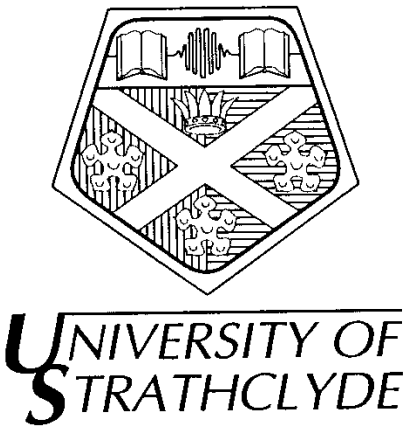

Introduction to MATLAB Coursework

Written by

Euan Foster

April 2019

Email: e.foster@strath.ac.uk
Tel: 07539 450593

1 INTRODUCTION

Develop and produce a MATLAB code to solve the following Sudoku problem:

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Figure 1 - Sudoku Problem

2 EXPLANATION OF MATLAB CODE

2.1 PSEUDOCODE STRUCTURE OF OVERALL SCRIPT

The overall MATLAB script generated can be described by the following flow chart:

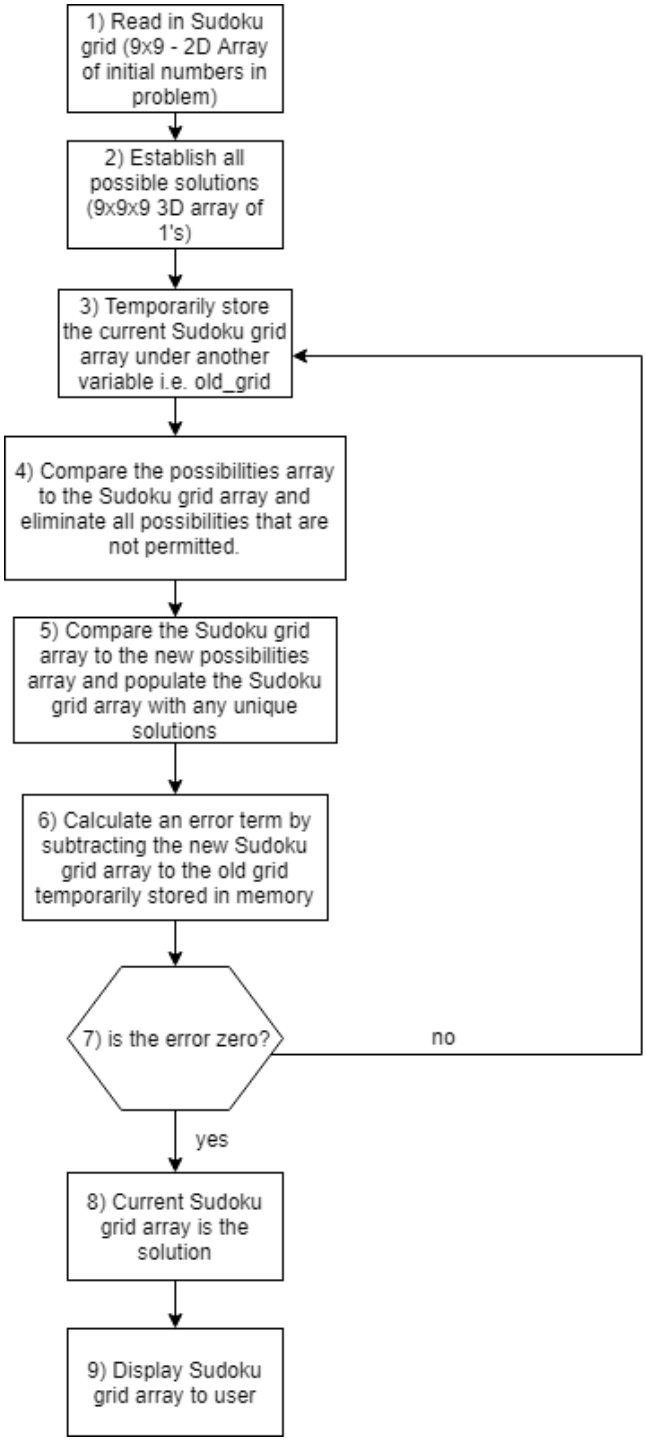

Figure 2 - Pseudo Code of MATLAB Script

Step 3-7 in Figure 2 are carried out in a while loop that breaks when the error term is zero. Steps 4 & 5 have been functionised and will be explained in more depth in the following sections.

It is important to note that the possibilities array is a 9x9x9 array with the first two dimensions being the spatial co-ordinates of the Sudoku grid and the last dimension being representative of all possible numbers 1-9. This is initially set so that all values are possible at all points of the grid, which is impossible for a solution to the problem. The possibilities array is refined as described in Figure 2 by comparing it to the Sudoku grid. The Sudoku grid is refined by populating any unique values found in the updated possibilities array. This process is repeated until the grid is no longer updated, and hence a solution has been found.

2.2 ELIMINATE POSSIBILITIES FUNCTION

The elim_poss.m function has 4 distinct sections to it. The first section loops through all values possibilities and compares it to the grid. Based from what is initially given, there will be certain values that are not possible and these are removed from the possibilities array at the position they are identified in the Sudoku grid array.

```
[r,c,d] = size(poss);

%Initial comparison to grid
for z = 1:d
    for y = 1:r
        for x = 1:c
            if grid(y,x) ~= 0 && grid(y,x) ~= z
                poss(y,x,z) = 0;
            end
        end
    end
end
```

Listing 1 – Removal of Possibilities from Initial Grid Comparison

The second section removes any possibility of a number in a row being used twice. This is in line with the rules of the problem that state that all number from one to nine can only be used once in each row. This section loops through the grid array and identifies any values that are non-zero, and removes the possibility of that number being used twice in the row it was identified being present in.

```
%compares to grid and removes row possibilities
for y=1:r   %LOOP THROUGH GRID IN Y/ROWS
    for x=1:c   %LOOP THROUGH GRID IN X/COLUMNS

        %STORES VALUE IF THERE IS ONE
        if grid(y,x) ~= 0

            B = grid(y,x);

            for x2=1:c  %LOOP THROUGH POSS COLUMNS X
                if poss(y,x2,B) ~= 0 && grid(y,x2) ~= grid(y,x)
                    poss(y, x2, B) = 0;
                end
            end

        end
    end
end
```

Listing 2 - Removal of Duplicate Values in Rows

The third section is very similar to the second section, but removes any possibility of a number in a column being used twice. This is in line with the rules of the problem that state that all number from one to nine can only be used once in each column. This section loops through the grid array and identifies any values that are non-zero, and removes the possibility of that number being used twice in the column it was identified being present in.

```
%compares to grid and removes column possibilities
for y=1:r   %LOOP THROUGH GRID IN Y/ROWS
    for x=1:c   %LOOP THROUGH GRID IN X/COLUMNS

        %STORES VALUE IF THERE IS ONE
        if grid(y,x) ~= 0

            B = grid(y,x);

            for y2=1:c  %LOOP THROUGH POSS COLUMNS X
                if poss(y2,x,B) ~= 0 && grid(y2,x) ~= grid(y,x)
                    poss(y2, x, B) = 0;
                end
            end

        end
    end
end
```

Listing 3 – Removal of Duplicate Values in Columns

The fourth section compares each sub 3x3 array in the Sudoku grid array, and removes the possibility of a number being used twice in every sub grid in the array. Numerous if statements are used to convert between each sub grid with respect to the full Sudoku grid. This is shown graphically in Figure 3.

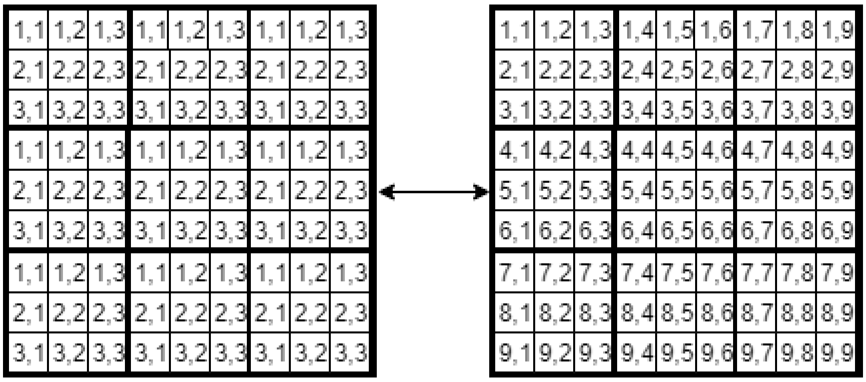

Figure 3 - Array Conversion

```
%Compares possibilities to grid and removes possibilites that are not
%possible in the subgrid
for z = 1:d
    for ssgy = 1:3

        if ssgy ==1
            sgy=1;
        elseif ssgy == 2
            sgy = 4;
        elseif ssgy == 3
            sgy = 7;
        end

        for ssgx = 1:3

            if ssgx == 1
                sgx = 1;
            elseif ssgx == 2
                sgx = 4;
            elseif ssgx == 3
                sgx = 7;
            end

            for gy = 0:2
                for gx = 0:2
                    if grid((sgy+gy),(sgx+gx)) == z

                        for gy2=0:2
                            for gx2=0:2
                                if poss((sgy+gy2),(sgx+gx2),z) == 1 &&
grid((sgy+gy2),(sgx+gx2))~= z
                                    poss(x,y,z) = 0;
                                end
                            end
                        end
                    else
                        break
                    end
                end
            end
        end
    end
end
```

Listing 4 – Sub Grid Comparison and Duplicate Number Removal

2.3 POSSIBILITY TO GRID COMPARISON FUNCTION

Like in the eliminate possibilities function, the Possibility to Grid comparison function has 4 sections to it. Each section identifies any new unique value based on the updated possibilities array, and updates the Sudoku grid array.

The first section identifies any position of the Sudoku grid that contains no value and identifies a unique value in the possibilities array via a depth wise summation. If both of those conditions are true, then the Sudoku grid array is populated with the value identified.

```
[r,c,d] = size(poss);

%if grid summation = 1, the value is stored, identified and implemented
sumpossd = sum(poss,3);

for y = 1:r
    for x = 1:c
        if grid(y,x) == 0 && sumpossd(y,x) == 1
            for z = 1:d
                if poss(y,x,z) == 1
                    grid(y,x) = z;
                end
            end
        else
            break
        end
    end
end
```

Listing 5 – Depth Wise Summation and Unique Value Implemetation

The second section of the Possibility to Grid comparison function, identifies any position of the Sudoku grid that contains no value and identifies if a unique value can be populated in this position with a column wise summation. The same is true for the third section of the Possibility to Grid Comparison function except a row wise summation is performed. This is shown in Listing 6 & Listing 7 respectively.

```
%if grid summation = 1, the value is stored, identified and implemented
sumpossc = sum(poss,2);

for z = 1:d
    for y = 1:r
        if sumpossc(y,1,z) == 1
            for x = 1:c
                if poss(y,x,z) == 1 && grid(y,x) == 0
                    grid(y,x) = z;
                end
            end
        else
            break
        end
    end
end
```

Listing 6 – Column Wise Summation and Unique Value Implementation

```
%Updates grid based on longitudinal summation
%if grid summation = 1, the value is stored, identified and implemented
sumpossr = sum(poss,1);

for z = 1:d
    for x = 1:c
        if sumpossr(1,x,z) == 1
            for y = 1:r
                if poss(y,x,z) == 1 && grid(y,x) == 0
                    grid(y,x) = z;
                end
            end
        else
            break
        end
    end
end
```

Listing 7 – Row Wise summation and Unique Value Implementation

The forth section of the Possibility to Grid comparison function, performs summations over the various 3x3 sub grids in the possibilities array over the 9x9x9 full array of possibilities. If a summation of one is identified and a possibility of one is also identified to be unique in each sub grid, the Sudoku grid at that position takes on that value.

This leads to a converged solution after many iterations as outlined in Section 2.1.

```
%updates grid based on sub grid values
for z = 1:d
    for ssgy = 1:3

        if ssgy ==1
            sgy=1;
        elseif ssgy == 2
            sgy = 4;
        elseif ssgy == 3
            sgy = 7;
        end

        for ssgx = 1:3

            if ssgx == 1
                sgx = 1;
            elseif ssgx == 2
                sgx = 4;
            elseif ssgx == 3
                sgx = 7;
            end

            for gy = 0:2
                for gx = 0:2
                    B = sum(sum(poss(sgy:sgy+2,sgx:sgx+2,z)));

                    if poss((sgy+gy),(sgx+gx),z) == 1 && B == 1
                        grid((sgy+gy),(sgx+gx))=z;
                    end
                end
            end
        end
    end
end
```

Listing 8 – Sub Grid Summation and Unique Value Implementation