Project Documentation:

LDU Factorisation Web Application

Overview

This project implements a web-based application designed to perform LDU (Lower-Diagonal-Upper) decomposition of an n×n matrix. The application breaks down a square matrix into the following components:

- **L**: Lower triangular matrix
- **D**: Diagonal matrix
- **U**: Upper triangular matrix

The application is a learning tool for matrix decomposition and features a dynamic interface for matrix input and visualization of results.

---

1. User Interface

The interface of the application consists of the following features:

- **Homepage**:
  The homepage includes navigation links to two main sections:
    - **Matrix Generator Page**: Allows users to specify and input matrices for factorisation.
    - **Record Page**: Displays logs and team records for the application development.
- **Matrix Size Input**:
    - Users can specify the size (n) of the matrix using a numeric input field.
    - Upon specifying the size, a dynamic grid of input fields is generated for the matrix elements.
- **Matrix Input Grid**:
    - The grid is dynamically created based on the matrix size, where each cell is an editable field initialized to 0.
- **Factorise Button**:
    - Users can click this button to trigger the LDU factorisation process.
    - The results, along with step-by-step computations, are displayed below the input grid.
- **Results Display**:

- The decomposed matrices (L, D, U) are displayed in a neat, grid-based format.
- Steps of computation are also provided for better understanding.

---

2. Visual Design

The application features a clean and user-friendly design:

- **Background**:
  A visually appealing colour scheme with soft, gradient backgrounds.
- **Input Area**:
  Matrix inputs are centrally aligned and styled for clarity, ensuring a structured and organized appearance.
- **Interactive Buttons**:
  Styled with hover effects to make the interface interactive and visually engaging.
- **Matrix Display**:
  - Outputs are shown in a structured grid, ensuring values are easy to read.
  - Font styles and colours are chosen to enhance readability.
- **Responsive Layout**:
  The interface is fully responsive, making it accessible on desktops, tablets, and mobile devices.

---

3. Functionality

The main functionality revolves around creating and factorizing matrices. Key functions include:

- **Matrix Generation**:
  - Users specify the matrix size (n), and a grid of input fields is dynamically created.
  - The matrix grid adjusts automatically based on the size entered.
- **LDU Factorisation**:
  - The application performs LDU decomposition of the input matrix, displaying intermediate steps and the final results.
  - If the matrix cannot be factorised (e.g., due to zero diagonal elements), the application provides error handling with appropriate messages.

---

4. Key Features

- **Dynamic Matrix Input**:
  A responsive and interactive input grid for any n×n matrix.

- **Step-by-Step Calculations**:
  Detailed steps of the LDU decomposition process are displayed for user understanding.
- **Interactive Design**:
  Hover effects and animations enhance the user experience.
- **Error Handling**:
  Alerts users when invalid inputs or computational errors (like division by zero) occur.

---

5. Technologies Used

- **HTML**:
  For structuring the web application, creating input fields, and interactive buttons.
- **CSS**:
  - Styles the overall interface, including the matrix grid and buttons.
  - Adds responsiveness and visual effects like animations.
- **JavaScript**:
  - Handles the logic for matrix generation and LDU factorisation.
  - Provides dynamic updates to the interface based on user input.

JavaScript Code Explanation

In this section, we will go through the JavaScript code step by step to explain its functionality and how it implements the LDU factorisation process for an n×n matrix. The code is responsible for generating the matrix input grid, performing the factorization, and displaying the results.

---

1. Matrix Initialization and Input

The JavaScript code starts by setting up empty matrices and initializing variables:

- A: This matrix holds the input matrix provided by the user.
- L: This is the lower triangular matrix that will be computed during the LDU factorisation.
- D: This stores the diagonal matrix.
- U: This is the upper triangular matrix.
- steps: This array keeps a record of the steps during factorisation, which is useful for debugging or providing a step-by-step explanation to the user.

---

2. Creating and Resetting the Matrix

- This function is triggered when the user specifies the size of the matrix. It reads the value from the input field with id="matrixSize".
- The function validates the input to ensure it is a positive integer.
- It then dynamically generates an HTML grid (<input> fields) for the matrix elements, allowing the user to input values for each element.
- The resetMatrices() function is called to initialize empty matrices for A, L, D, and U.

---

3. Resetting Matrices

- This function resets all matrices (A, L, D, U) to their initial values.
- The L, D, and U matrices are initialized with 0s using Array.from(), and L[i][i] and U[i][i] will later be set to 1 for the factorisation process.

---

4. Initializing Matrices with User Input

- This function reads the values entered by the user in the matrix grid and stores them in the matrix A.
- The diagonal elements of L and U are set to 1 as part of the LDU decomposition.

---

5. LDU Factorisation Process

- **Factorisation Loop:**
  - The function begins by initializing matrices with user input values.
  - Steps is populated with the initial state of matrices A, L, D, and U.
  - The LDU decomposition is performed in a loop for each row k from 0 to n−1:
    - Diagonal matrix (D): The diagonal element D[k][k] is set to A[k][k].
    - Lower matrix (L): The elements below the diagonal in L are computed by dividing A[i][k] by D[k][k].
    - Upper matrix (U): The elements above the diagonal in U are computed by dividing A[k][j] by D[k][k].
    - Update A: After computing L and U, the elements of A are updated by subtracting the product of L[i][k], D[k][k], and U[k][j].
  - The process continues until the LDU factorization is complete.

---

6. Displaying Results

- **Displaying Steps**:
    - This function takes the steps array and formats it for display.
    - Each step is displayed as a heading and includes the matrix at that step.
    - The matrices L, D, and U are displayed in their final state after the factorisation process.

---

7. Displaying Matrix

- This function formats and displays a matrix in a readable grid format.
- It loops through the matrix and displays each value with two decimal places (toFixed(2)).

Conclusion

The JavaScript code implements the core functionality of the LDU factorisation application. It handles matrix creation, LDU decomposition, and step-by-step computation, along with displaying the results in a user-friendly format.

This web application serves as an educational tool for understanding and visualizing the LDU decomposition process. It currently computes and displays the decomposition along with steps. With its user-friendly interface and responsive design, the tool is well-suited for both students and educators in linear algebra.