Documentación de la Aplicación de Métodos de Optimización

Universidad - Modelado Matemático

Fecha: 26/10/2025

# ÍNDICE

**1.** Introducción

**2.** Arquitectura de la Aplicación

**3.** Métodos de Optimización Implementados

**4.** Análisis del Código por Componentes

**5.** Funcionalidades Principales

**6.** Tecnologías Utilizadas

# 1. INTRODUCCIÓN

Esta aplicación web desarrollada en React implementa seis métodos de optimización unidimensional para encontrar el mínimo de funciones matemáticas. La aplicación proporciona una interfaz gráfica intuitiva que permite visualizar el proceso de optimización y los resultados obtenidos.

Los métodos implementados son fundamentales en el campo de la optimización matemática y se utilizan ampliamente en ingeniería, economía, y ciencias computacionales para resolver problemas de minimización.

# 2. ARQUITECTURA DE LA APLICACIÓN

La aplicación sigue una arquitectura modular basada en componentes React:

**•** App.js: Componente principal que renderiza el selector de métodos

**•** OptimizationMethodSelector.jsx: Componente selector que permite elegir entre los diferentes métodos

**•** Métodos individuales: Cada método de optimización está implementado en su propio componente

**•** Visualización: Utiliza la librería Recharts para generar gráficos interactivos

# 3. MÉTODOS DE OPTIMIZACIÓN IMPLEMENTADOS

|  |  |  |
| --- | --- | --- |
| Método | Descripción | Ventajas |
| Sección Aurea | Divide el intervalo usando la proporción áurea | Convergencia garantizada |
| Fibonacci | Usa la secuencia de Fibonacci para dividir intervalos | Eficiente para funciones unimodales |
| Búsqueda Secuencial | Evalúa puntos de forma secuencial | Simple de implementar |
| Búsqueda Dicotómica | Divide el intervalo por la mitad | Convergencia lineal |
| Bisección | Encuentra raíces de la derivada | Requiere derivada |
| Newton Unidimensional | Usa primera y segunda derivada | Convergencia cuadrática |

# 4. ANÁLISIS DEL CÓDIGO POR COMPONENTES

## 4.1 Componente Principal (App.js)

El componente App.js es el punto de entrada de la aplicación. Su estructura es simple pero fundamental:

```javascript

import OptimizationMethodSelector from './components/OptimizationMethodSelector';

import './App.css';

function App() {

return (

<div className="App">

<OptimizationMethodSelector />

</div>

);

}

export default App;

```

Este componente importa el selector de métodos y lo renderiza dentro de un contenedor con la clase CSS 'App'.

## 4.2 Selector de Métodos (OptimizationMethodSelector.jsx)

El selector de métodos es el componente central que permite al usuario elegir entre los diferentes algoritmos de optimización. Utiliza el hook useState para manejar el estado del método seleccionado y renderiza condicionalmente el componente correspondiente.

Características principales:

**•** Estado reactivo: Maneja el método seleccionado usando useState

**•** Renderizado condicional: Muestra solo el componente del método elegido

**•** Interfaz intuitiva: Botones claros para cada método de optimización

## 4.3 Implementación de Métodos de Optimización

Cada método de optimización está implementado como un componente React independiente. Todos siguen una estructura similar pero con algoritmos específicos:

Estructura común de los componentes:

**•** Estados locales: Parámetros del algoritmo, resultados, y configuración

**•** Funciones matemáticas: Implementación del algoritmo específico

**•** Visualización: Gráficos usando Recharts para mostrar el proceso

**•** Interfaz de usuario: Controles para configurar parámetros y ejecutar el algoritmo

## 4.4 Funciones Personalizadas

Una característica importante de la aplicación es la capacidad de ingresar funciones matemáticas personalizadas. Esto se implementa mediante:

```javascript

function parseCustomFunction(funcString) {

try {

// Validación de caracteres permitidos

const allowedChars = /^[x0-9+\-\*/().\s,^sin|cos|tan|log|exp|sqrt|abs|pow|min|max]+$/;

if (!allowedChars.test(funcString)) {

throw new Error("Función contiene caracteres no permitidos");

}

// Conversión de operadores matemáticos

let processedFunc = funcString

.replace(/\^/g, '\*\*') // Convertir ^ a \*\*

.replace(/sin\(/g, 'Math.sin(')

.replace(/cos\(/g, 'Math.cos(')

// ... más conversiones

// Crear función JavaScript

const func = new Function('x', `return ${processedFunc}`);

return func;

} catch (error) {

throw new Error(`Error al parsear función: ${error.message}`);

}

}

```

Esta función permite al usuario ingresar expresiones matemáticas como 'x^2-4\*x+3' y las convierte a código JavaScript ejecutable de forma segura.

# 5. FUNCIONALIDADES PRINCIPALES

## 5.1 Visualización Gráfica

La aplicación utiliza la librería Recharts para crear visualizaciones interactivas que muestran:

**•** Función objetivo: La gráfica de la función que se está optimizando

**•** Puntos de evaluación: Los puntos donde el algoritmo evalúa la función

**•** Punto mínimo: El resultado final del algoritmo destacado visualmente

**•** Puntos auxiliares: Puntos intermedios del proceso de optimización

## 5.2 Configuración de Parámetros

Cada método permite configurar parámetros específicos:

**•** Intervalo de búsqueda: Límites inferior y superior del dominio

**•** Tolerancia: Precisión deseada para la convergencia

**•** Número máximo de iteraciones: Límite para evitar bucles infinitos

**•** Punto inicial: Valor de inicio para métodos que lo requieren

## 5.3 Funciones Predefinidas

La aplicación incluye funciones matemáticas comunes para pruebas:

**•** x^2 - 4x + 3: Parábola con mínimo en x = 2

**•** x^3 - 6x^2 + 9x + 1: Función cúbica con punto crítico

**•** sin(x)^2 + cos(x)^2: Función trigonométrica

**•** exp(-x^2): Función exponencial (campana de Gauss)

# 6. TECNOLOGÍAS UTILIZADAS

## 6.1 Frontend

**•** React: Framework de JavaScript para interfaces de usuario

**•** Recharts: Librería para visualización de datos y gráficos

**•** Tailwind CSS: Framework de CSS para estilos responsivos

**•** Framer Motion: Librería para animaciones fluidas

## 6.2 Herramientas de Desarrollo

**•** Node.js: Entorno de ejecución de JavaScript

**•** npm: Gestor de paquetes de Node.js

**•** ESLint: Herramienta de análisis de código

**•** Webpack: Empaquetador de módulos

# CONCLUSIÓN

Esta aplicación demuestra la implementación práctica de métodos de optimización unidimensional en una interfaz web moderna. La combinación de algoritmos matemáticos robustos con tecnologías web actuales proporciona una herramienta educativa y práctica para el estudio de la optimización matemática.

La arquitectura modular permite fácil extensión y mantenimiento, mientras que las características de visualización interactiva facilitan la comprensión de los procesos de optimización.

El código está diseñado para ser educativo, mostrando claramente la implementación de cada algoritmo y proporcionando retroalimentación visual inmediata sobre el comportamiento de los métodos de optimización.

# INFORMACIÓN ADICIONAL

Para ejecutar la aplicación:

**1.** Instalar dependencias: **npm install**

**2.** Iniciar servidor de desarrollo: **npm start**

**3.** Abrir navegador en: **http://localhost:3000**

Desarrollado como parte del curso de Modelado Matemático - Universidad