Project Report

On

Sudoku Verifier

By

Jay Shah : AU2340069

Course Name: ENR112- Linear Algebra Laboratory

Course Instructor: Prof. Bhawnath Tiwari

School of Engineering and Applied Science

Ahmedabad University, Gujarat, India

● Problem Defination

The Sudoku solver is an Matlab algorithm that would help people to

Verify if their sudoku solution is correct or not.

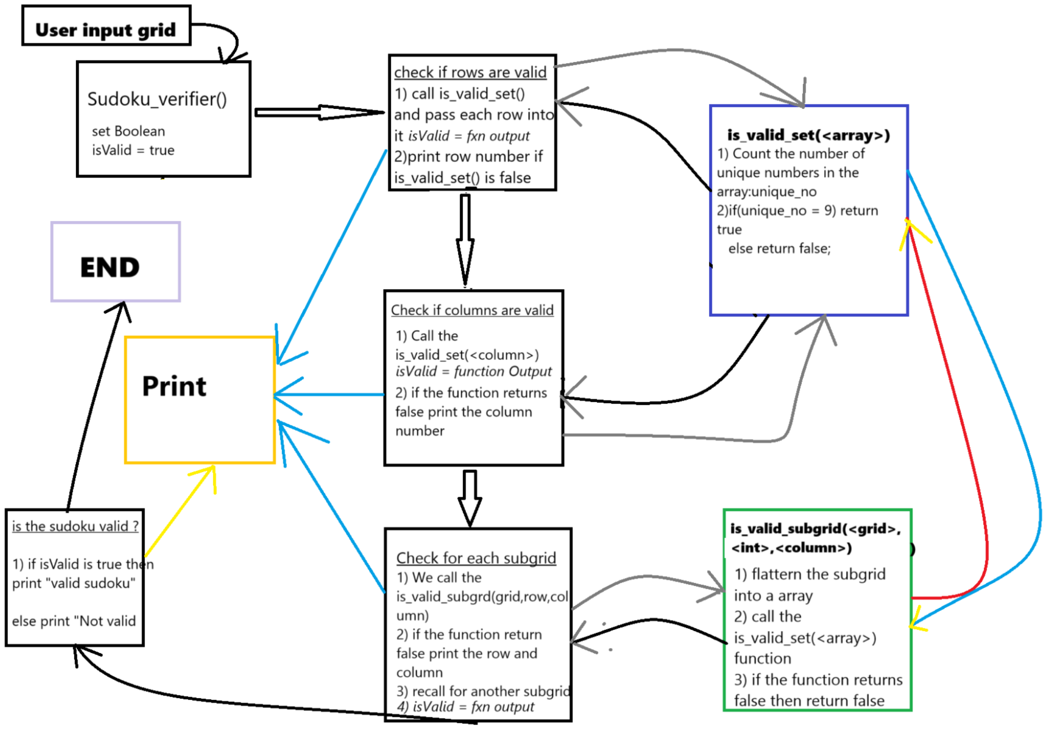
Features:

- ➔ Verify the given sudoku matrix (Sudoku Attempt by user)
- ➔ Conditions for valid sudoku:
    - ◆ Each **column** in the solution matrix must have **unique** numbers
    - ◆ Each **row** in the solution matrix must have **unique** numbers
    - ◆ The sudoku matrix is a 9X9 matrix and consists of 3X3 mini squares, each mini square should contain **unique** numbers
- ➔ Informs the user of the location of the mistake, I.e. the location at which the user solution is incorrect
- ➔ Helps user rectify the solution if it is incorrect.

.

• Solution Algorithm:

➔ Flowchart



➔ Pseudo code of the algorithm:

- ◆ User calls the Function sudoku_verifier(<grid>)
- ◆ Set isValid = true
- ◆ Check if all rows have unique elements by calling the is_valid_set(<row>) function.
  - The is_valid_set(<row>) counts the number of unique elements in the given row.
  - If unique_elements = 9, all elements in that row are equal
  - If unique_elements!= 9 function return false, make isValid = false

- Similarly, check for all columns using the is_valid_set(<column>) function
- Print the location at which the sudoku solution is failing
- Now check for the 3X3 subgrids by calling the is_valid_subgrid(<grid>, <row>, <column>) function
    - From the given row and column, we can make the subgrid by taking a square of length and breadth three from the origin as <row>,<column>
    - Now we need to flatten the grid from a 3x3 form to a 1x9 form
    - Now call the is_valid_set(<flattern_grid>)
        - If returns false, print the location
        - Make isValid = false
        - Else, continue to the next subgrid
- If the boolean isValid is equal to true at the end of this algorithm, then the given sudoku grid is a correct solution.

➔ Matlab function code:

- **Code:**

```
fprintf('Enter your 9x9 Sudoku grid:\n');
   grid = zeros(9, 9);
   for i = 1:9
       rowInput = input(sprintf(''), 's');  % Take row as a string
       grid(i, :) = str2num(rowInput);  % Convert string to numeric row
   end
sudoku_verifier(grid);
```

- **Code sudoku_verifier() function:**

```
function isValid = sudoku_verifier(board)
   isValid = true;
   % Check rows
   for row = 1:9
```

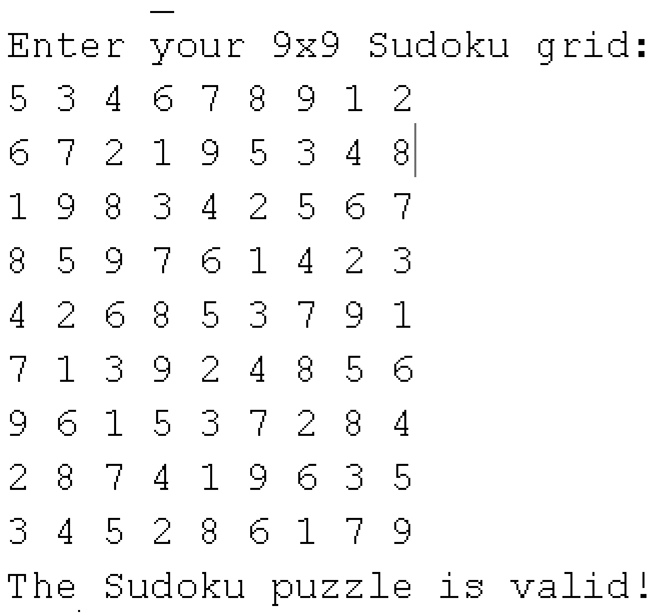
```
       if ~is_valid_set(board(row, :))
           fprintf('Invalid row %d\n', row);
           isValid = false;
       end
   end

   % Check columns
   for col = 1:9
       if ~is_valid_set(board(:, col))
           fprintf('Invalid column %d\n', col);
           isValid = false;
       end
   end

   % Check 3x3 subgrids
   for row = 1:3
       for col = 1:3
           if ~is_valid_subgrid(board, row, col)
               fprintf('Invalid 3x3 subgrid at (%d, %d)\n',
row, col);
               isValid = false;
           end
       end
   end

   if isValid
       disp('The Sudoku puzzle is valid!');
   else
       disp('The Sudoku puzzle is invalid.');
   end
end
function valid = is_valid_set(nums)
   valid = all(nums >= 1 & nums <= 9) && numel(unique(nums))
== 9;
end
function valid = is_valid_subgrid(board, row, col)
   subgrid = zeros(1, 9);  % Preallocate a 1x9 vector for
the subgrid
   k = 1;  % Index for the subgrid vector
   % Calculate the starting indices of the subgrid
   rowStart = (row - 1) * 3 + 1;
   colStart = (col - 1) * 3 + 1;
   % Iterate over the 3x3 subgrid using nested loops
   for i = 0:2  % Row offset within the subgrid
       for j = 0:2  % Column offset within the subgrid
           subgrid(k) = board(rowStart + i, colStart + j);
           k = k + 1;  % Move to the next position in the
vector
```

```
        end
    end
    % Validate the subgrid
    valid = is_valid_set(subgrid);
end
%function valid = is_valid_subgrid(board, row, col)
%   subgrid = board((row-1)*3+1:row*3, (col-1)*3+1:col*3);
 %  valid = is_valid_set(subgrid(:));
%end
```

- ## Matlab functionalities applied:
  - ➔ **Matix operations**
    - ◆ Extraction of entire rows and columns
    - ◆ Conversion of a matrix into an array.
  - ➔ **Looping and Operations**
    - ◆ Using nested for loops to access all the elements of the matrix
    - ◆ Using basic logical operators like && (*and*) and |~(*not*)
  - ➔ **Functions**
    - ◆ User-defined functions is_valid_subgrind()
    - ◆ **Unique():** Counts the number of unique elements in the array
    - ◆ **Numel():** Counts the number of elements
    - ◆ **fprintf():** To print
    - ◆ **disp()**: To print
    - ◆ **zeros():** To pre-allocate 0 to the array/matrix
    - ◆ **str2num():** To convert string to a numeric row
    - ◆ **input():** Used to take the input

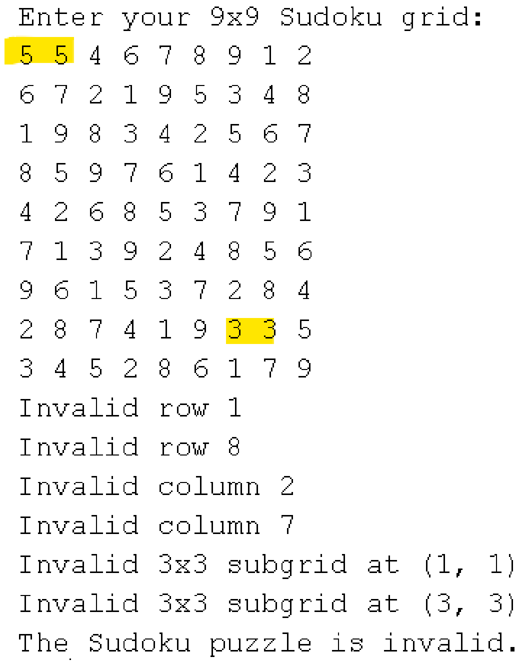
• Demonstration

➔ Lets try a valid sudoku grid



```
Enter your 9x9 Sudoku grid:
5 3 4 6 7 8 9 1 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 1 7 9
The Sudoku puzzle is valid!
```

➔ Lets try a invalid sudoku grid



```
Enter your 9x9 Sudoku grid:
5 5 4 6 7 8 9 1 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 3 3 5
3 4 5 2 8 6 1 7 9
Invalid row 1
Invalid row 8
Invalid column 2
Invalid column 7
Invalid 3x3 subgrid at (1, 1)
Invalid 3x3 subgrid at (3, 3)
The Sudoku puzzle is invalid.
```

● Conclusion

1) I was successfully able to construct an algorithm in MATLAB that can be used to tell if a sudoku grid is correct or not.
2) Learnt to make user-defined functions
3) I learnt to use various new in-built functions from Matlab like unique, Numel and str2sum
4) Learnt to take input from users in Matlab
5) Applied matrix manipulation methods, especially converting the subgrid to an array i.e. from 2D to 1D.

This project can be used in the following real-world applications:

- ➔ It can be used in digital sudoku puzzles to check if the user's solution is correct or not
- ➔ Can be used to give feedback to the user in case the solution is incorrect