite de seguridad para evitar bucles infinitos
  if (n > 100) break;
}

// Actualiza la información del resultado
document.getElementById('x-value').textContent = x;
document.getElementById('real-value').textContent = resultadoReal.toFixed(10);
document.getElementById('terms-needed').textContent = n + 1;
```

Estilos y Animaciones

El programa utiliza CSS moderno para crear una interfaz atractiva y responsiva:

- Fondo animado**: Gradiente que cambia de colores suavemente
- Efecto de flotación**: El contenedor principal tiene una animación sutil de flotación
- Animaciones de carga**: Spinner durante el cálculo
- Animaciones de tabla**: Las filas aparecen con un efecto escalonado
- Efectos hover**: Elementos interactivos responden visualmente al pasar el cursor

Mejores Prácticas Implementadas

El programa implementa varias buenas prácticas de desarrollo web:

- **Código comentado:** Todas las funciones tienen comentarios explicativos
- **Validación de entrada:** Verifica que los datos ingresados sean válidos
- **Procesamiento asíncrono:** Usa setTimeout para no bloquear la interfaz
- **Diseño responsivo:** Se adapta a diferentes tamaños de pantalla
- **Feedback visual:** Proporciona retroalimentación clara al usuario
- **Límites de seguridad:** Evita bucles infinitos con un límite máximo de iteraciones
- **Accesibilidad por teclado:** Permite navegar y ejecutar con la tecla Enter