# SOLVING SUDOKU WITH MATLAB

Raluca Marinescu, Andrea Garcia, Ivan Castro, Eduard Enoiu

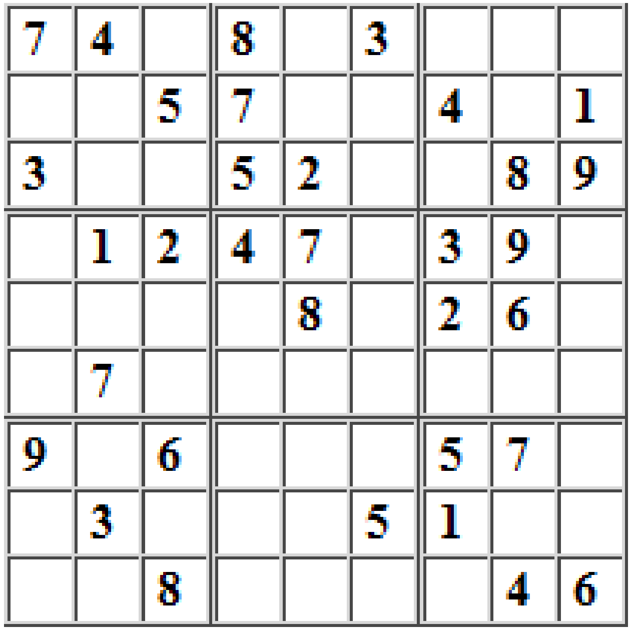
Mälardalen University, Västerås, 28.02.2011

# Background Of Sudoku

- From the Japanese: ”SU”, means ”number” and ”DOKU”, means ”single”.
- It is based on the concept of ***Latin Squares*** (similar to magic squares) introduced by Leonhard Euler in the 18th century.
- The board is composed by a **9x9 grid**. The whole grid is divided into sub-squares containing a **3x3** grid each.
- The purpose of the game is to insert numbers (1-9) in the board ***without repeating a number in each row, column or sub-square***.

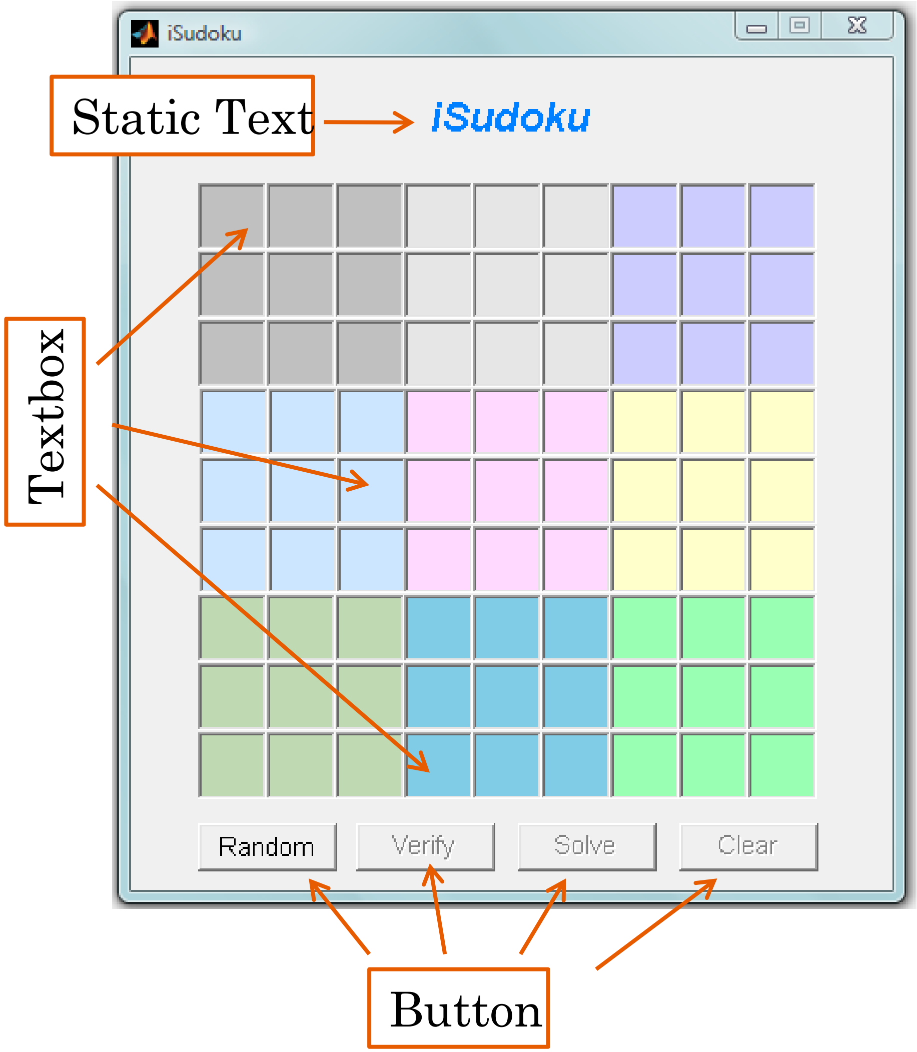
# Problem Description

- Design and Implement a Sudoku Puzzle Solver using Matlab.



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

# Graphical User Interface

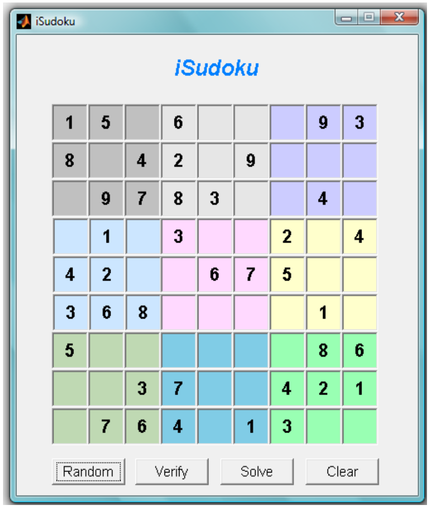


- Implemented using Matlab’s GUI Design Environment (**GUIDE**).
- Used **drag & drop** components to create the layout.
- Each component has:
  - A list of properties that can be edited (color, size, position, etc)
  - **Callback functions** to model its behavior.

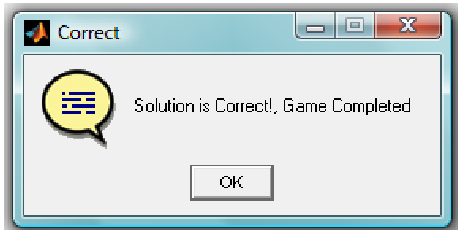
# Graphical User Interface

Functionality is achieved using **four buttons**:

- **Random**
  - Creates and displays a random game
- **Solve**
  - Solves the game and displays the solution
- **Verify**
  - Verifies the correctness of the game
- **Clear**
  - Clears the board



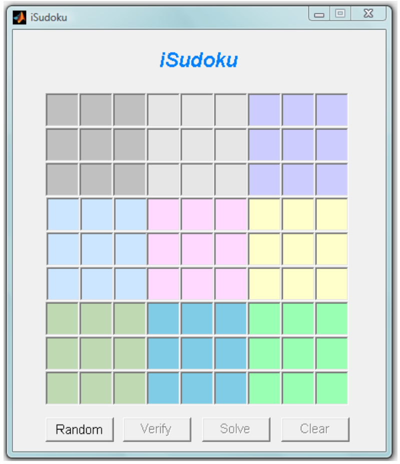
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 5 | | 6 | | | | 9 | 3 |
| 8 | | 4 | 2 | | 9 | | | |
| | 9 | 7 | 8 | 3 | | | 4 | |
| | 1 | | 3 | | | 2 | | 4 |
| 4 | 2 | | | 6 | 7 | 5 | | |
| 3 | 6 | 8 | | | | | 1 | |
| 5 | | | | | | | 8 | 6 |
| | | 3 | 7 | | | 4 | 2 | 1 |
| | 7 | 6 | 4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



Solution is Correct!, Game Completed

# Graphical User Interface

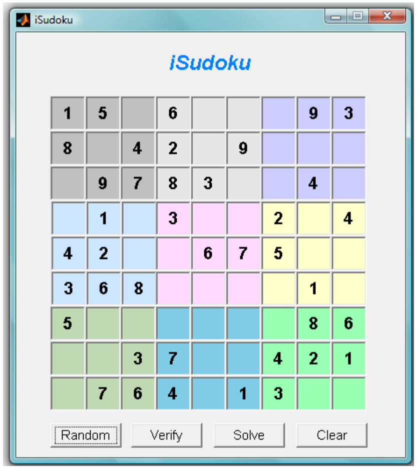
Clear



1

Random →

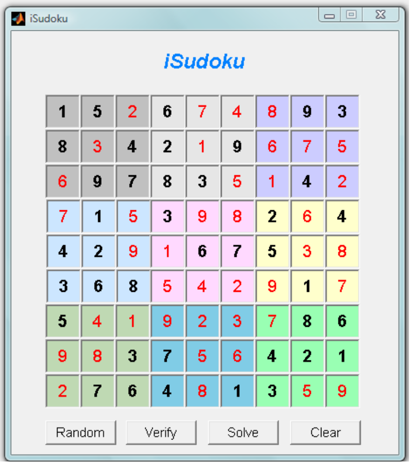
2



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

Solve →

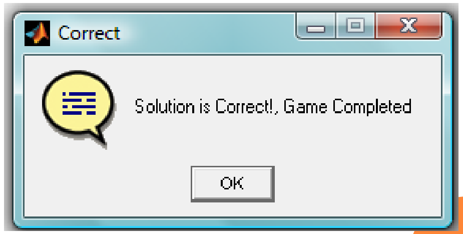
3



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

4

Verify



Solution is Correct!, Game Completed

Simplified program operation:

1. Games are loaded from an Excel database.
2. Random game is displayed.
3. Game matrix is sent to the sudoku algorithm.
4. Solved matrix is received and displayed.

# Graphical User Interface

- Example of a callback function (Clear button):

Function that is called after the event (mouse click)

```
% --- Executes on button press in ClearBtn.
function ClearBtn_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)

%Clear the board
for rowInd = 1:9
    for colInd = 1:9
        cName = ['c' num2str(rowInd) num2str(colInd)];
        cValue = '';
        expr = ['set(handles.' cName ', ''String'', ''' cValue ''',...
                ''FontWeight'', ''normal'', ''Enable'', ''on'')'];
        eval(expr);
    end
end

set(handles.SolveBtn, 'Enable', 'off');
set(handles.verifyBtn, 'Enable', 'off');
```

Puts an empty string on every textbox component in the board

Disables the buttons

# SOLVING SUDOKU WITH MATLAB

- ALGORITHM
  - The key internal function is:

    *iSudokuALG(A)*

# SOLVING SUDOKU WITH MATLAB

- ALGORITHM
  - The key internal function is:

    *iSudokuALG(A)*
  - Steps:
    - *(1) Find all the possible values for the empty cell*

# SOLVING SUDOKU WITH MATLAB

- ALGORITHM
  - The key internal function is:

    *iSudokuALG(A)*

  - Steps:
    - *(1) Find all the possible values for the empty cell*

```
for i=1:9
    for j=1:9
        if (A(i,j)+B(i,j))==0
            possible=[];
            for k=1:9
                B(i,j)=k;
                flag2=verific(A+B);
                if flag2==0
                    possible=[possible k];
                end
            end
            B(i,j)=0;
            if length(possible)==1
                B(i,j)=possible;
            end
        end
    end
end
```

Empty cell found

Look for all possible values

# SOLVING SUDOKU WITH MATLAB

- ALGORITHM
  - The key internal function is:

    *iSudokuALG(A)*

  - Steps:
    - *(2) if the cell has only one possible value, fill it*

```
for i=1:9
    for j=1:9
        if (A(i,j)+B(i,j))==0
            possible=[];
            for k=1:9
                B(i,j)=k;
                flag2=verific(A+B);
                if flag2==0
                    possible=[possible k];
                end
                end
                B(i,j)=0;
                if length(possible)==1
                    B(i,j)=possible;
                end
            end
        end
    end
```

Fill the possible value

# SOLVING SUDOKU WITH MATLAB

- ALGORITHM
  - The key internal function is:

    *iSudokuALG(A)*
  - Steps:
    - *(1) Find all the possible values for the empty cell*
    - *(2) if the cell has only one possible value, fill it*
    - *(3) If  all the cells have more than one possible value we fill in a tentative value for one cell*
    - *(4) Verify the puzzle:*

      *function [val]=verific(A)*

# SOLVING SUDOKU WITH MATLAB

- ALGORITHM EXAMPLE
  - We assume for simplification in this example a simpler 4 by 4 grid with 2 by 2 blocks

# SOLVING SUDOKU WITH MATLAB

- ALGORITHM EXAMPLE

| | | | |
|---|---|---|---|
| | | | 1 |
| | | 2 | |
| | 3 | | |
| 4 | | | |

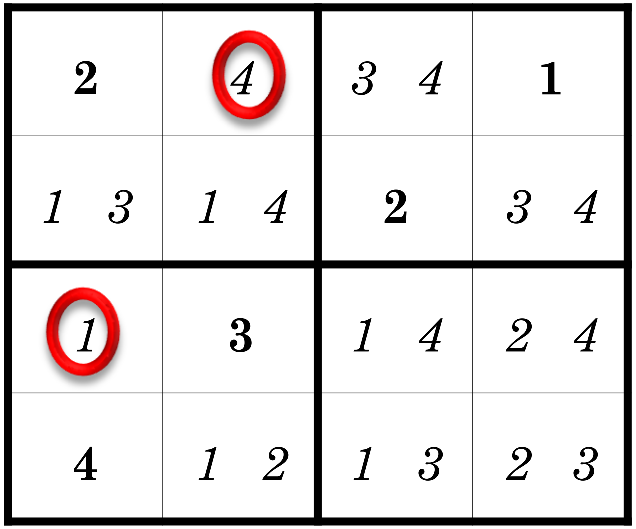
# SOLVING SUDOKU WITH MATLAB

- ALGORITHM EXAMPLE

| | | | |
|---|---|---|---|
| [✓ *2*] *3* | *2 4* | *3 4* | **1** |
| *1 3* | *1 4* | **2** | *3 4* |
| *1 2* | **3** | *1 4* | *2 4* |
| **4** | *1 2* | *1 3* | *2 3* |

# SOLVING SUDOKU WITH MATLAB

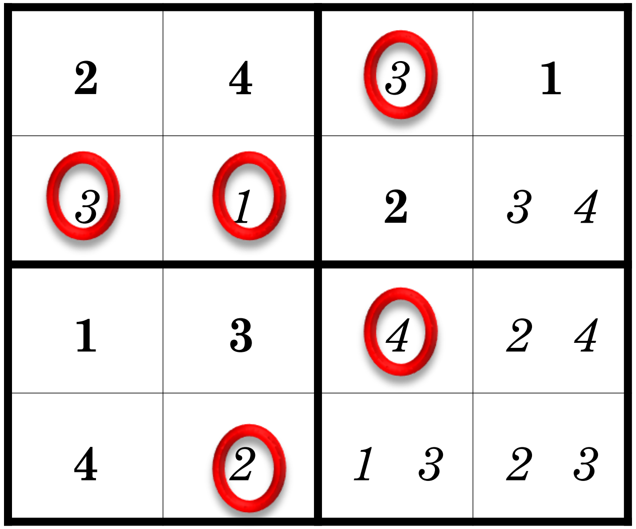
- ALGORITHM EXAMPLE



| | | | |
|---|---|---|---|
| **2** | *4* | *3 4* | **1** |
| *1 3* | *1 4* | **2** | *3 4* |
| *1* | **3** | *1 4* | *2 4* |
| **4** | *1 2* | *1 3* | *2 3* |

# SOLVING SUDOKU WITH MATLAB

- ALGORITHM EXAMPLE



| | | | |
|---|---|---|---|
| **2** | **4** | *3* | **1** |
| *3* | *1* | **2** | *3 4* |
| **1** | **3** | *4* | *2 4* |
| **4** | *2* | *1 3* | *2 3* |

# SOLVING SUDOKU WITH MATLAB

- ALGORITHM EXAMPLE

| | | | |
|---|---|---|---|
| 2 | 4 | 3 | 1 |
| 3 | 1 | 2 | *4* |
| 1 | 3 | 4 | *2* |
| 4 | 2 | *1* | *3* |

# SOLVING SUDOKU WITH MATLAB

- ALGORITHM EXAMPLE

| | | | |
|---|---|---|---|
| 2 | 4 | 3 | 1 |
| 3 | 1 | 2 | 4 |
| 1 | 3 | 4 | 2 |
| 4 | 2 | 1 | 3 |

# Conclusions

- There is a large number of possible algorithms to solve Sudoku puzzles, from the brute force algorithm to stochastic search algorithms.
- Finding a suitable algorithm to solve any particular Sudoku game proved to be very difficult.
- Using a GUI helped the developers to generate Sudoku games and verify solutions in a simple and quick way.
- The obtained results using the implemented Sudoku solver have been successful, for this reason we don't foresee any major changes to our solution.

# Demo