Manual and Technical Document

By *LCC Corps*

March 12, 2011

Table of Contents

Part 1: User manual

1. Requirement

In order to run Kakuro solver, the computer need to be installed with **MATLAB 2008a** and above (though previous version may still work) as well as latest version of **YALMIP**, available at http://users.isy.liu.se/johanl/yalmip/. Other mixed integer linear programming (MILP) solver such as lpsolve, cplex, etc can be used instead of MATLAB built-in linprog.

2. Kakuro solver GUI

The GUI provided is aimed to imitate the actual Kakuro problem as close as possible. The key GUI features shown in **Error! Reference source not found.**are as follows.

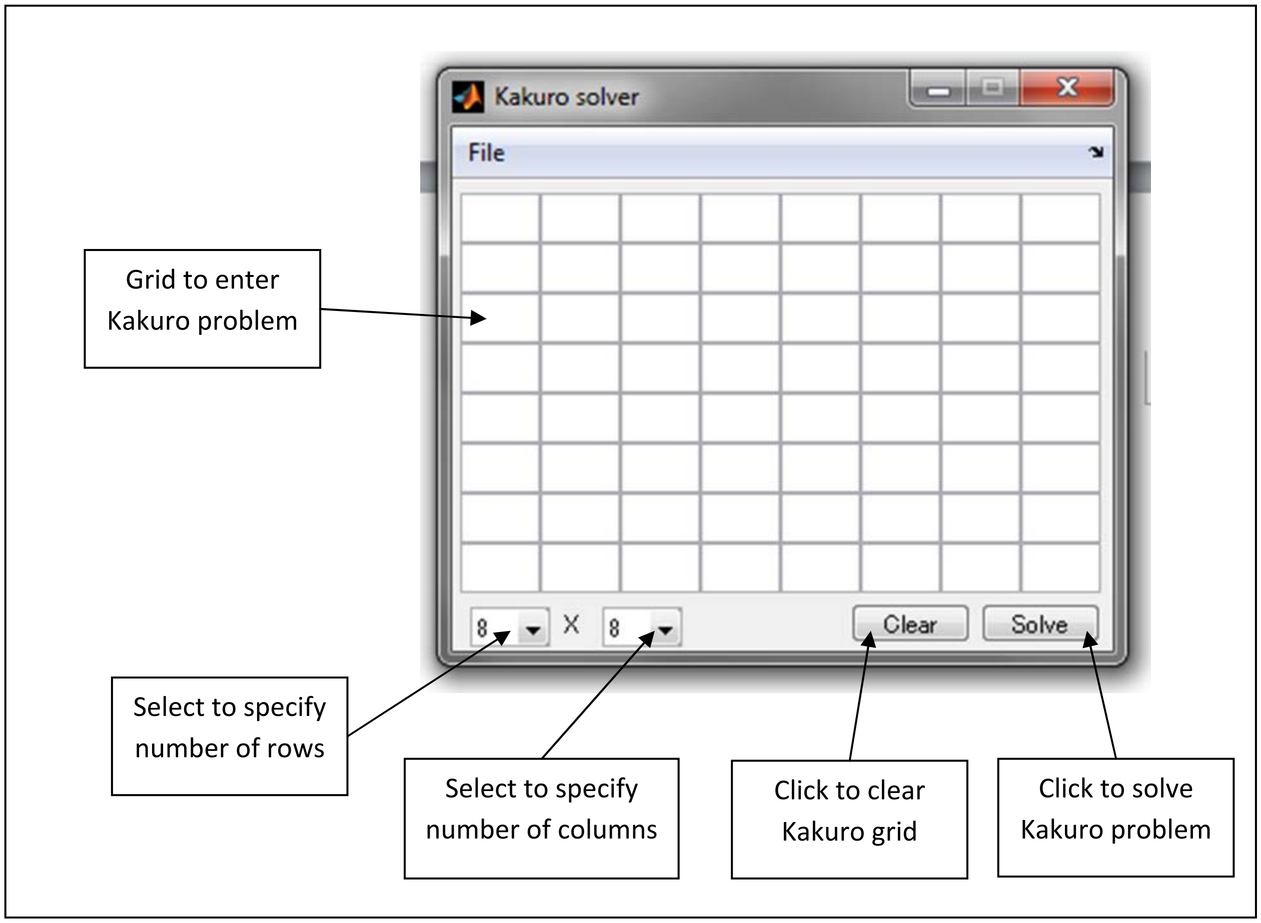

Kakuro solver
File
8
X
8
Clear
Solve
Grid to enter Kakuro problem
Select to specify number of rows
Select to specify number of columns
Click to clear Kakuro grid
Click to solve Kakuro problem

Figure 1 - Kakuro solver GUI features.

1. **Main Kakuro grid**

   The grid serves as a place for user to enter a Kakuro problem. The grid is represented as cloas as possible to the actual Kakuro problem. Each grid also contains simple input check to detect errors in user inputs. An erroneous input is marked by the grid cell in red colour.Valid input will be marked by the grid cell in cyan. When a Kakuro problem is solved, the solution will be presented into this grid as well, as a form of completed Kakuro problem.

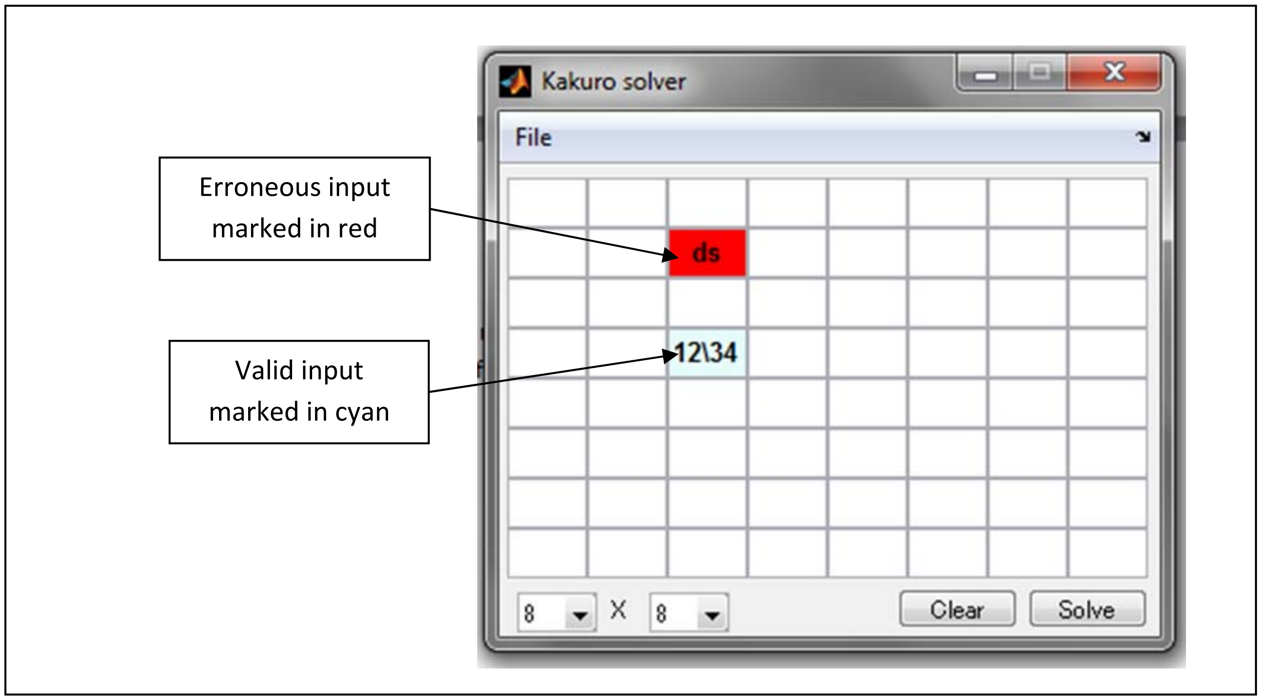


Figure 2 - Input check feature.

2. **Grid size drop-down menu**
   The two drop-down menu can be used to change the size of the Kakuro grid. The first drop-down menu changes the number of rows, while the second one for the number of columns. Adjusting the Kakuro grid will automatically clear the existing Kakuro problem. Currently, the solver can solved up to 16 X 16 grid of Kakuro problem, though this limitation can be easily changed.

3. **“Clear” button**
   The “Clear” button is provided to clear the Kakuro grid for new problem or in case the user decided to reenter a Kakuro problem.

4. **“Solve” button**
   Clicking the “Solve” button will initiate the solving of the given Kakuro problem. Note that the solving process may take a while to complete.

3. Instruction by example

As an easy and straightforward instruction on how to use Kakuro solver, a demonstration will be done using a Kakuro example problem shown in Figure 3.

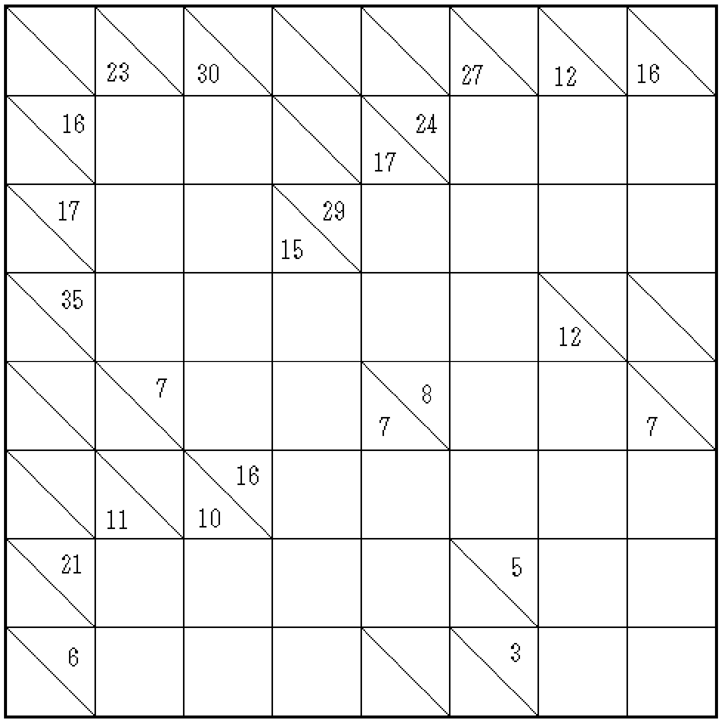


Figure 3 - A Kakuro example problem.

Step 1: Select the two drop-down menu to have a 8 X 8 Kakuro grid.

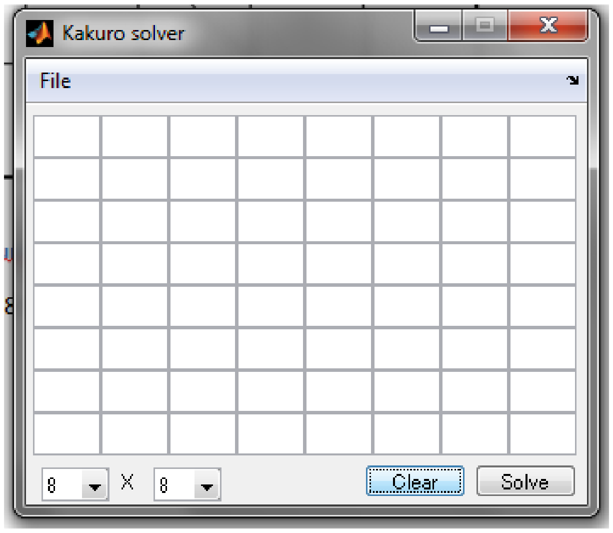
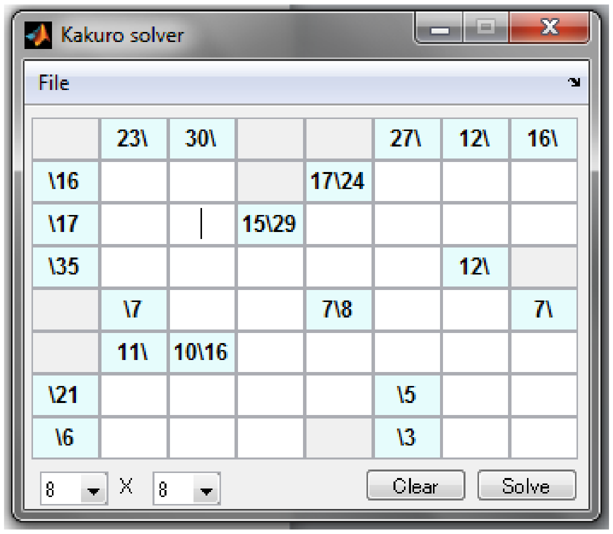


Figure 4 - Step 1: select desired grid size.

Step 2: Enter the Kakuro problem into the Kakuro grid. Note that the input follows certain format and representation as follows.

a. Right click a cell to toggle between unused and available cell. An unused cell will be coloured in light gray, whereas an available cell in white. Note that all unused cell in the actual Kakuro problem must be properly toggled in the Kakuro grid in the GUI.
b. A vertical sum, e.g. 23, from the actual Kakuro problem is entered either as '23' or '23\' without the quotes.
c. A horizontal sum, e.g. 16, from the actual Kakuro problem is entered either as '\16' without the quotes.
d. A combination of both vertical and horizontal sum, e.g. 15 (vertical sum) and 29 (horizontal sum) is entered as '15\29' without the quotes.
e. Cells which are to be filled with the answer are left blank.

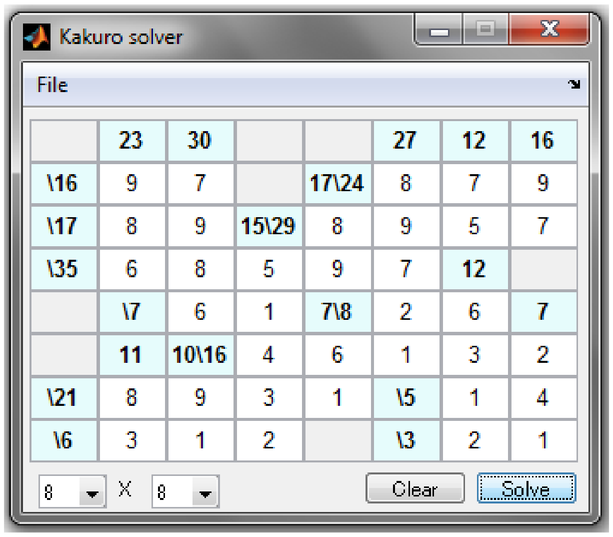
The Kakuro example problem from Figure 3 that has been entered into the Kakuro solver GUI is shown in Figure 5. Please compare Figure 3 and Figure 5 to understand the five input rules stated in this step (above).



	23\	30\			27\	12\	16\
\16				17\24			
\17			15\29				
\35						12\	
	\7			7\8			7\
	11\	10\16					
\21					\5		
\6					\3		

Figure 5 - Step 2: Enter Kakuro problem.

Step 3: Click the "Solve" button to begin the solving process. Note that this may take a while to complete depending on the computer. Once solved, the solution will be displayed in the Kakuro grid as a completed Kakuro problem as shown in Figure 6.



	23	30			27	12	16
\16	9	7		17\24	8	7	9
\17	8	9	15\29	8	9	5	7
\35	6	8	5	9	7	12	
	\7	6	1	7\8	2	6	7
	11	10\16	4	6	1	3	2
\21	8	9	3	1	\5	1	4
\6	3	1	2		\3	2	1

Figure 6 - Step 3: Returned Kakuro solution.

4. Known Issues

Although simple input check has been implemented, it does not capture all the input errors. Errors such as infeasible Kakuro problems and other logic errors may not be captured and the user needs to ensure the feasibility and correctness of the specified Kakuro problem.

The Kakuro solver has been implemented and tested on MATLAB 2008a on Windows XP 32-bit. Hence, it is more likely to run properly on such machine. In case the Kakuro solver fails to work on some version of YALMIP release, please update the YALMIP package to the latest version.

Part 2: Technical information

1. Kakuro modelling in MATLAB

Kakuro problem is modelled using a 2D complex number array in MATLAB. The unused cell is represented by -1 (negative one), vertical sum represented by the real part of the cell value, horizontal part by the imaginary part of the cell value, and empty cell (where the solution is to be filled) by 0 (zero).

As an example, the Kakuro example problem in Figure 3 is represented as shown in Table 1 in MATLAB. Note that Table 1 is indeed a 2D complex number array in MATLAB and is not an actual table.

Table 1 - MATLAB representation of Kakuro example problem inFigure 3.

-1	23+j0	30+j0	-1	-1	27+j0	12+j0	16+j0
0+j16	0	0	-1	17+j24	0	0	0
0+j17	0	0	15+j29	0	0	0	0
0+j35	0	0	0	0	0	12+j0	-1
-1	0+j7	0	0	7+j8	0	0	7+j0
-1	11+j0	10+j16	0	0	0	0	0
0+j21	0	0	0	0	0+j51	0	0
0+j6	0	0	0	-1	0+j3	0	0

Note: For cell containing only the real number, representing only the vertical sum (and no horizontal sum, e.g. cell (1, 2)), we add a small imaginary number (j*ε) to it in order to make it a complex number, to distinguish it from the real number used to represent variable indices (discussed in next section).

2. Conversion of Kakuro problem into MILP constraint

Before we convert the Kakuro problem into mixed integer linear programming (MILP) constraints, the zeros in Table 1 are first filled with positive integer starting from 1 to represent the variable (solution to Kakuro problem to be found) indices. The order on how the cells are filled is not important and Table 2 shows one of the possibilities.

Table 2 - Kakuro problem array in MATLAB filled with variable indices.

-1	23+j0	30+j0	-1	-1	27+j0	12+j0	16+j0
0+j16	14	10	-1	17+j24	21	26	32
0+j17	19	11	15+j29	17	22	27	33
0+j35	25	8	12	18	23	12+j0	-1
-1	0+j7	9	13	7+j8	24	28	7+j0
-1	11+j0	**10+j16**	1	2	3	4	5
0+j21	29	6	15	20	0+j51	30	35
0+j6	34	7	16	-1	0+j3	31	36

Next, we convert the Kakuro problem into MILP constraints. For simplicity, we only consider one example, as marked by the value in red. We have

$y_1 = x_6 + x_7$ (for vertical sum)

$y_2 = x_1 + x_2 + x_3 + x_4 + x_5$ (for horizontal sum)

where we can rewrite the constraints in a compact matrix form

$$\mathbf{Ax} = \mathbf{y}$$

with

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & a_{1,6}=1 & a_{1,7}=1 & \dots \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix}$$

where the matrix **A**contains one for the element of x that contributes to the sum. Other constraints are $1 \le x_i \le 9$, $\forall i$ and the values of x within each particular sum cannot be repeating.

The algorithm for converting the Kakuro problem into MILP problem is as follows.

```
Search the Kakuro problem matrix for complex number.
Initialise matrix A and vector y to zero.
For each complex number,
    If floor(real part) > 0,
        - Increment constraint counter.
        - Get all the positive, real values below the complex number until either -1, a complex
          number or end of matrix is encountered.
        - Set the elements in matrix A (with row index = constraint counter, and column indices
          = the positive, real values obtained above) to ones.
        - Set constraint such that variables x indexed by the indices CANNOT be repeating/same
          numbers.
    End if
    If floor(imaginary part) > 0,
        - Increment constraint counter.
        - Get all the positive, real values to the right of the complex number until either -1, a
          complex number or end of matrix is encountered.
        - Set the elements in matrix A (with row index = constraint counter, and column indices
          = the positive, real values obtained above) to ones.
        - Set constraint such that variables x indexed by the indices CANNOT be repeating/same
          numbers.
    End if
End for
Set constraint $1 \le x_i \le 9$, $\forall i$.
```

Solve using MILP.

Put the solution back into Kakuro problem matrix.

Search the Kakuro problem matrix for positive, real numbers.

Set value[r, c] = **x**[value[r, c]], where r and c are the row and column indices corresponding to the elements with positive, real numbers, and **x** is the variable vector containing the solution.

3. Formulation using binary model

Formulation in binary model is less intuitive. Here, we declared the variables **X** to be optimized/found to be a 2D matrix, with number of rows equals to number of variables, and number of column equals to number of allowed values (e.g. if $1 \le x_i \le 9$, $\forall i$, then number of columns is 9). Here, the row indices indicate the variable indices, and the column indices indicate the value assigned to that variable. For example,

$$\begin{array}{c|cccc|} & 1 & 2 & \dots & 9 \\ x_1 & 1 & 0 & \dots & 0 \\ x_2 & 0 & 0 & \dots & 1 \\ \dots & \dots & \dots & \dots & \dots \\ x_N & 0 & 1 & \dots & 0 \end{array}$$

means that $x_1 = 1$, $x_2 = 9$, and $x_N = 2$.For each of the sum constraints, the formulation for binary model becomes (we only show for 2 examples),

Sum constraint #1: $y_1 = x_6 + x_7$ → $$y_1 = sum\left(\begin{array}{c|cccc|} & 1 & 2 & \dots & 9 \\ x_6 & ? & ? & ? & ? \\ x_7 & ? & ? & ? & ? \end{array} \times \begin{bmatrix} 1 \\ 2 \\ \dots \\ 9 \end{bmatrix} \right)$$

Constraint for no repeating value in a sum #1: $$0 \le sum\left(\begin{array}{c|cccc|} & 1 & 2 & \dots & 9 \\ x_6 & ? & ? & ? & ? \\ x_7 & ? & ? & ? & ? \end{array}, 1 \right) \le 1$$

Sum constraint #2: $y_2 = x_1 + x_2 + x_3 + x_4 + x_5$ → $$y_2 = sum\left(\begin{array}{c c} & \begin{array}{cccc} 1 & 2 & \ldots & 9 \end{array} \\ \begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{array} & \begin{bmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix} \times \begin{bmatrix} 1 \\ 2 \\ \ldots \\ 9 \end{bmatrix} \end{array}\right)$$

Constraint for no repeating value in a sum #1: $$0 \le sum\left(\begin{array}{c c} & \begin{array}{cccc} 1 & 2 & \ldots & 9 \end{array} \\ \begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{array} & \begin{bmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix} \end{array}, 1\right) \le 1$$

The last constraint for $1 \le x_i \le 9$, $\forall i$ becomes sum(**X**, 2) = 1.

Reference

[1] *YALMIP : A Toolbox for Modeling and Optimization in MATLAB*. J. Löfberg. In Proceedings of the CACSD Conference, Taipei, Taiwan, 2004.