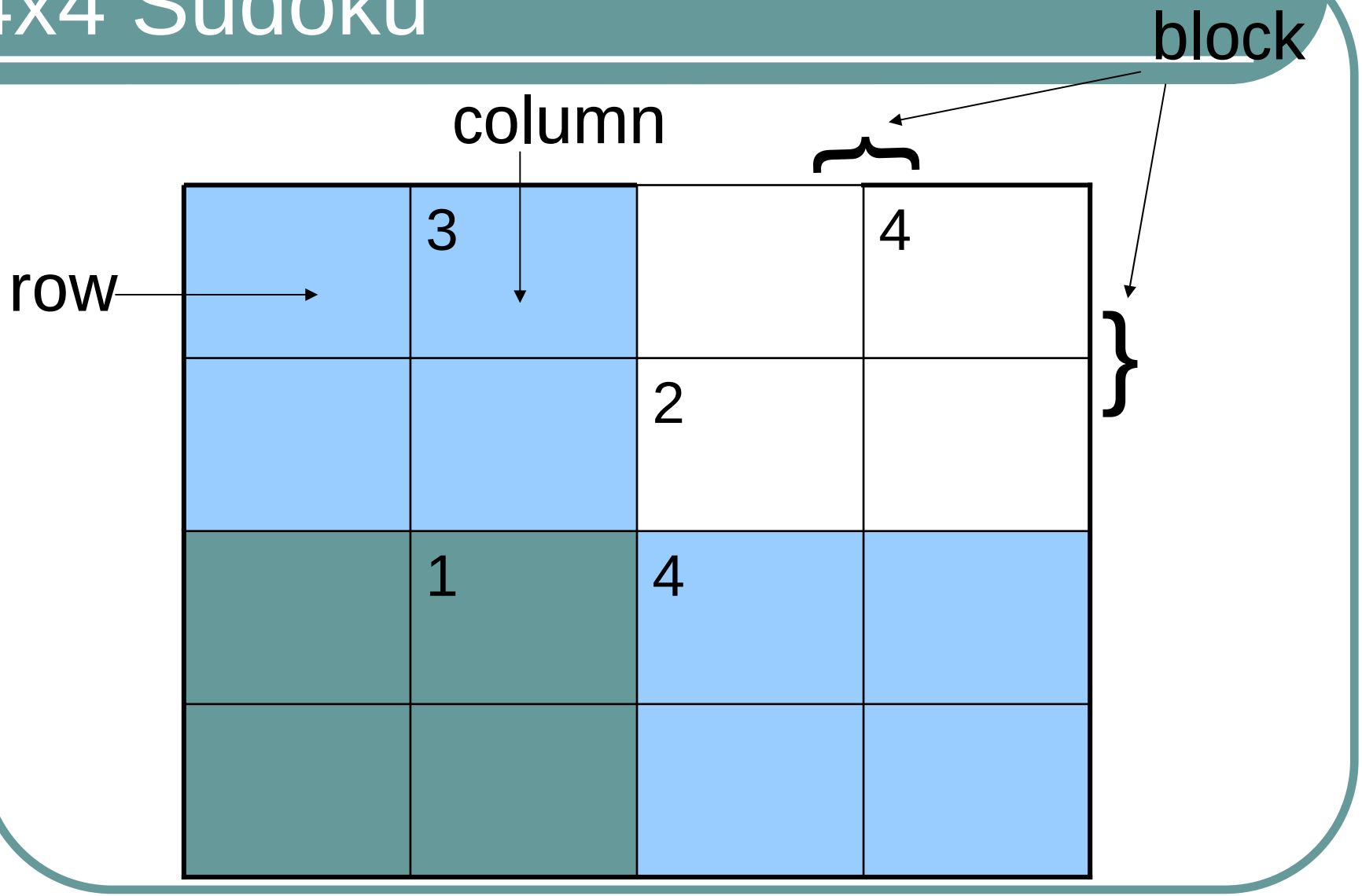
Lecture 8II SUDOKU PUZZLE

- SUDOKU
  - New
  - Play
  - Check

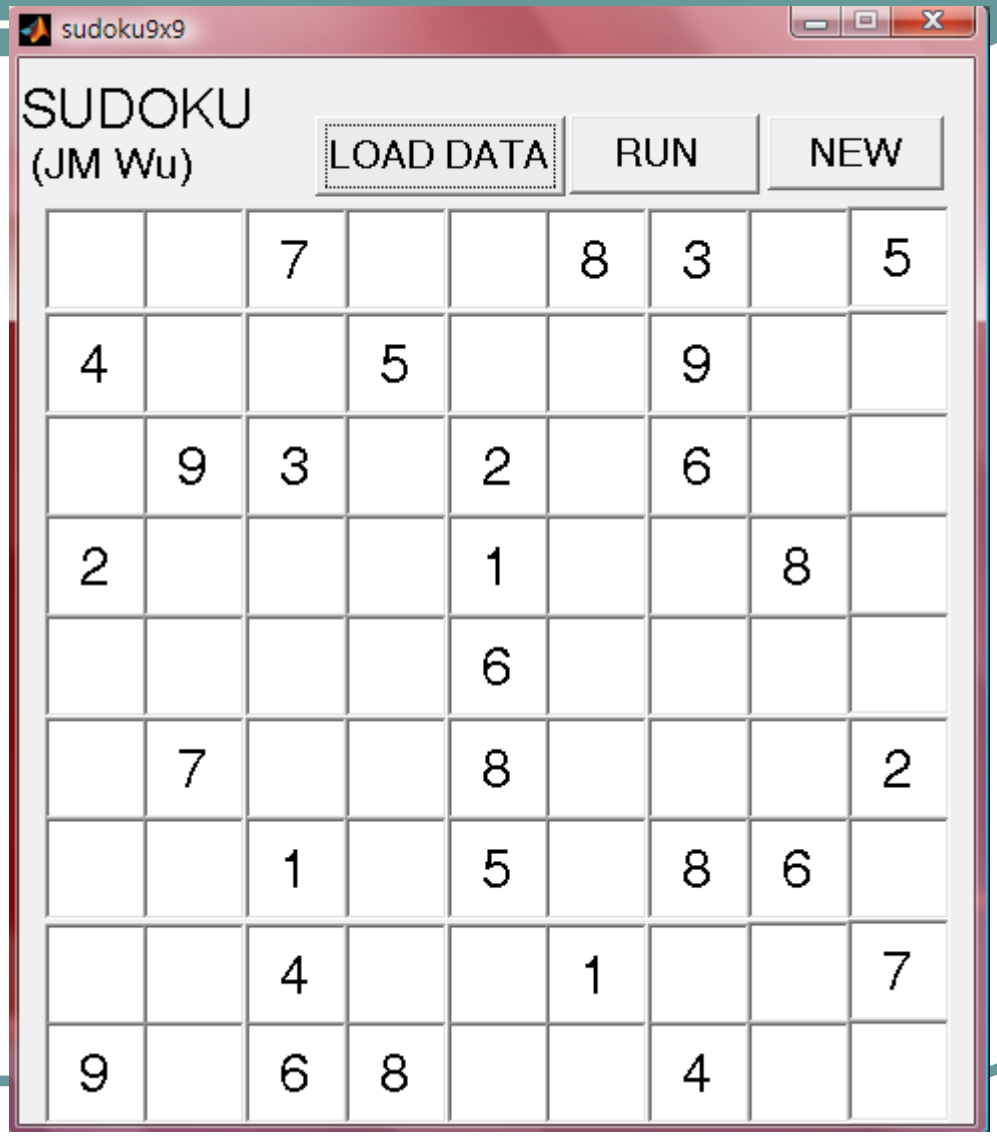
4x4 Sudoku



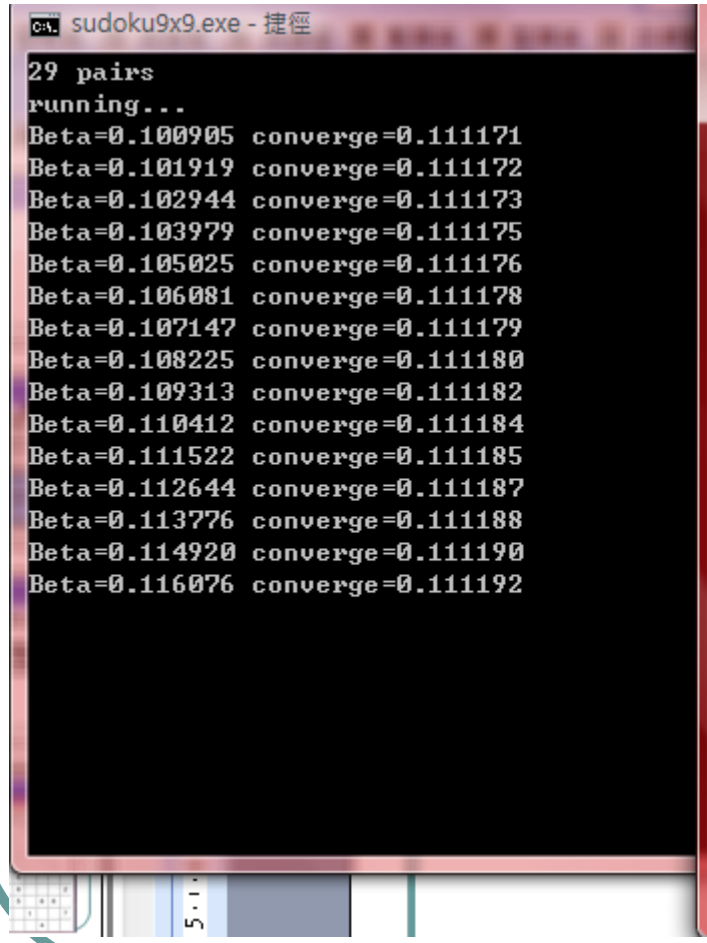
Sudoku Puzzle

- Numbers in the puzzle belong {1,2,3,4}
- Constraints
  - Each column must contain four different numbers
  - Each row must be filled with four different numbers
  - Each block must be filled with four different numbers

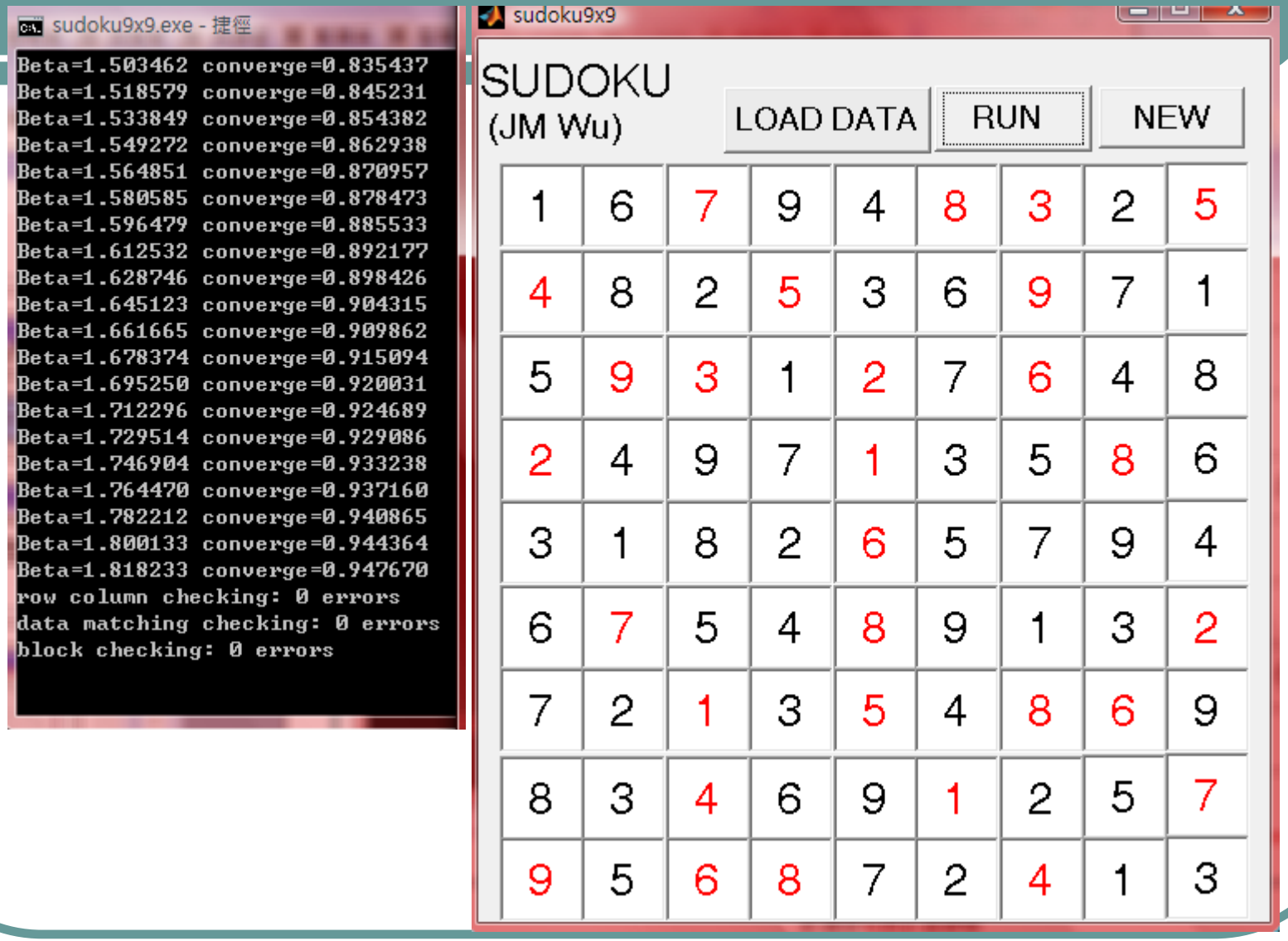
9-by-9 Sudoku


sudoku9x9
SUDOKU
(JM Wu)
LOAD DATA
RUN
NEW
7 8 3 5
4 5 9
9 3 2 6
2 1 8
6
7 8 2
1 5 8 6
4 1 7
9 6 8 4

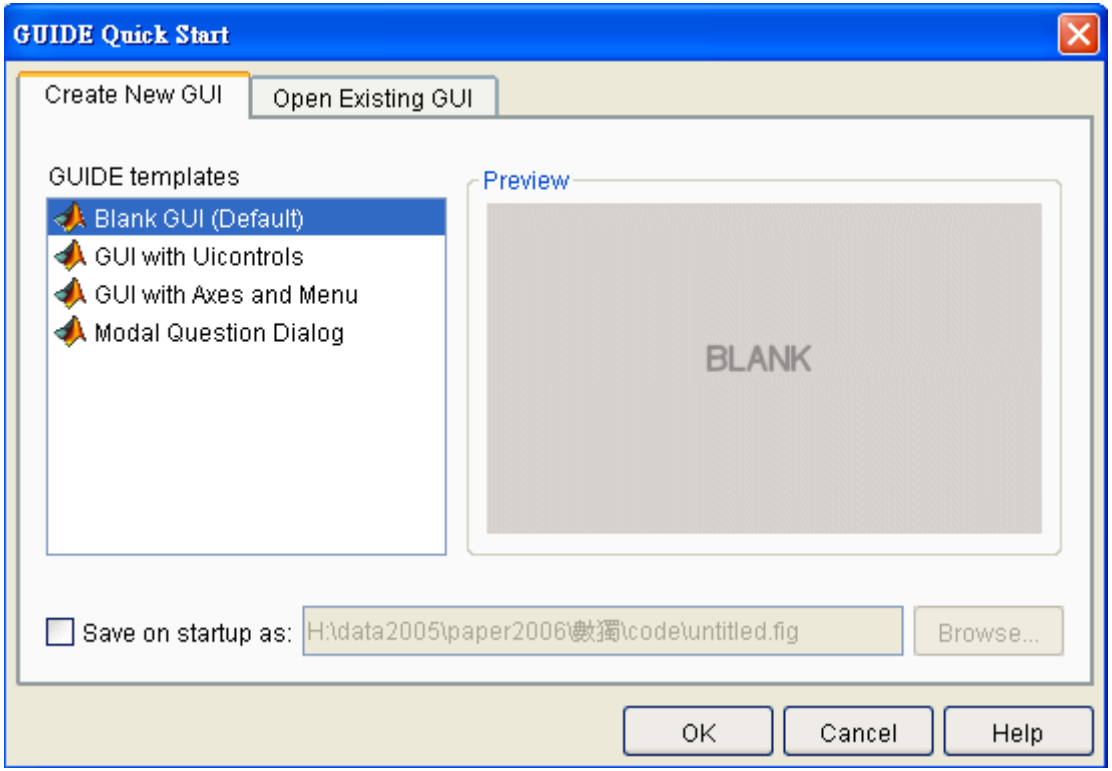
Neural Computation



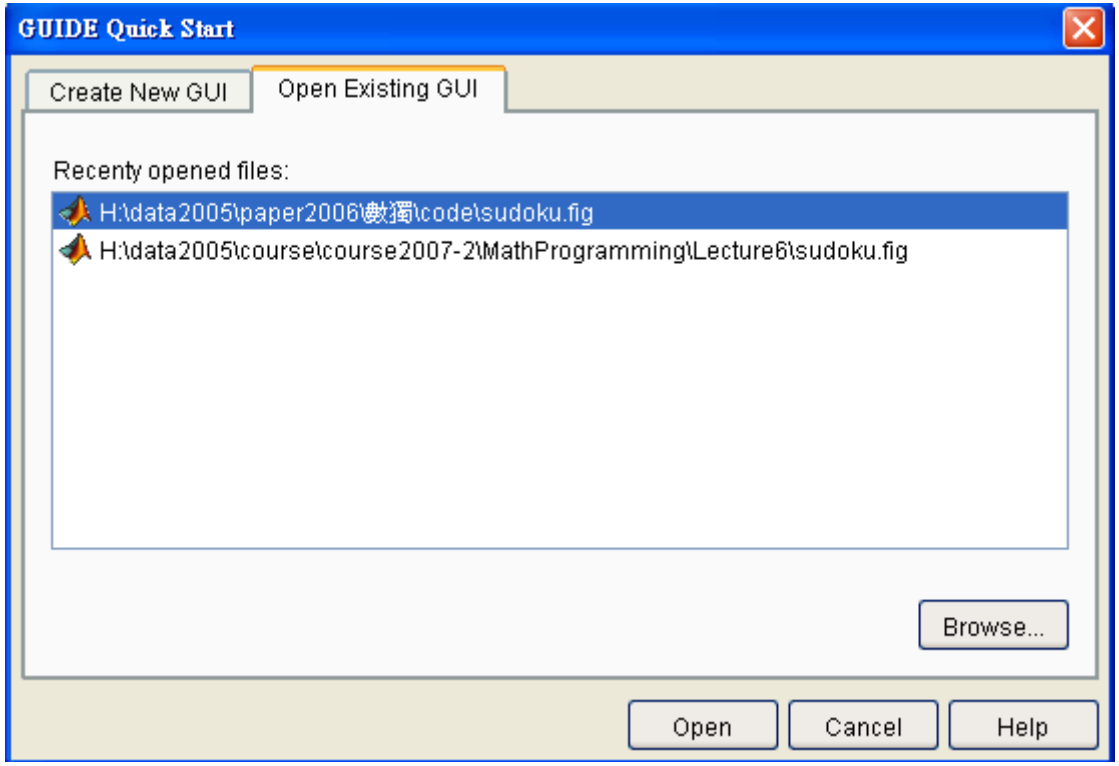
		7			8	3		5
4			5			9		
	9	3		2		6		
2				1			8	
				6				
	7			8				2
		1		5		8	6	
		4			1			7
9		6	8			4		


sudoku9x9.exe - 捷徑
Beta=1.503462 converge=0.835437
Beta=1.518579 converge=0.845231
Beta=1.533849 converge=0.854382
Beta=1.549272 converge=0.862938
Beta=1.564851 converge=0.870957
Beta=1.580585 converge=0.878473
Beta=1.596479 converge=0.885533
Beta=1.612532 converge=0.892177
Beta=1.628746 converge=0.898426
Beta=1.645123 converge=0.904315
Beta=1.661665 converge=0.909862
Beta=1.678374 converge=0.915094
Beta=1.695250 converge=0.920031
Beta=1.712296 converge=0.924689
Beta=1.729514 converge=0.929086
Beta=1.746904 converge=0.933238
Beta=1.764470 converge=0.937160
Beta=1.782212 converge=0.940865
Beta=1.800133 converge=0.944364
Beta=1.818233 converge=0.947670
row column checking: 0 errors
data matching checking: 0 errors
block checking: 0 errors
sudoku9x9
SUDOKU
(JM Wu)
LOAD DATA
RUN
NEW
1 6 7 9 4 8 3 2 5
4 8 2 5 3 6 9 7 1
5 9 3 1 2 7 6 4 8
2 4 9 7 1 3 5 8 6
3 1 8 2 6 5 7 9 4
6 7 5 4 8 9 1 3 2
7 2 1 3 5 4 8 6 9
8 3 4 6 9 1 2 5 7
9 5 6 8 7 2 4 1 3

GUI Design


GUIDE Quick Start
Create New GUI
Open Existing GUI
GUIDE templates
Blank GUI (Default)
GUI with Uicontrols
GUI with Axes and Menu
Modal Question Dialog
Preview
BLANK
Save on startup as:
H:\data2005\paper2006\數獨\code\untitled.fig
Browse...
OK
Cancel
Help

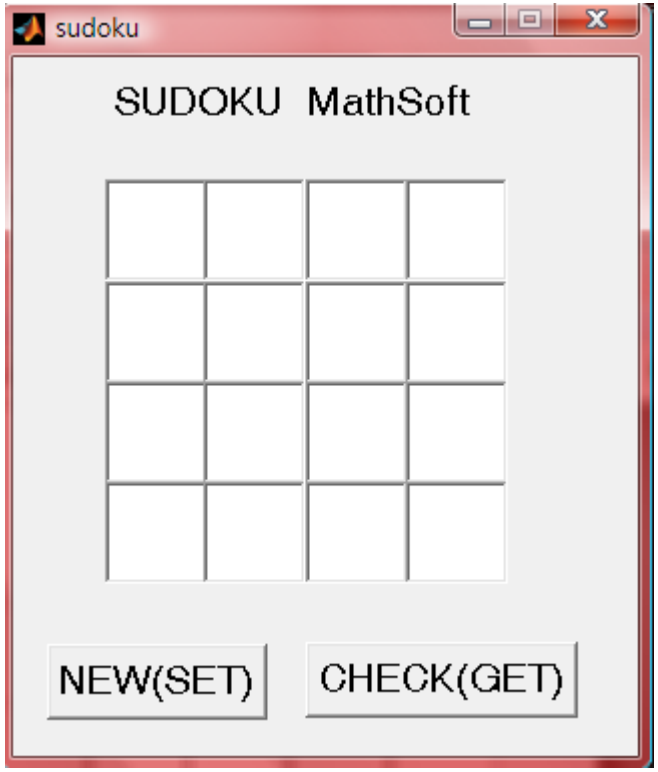
Open


GUIDE Quick Start
Create New GUI
Open Existing GUI
Recenty opened files:
H:\data2005\paper2006\數獨\code\sudoku.fig
H:\data2005\course\course2007-2\MathProgramming\Lecture6\sudoku.fig
Browse...
Open
Cancel
Help

A puzzle

sudoku.fig

Sudoku.m



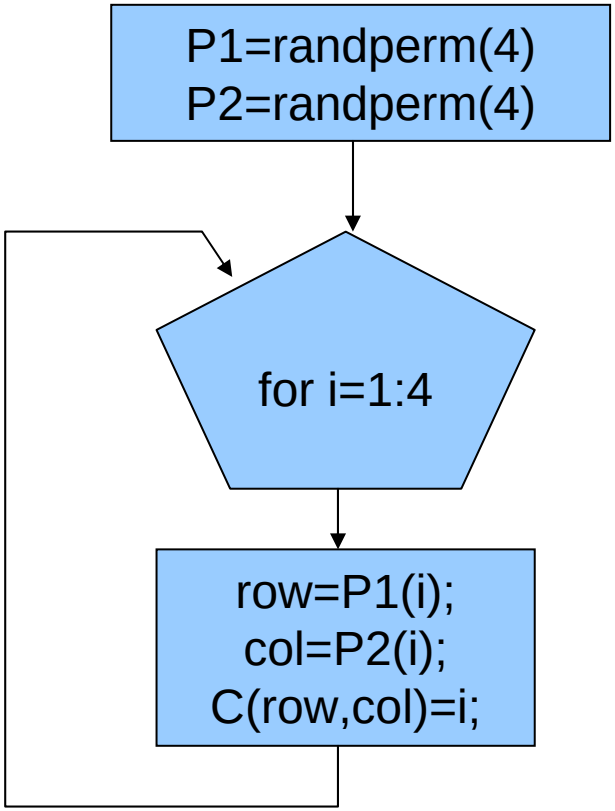
Random permutation

```
>> p1=randperm(4)

p1 =

    4    1    2    3
```

Flow Chart: New

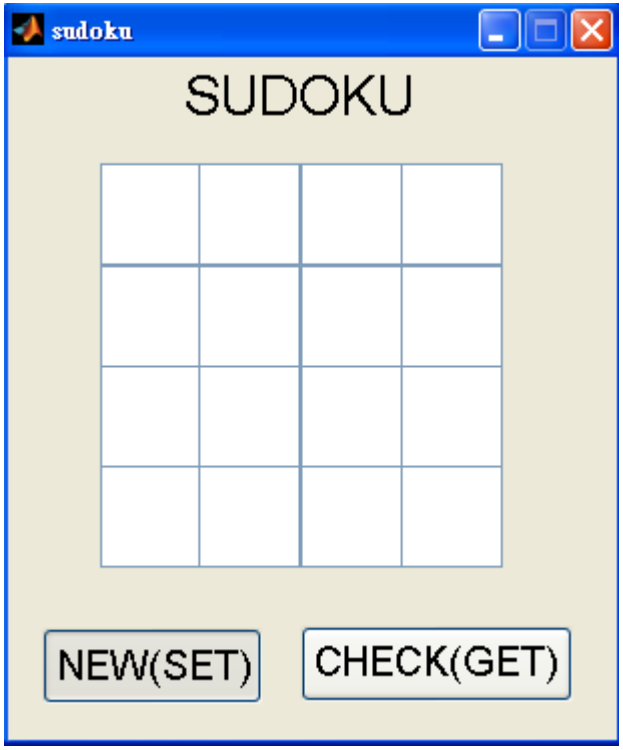


Procedure : New

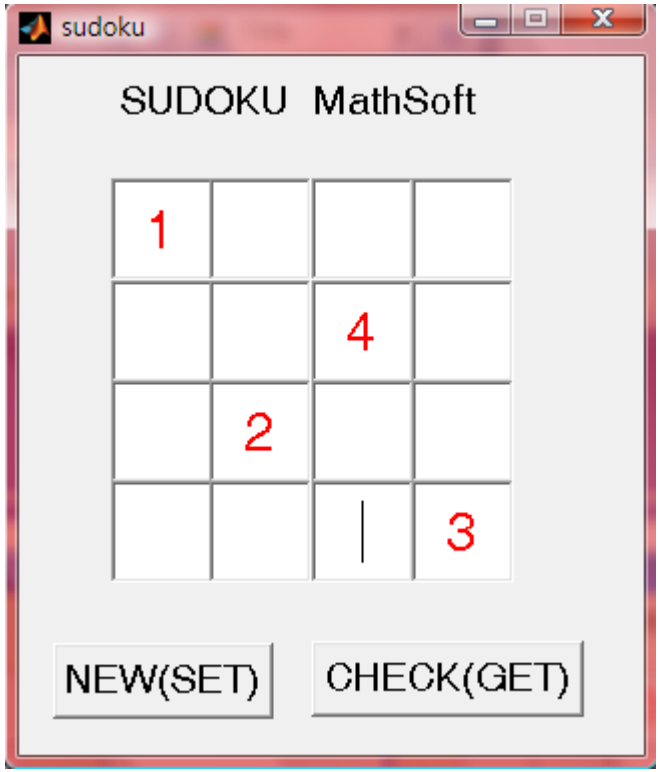
- Use randperm to generate two random permutations, p and q
- for i=1:4
  - j = p(i)
  - d = q(i)
  - C(i,j) = d
- Return C

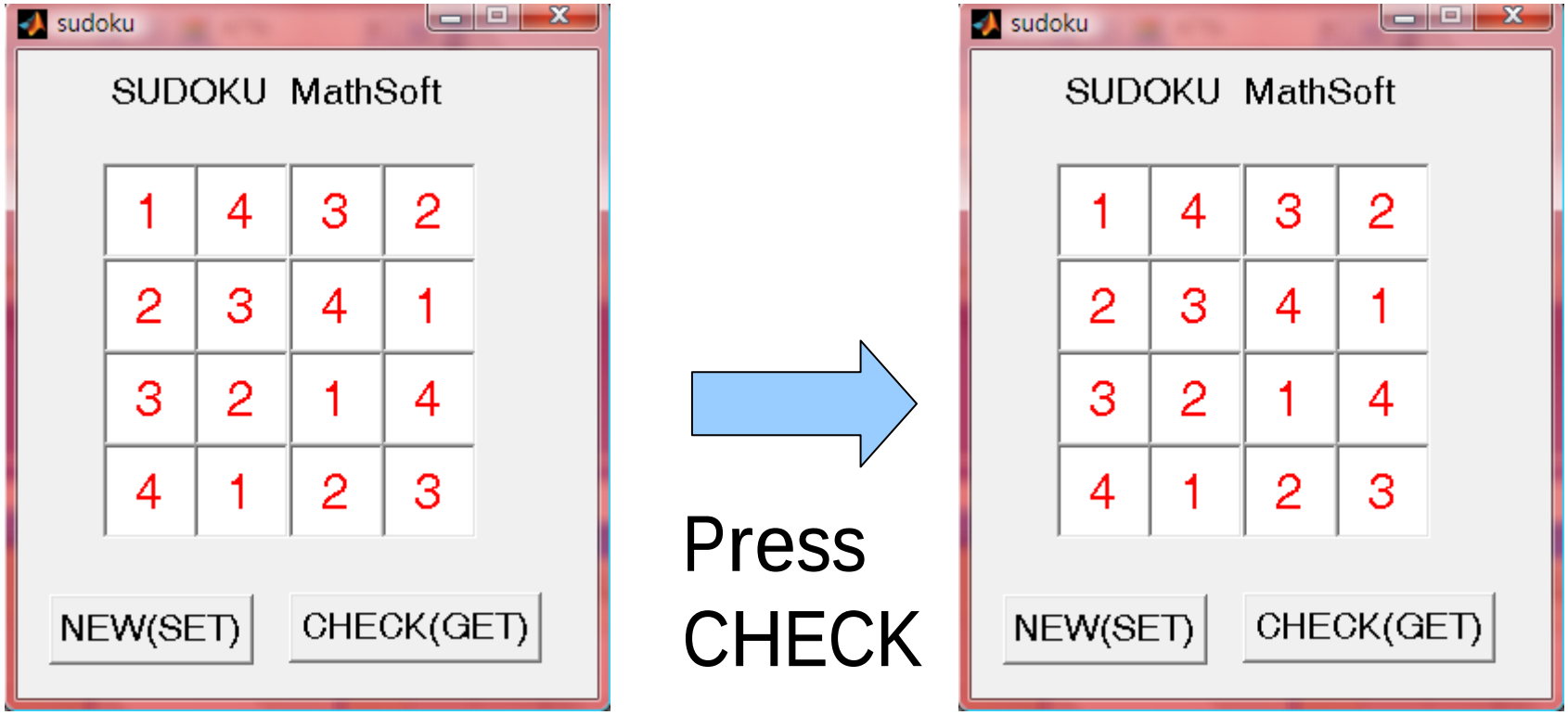
New

- Randomly assign four different digits to four cells in the Puzzle such that one and only one cell in each row and column is assigned
- p= randperm(4);q=randperm(4);
- for each i
  - Assign q(i) to the joint cell of column i and row p(i)
- Result
  - One number in each row
  - One number in each column
  - All four numbers assigned are different


sudoku
SUDOKU
NEW(SET)
CHECK(GET)

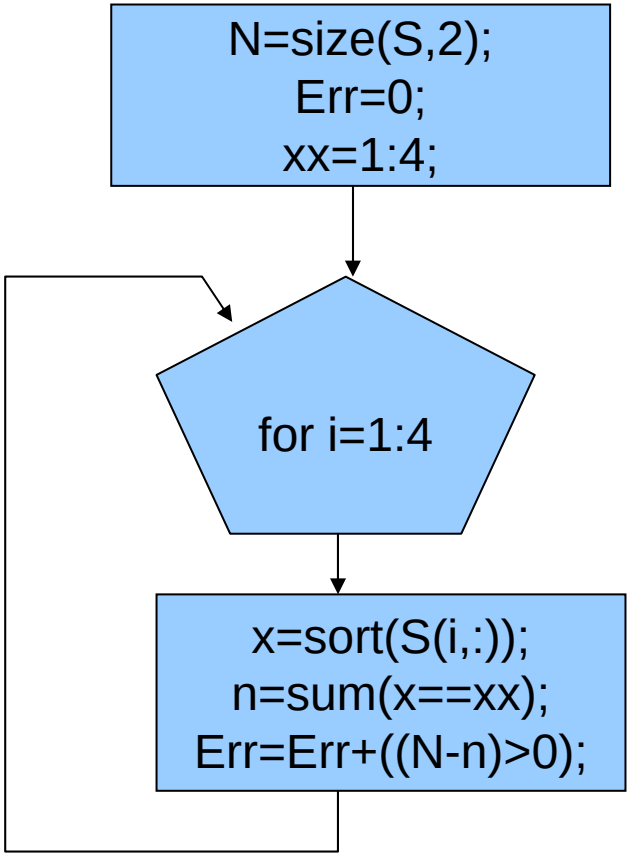
Press NEW


sudoku
SUDOKU MathSoft
1
4
2
3
NEW(SET)
CHECK(GET)


sudoku
SUDOKU MathSoft
1 4 3 2
2 3 4 1
3 2 1 4
4 1 2 3
NEW(SET)
CHECK(GET)
Press
CHECK
sudoku
SUDOKU MathSoft
1 4 3 2
2 3 4 1
3 2 1 4
4 1 2 3
NEW(SET)
CHECK(GET)

PLAY

Flow Chart : Row Checking

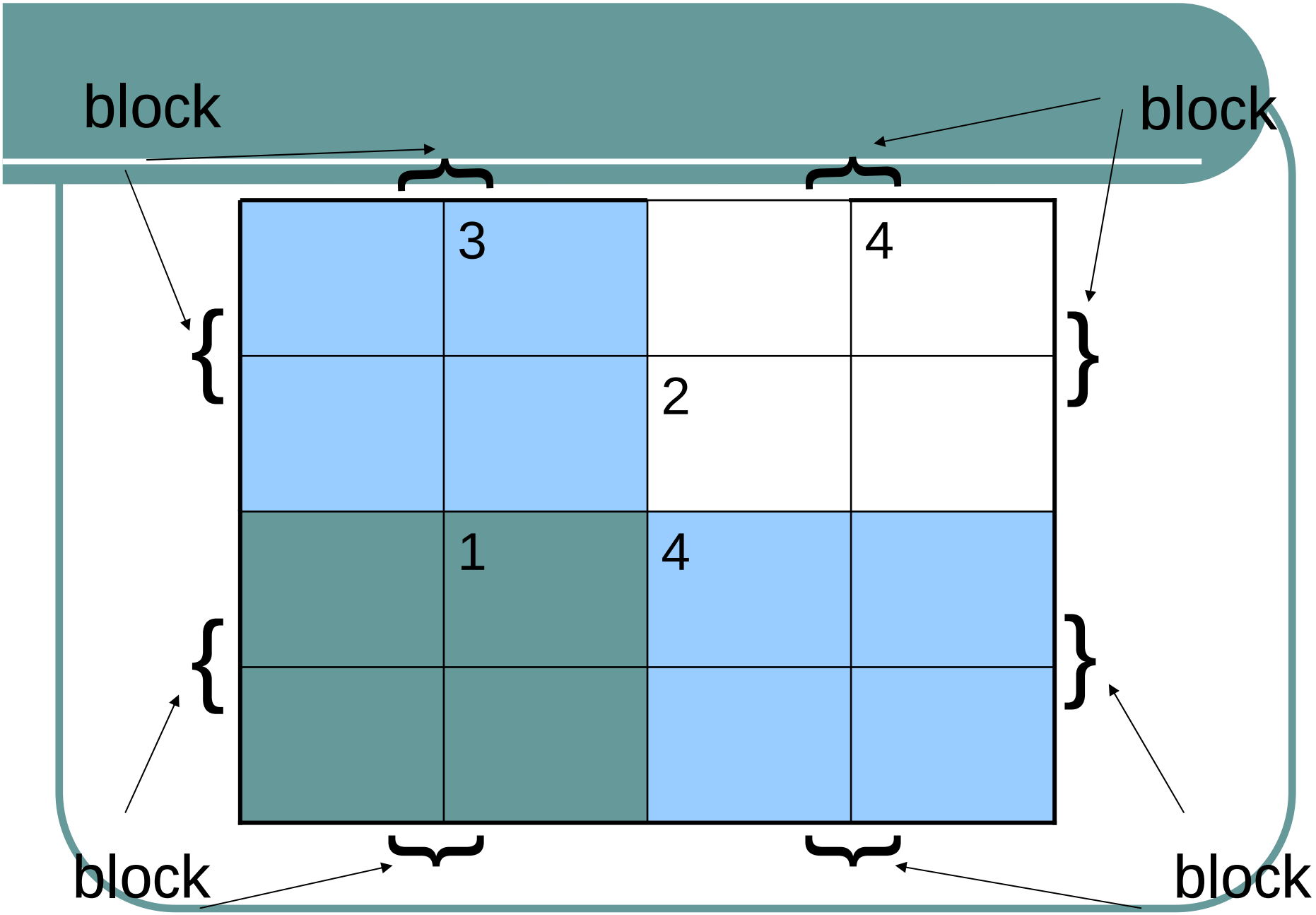


Procedure : Row Checking

- Input a 4-by-4 matrix S; N=4;
- Set err to zero
- for i=1:4
  A. Set n to the number of different digits in S(i,:)
  B. err =err +(N>n);
- Return err

Procedure : Column Checking

- Let S be a 4-by-4 matrix S; N=4
- Set err to zero
- for i=1:4
  A. Set n to the number of different digits in S(:,i)
  B. err =err +(N>n);
- Return err


block
block
3
4
2
1
4
block
block

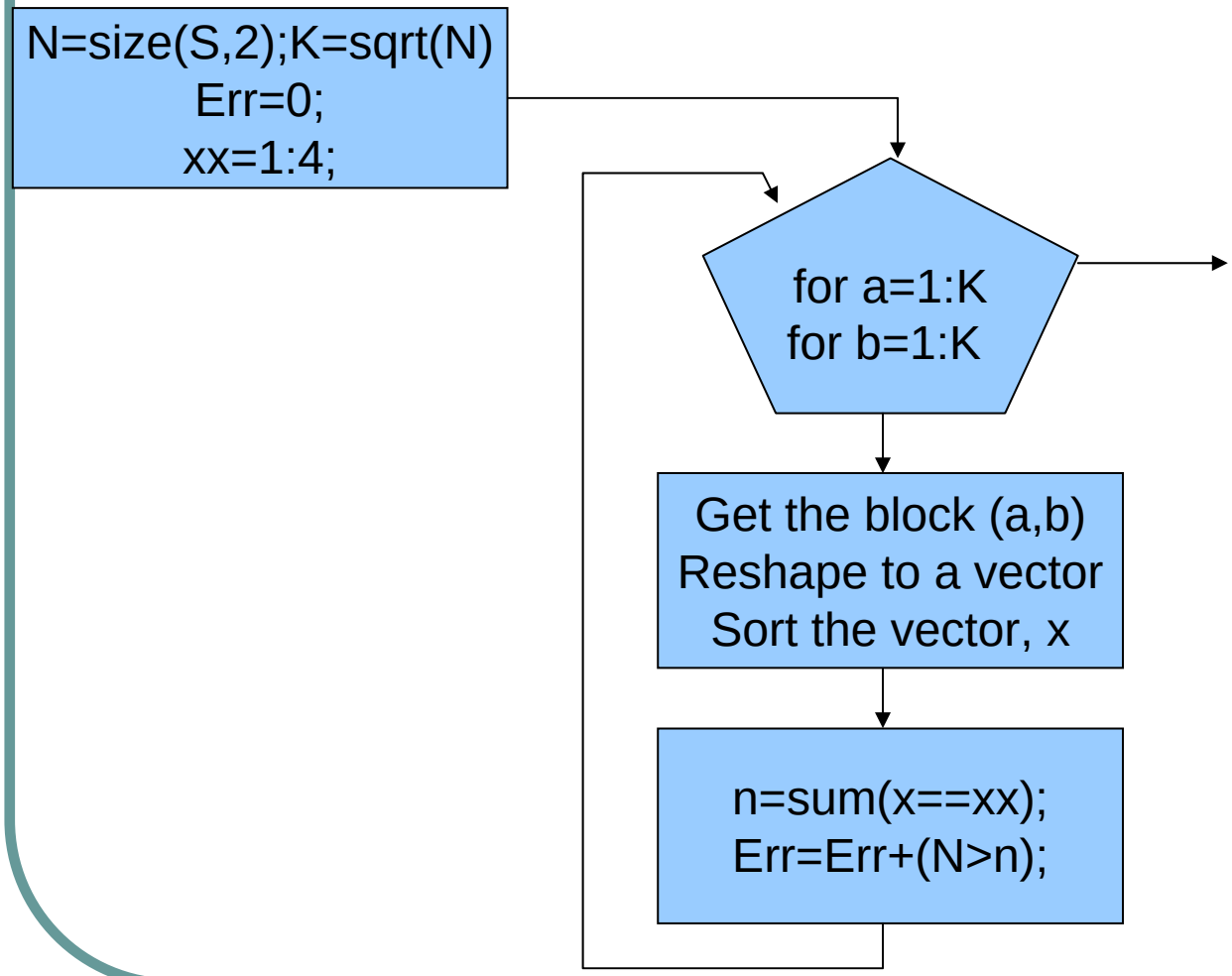
Indexing block

- Block 1: S(1:2,1:2)
- Block 2: S(1:2,3:4)
- Block 3: S(3:4,1:2)
- Block 4: S(3:4,3:4)

Indexing

- $K=2$; $N=4=K^2$
- Block 1 is denoted by (a,b)=(1,1)
  - S(1:2,1:2)
- Block 2 is denoted by (a,b)=(1,2)
  - S(1:2,3:4)
- Block 3 is denoted by (a,b)=(2,1)
  - S(3:4,1:2)
- Block 4 is denoted by (a,b)=(2,2)
  - S(3:4,3:4)
- Block (a,b)
  - S((a-1)*K+1:a*K, (b-1)*K+1:b*K)

Flow Chart: Block checking



Procedure: Block Checking

- Input a 4x4 matrix S; N = 4; K=2;
- Set Err to zero
- for a=1:K

  for b=1:K

  B=S((a-1)*K+1:a*K, (b-1)*K+1:b*K);

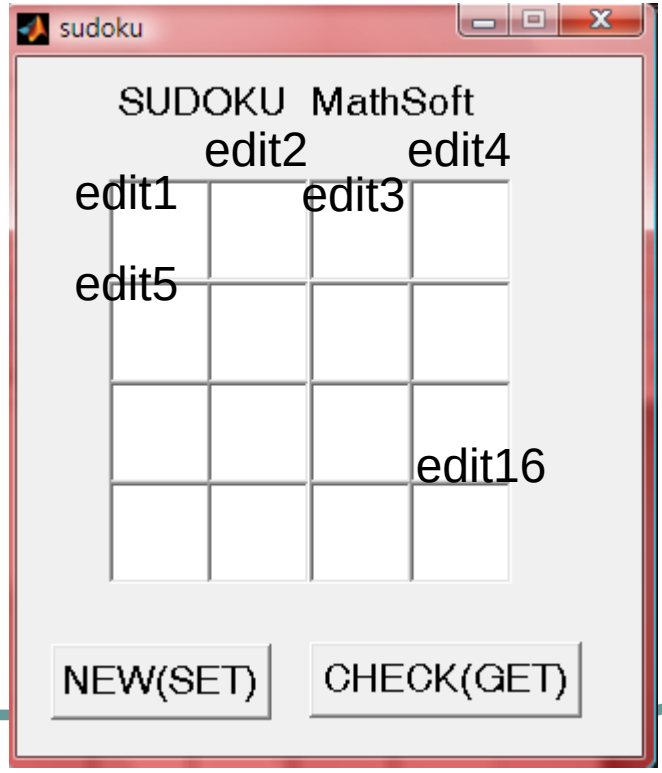
  x=sort(reshape(B,1,N));

  Set n to the number of different digits in x

  Err=Err+(N-n)>0
- Return err

Get puzzle

- Get N^2=16 digits in edit-cells of a puzzle
- Store 16 digits in a vector



Get Puzzle

```
function C=Get_Puzzle(hObject, eventdata, handles)
    C=zeros(1,16);
    for i=1:16
        str=['get(handles.edit' int2str(i) ',''String'')'];
        z=eval(str);
        if length(z) > 0
            C(i)=str2double(z);
        else
            C(i)=0;
        end
    end
```

Store a command to a string

```
>> str=['get(handles.edit' int2str(i) ','"String")']

str =

get(handles.edit0,'String')
```

eval

- Two equivalent codes

```
z=get(handles.edit0,'String')
```

```
z=eval(str);
```

```
str=['get(handles.edit' int2str(i) ','"String")'];
z=eval(str);
```

Get the content of the ith cell.
Cell names:
edit1, edit2, ..., edit16

Set Puzzle

```
function Set_Puzzle(hObject, eventdata, handles,C)
for i=1:16
    str=['set(handles.edit' int2str(i) ','''FontSize''',' '20)'];
    eval(str);
    if C(i) > 0
        str=['set(handles.edit' int2str(i) ','''ForegroundColor''',' '''Red''' ')'];
        eval(str);
    z=int2str(C(i));
    str=['set(handles.edit' int2str(i) ','''String''',' z ')'];
    eval(str);
    end
end
```

```
z=int2str(C(i));
str=['set(handles.edit' int2str(i) ',''String'',' z ')'];
eval(str);
```

Set the content of the ith cell to the number stored in C(i)

Exercise

- Download sudoku.m and sudoku.fig
- Implement the procedure for SUDOKU_New
- Implement procedures for row, column and block checking
- Revise sudoku.m
  - New a sudoku game if the button ‘NEW’ is pressed
  - Calculate errors of row, column and block checking

Function calling

- Create a SUDOKU game
  - Implement procedure SUDOKU_New

```
function S=New_Sudoku()
```

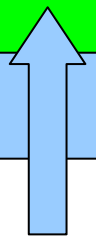
  - Call New_Sudoku at the beginning of pushbutton2_Callback

function C=New_Sudoku()

- Function name

  New_Sudoku
- Input
  - None
- Output
  - C: a 4x4 matrix

```
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
C=zeros(1,16);
C(1)=1;C(2)=2;C(3)=3;C(4)=4;
Set_Puzzle(hObject, eventdata, handles,C)
```



1. Call function New_Sudoku
2. The function output is a 4x4 matrix.
3. Reshape the output to a row vector before calling Set_Puzzle

Column-major reshape

```
S=zeros(4,4);
S(1,:)=[1 2 3 4];
C=reshape(S',1,16)
```

Function call

S=New_Puzzle();

function Call to New_Puzzle()

Store the return to S