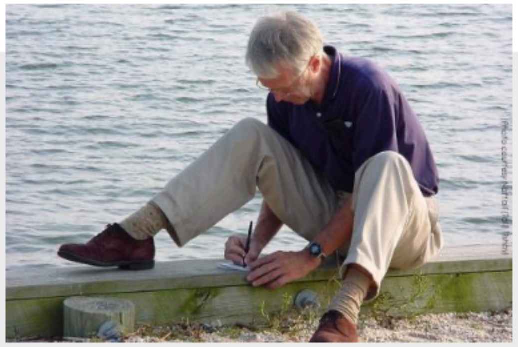
Integer programming and Sudoku Solving

Matlab intlinprog

MATLAB CVX

Python cvx



John J. Hopfield

Howard A. Prior Professor in the Life Sciences, Professor of Molecular Biology, Associated Faculty in PNI, Emeritus

Email
hopfield@princeton.edu

Office
150 Princeton Neuroscience Institute

Related Links:

- Now What?
- Videos
- Curriculum vitae
- Publications

Related News

PNI's John Hopfield receives Nobel Prize in physics

Searching for Memories, Sudoku, Implicit Check Bits, and the Iterative Use of Not-Always-Correct Rapid Neural Computation

Publisher: MIT Press

J. J. Hopfield **All Authors**

Abstract:
The algorithms that simple feedback neural circuits representing a brain area can rapidly carry out are often adequate to solve easy problems but for more difficult problems can return incorrect answers. A new excitatory-inhibitory circuit model of associative memory displays the common human problem of failing to rapidly find a memory when only a small clue is present. The memory model and a related computational network for solving Sudoku puzzles produce answers that contain implicit check bits in the representation of information across neurons, allowing a rapid evaluation of whether the putative answer is correct or incorrect through a computation related to visual pop-out. This fact may account for our strong psychological feeling of right or wrong when we retrieve a nominal memory from a minimal clue. This information allows more difficult computations or memory retrievals to be done in a serial fashion by using the fast but limited capabilities of a computational module multiple times. The

Bio-Inspired Hashing for Unsupervised Similarity Search

Chaitanya K. Ryali [1 2] **John J. Hopfield** [3] **Leopold Grinberg** [4] **Dmitry Krotov** [2 4]

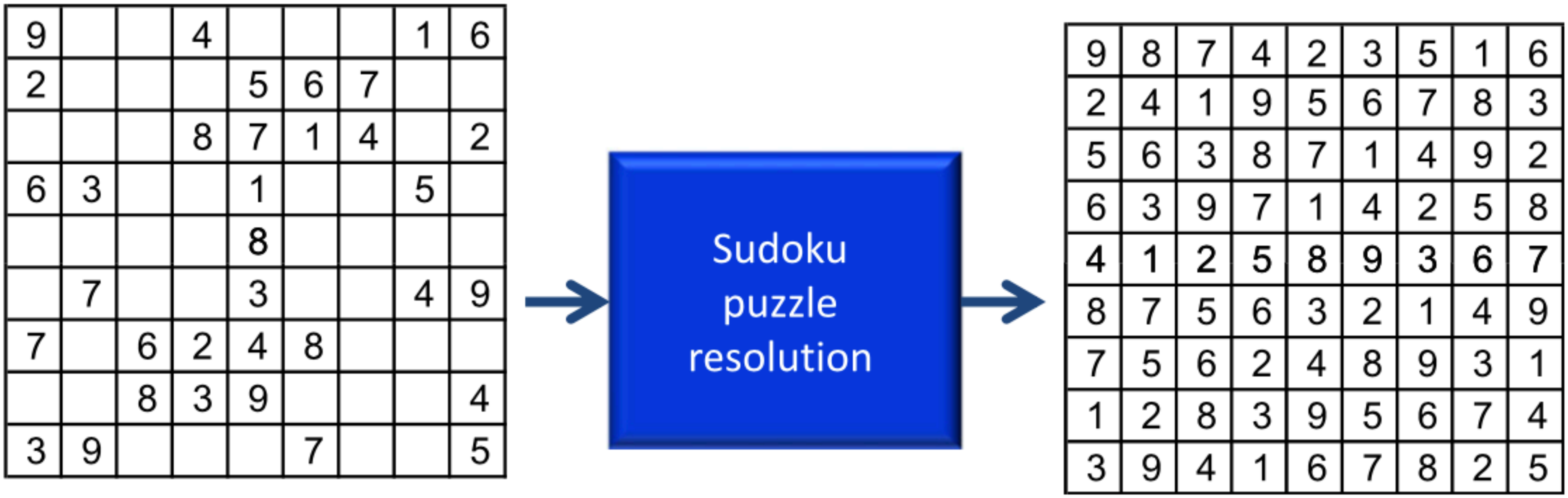
Abstract

The fruit fly Drosophila's olfactory circuit has inspired a new locality sensitive hashing (LSH) algorithm, `FlyHash`. In contrast with classical LSH algorithms that produce low dimensional hash codes, `FlyHash` produces sparse high-dimensional hash codes and has also been shown to have superior empirical performance compared to classical LSH algorithms in similarity search. However, `FlyHash` uses random projections and cannot *learn* from data. Building on inspiration from `FlyHash` and the ubiquity of sparse expansive representations in neurobiology, our work proposes a novel hashing algorithm `BioHash` that produces sparse high dimensional hash codes in a *data-driven* manner. We show that `BioHash` outperforms previously published benchmarks for various hashing methods. Since our learn-

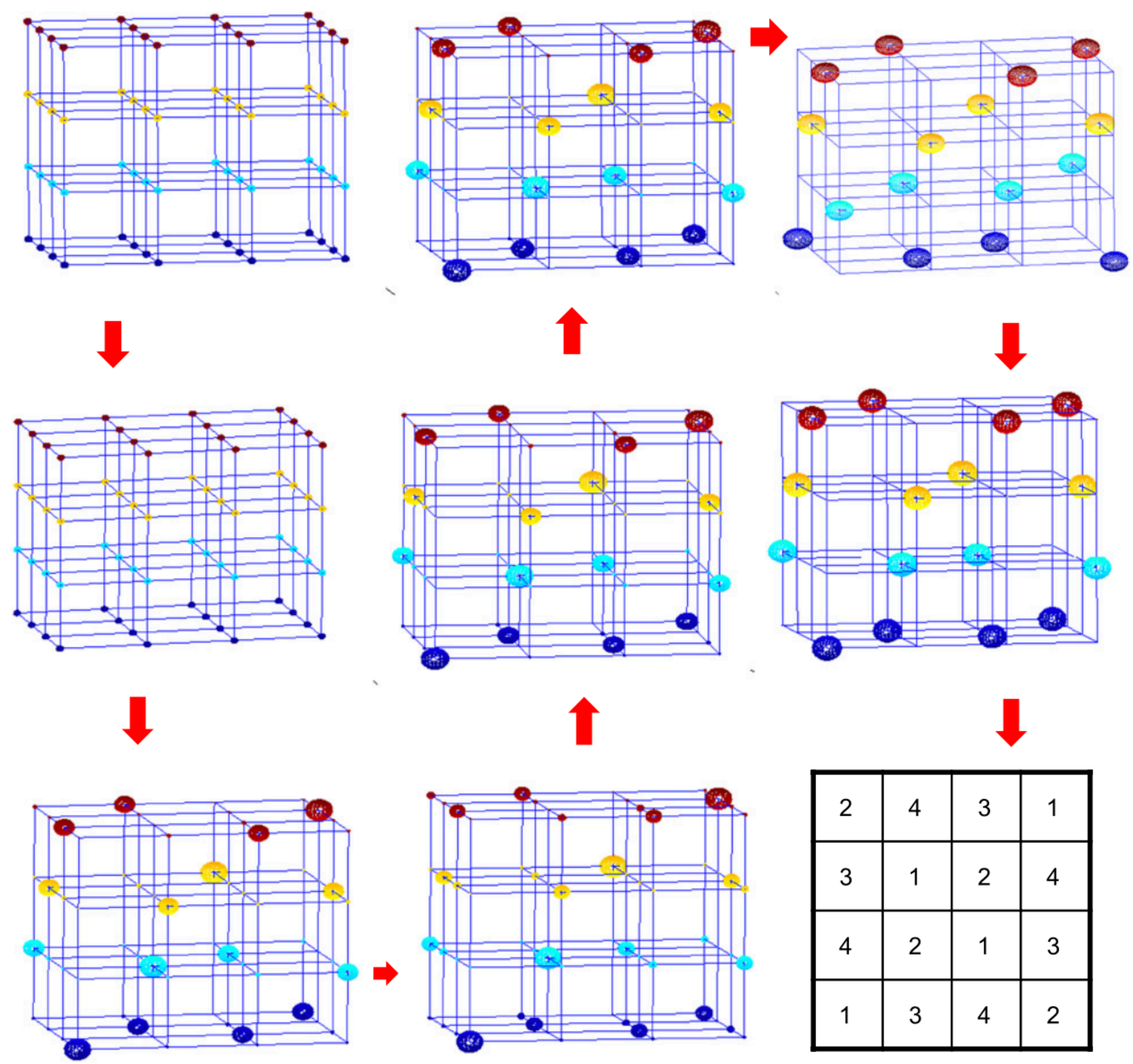
into a sparse code, where only a small number of secondary neurons respond to a given stimulus.

A classical example of the sparse expansive motif is the Drosophila fruit fly olfactory system. In this case, approximately 50 projection neurons send their activities to about 2500 Kenyon cells (Turner et al., 2008), thus accomplishing an approximately 50x expansion. An input stimulus typically activates approximately 50% of projection neurons, and less than 10% Kenyon cells (Turner et al., 2008), providing an example of significant sparsification of the expanded codes. Another example is the rodent olfactory circuit. In this system, dense input from the olfactory bulb is projected into piriform cortex, which has 1000x more neurons than the number of glomeruli in the olfactory bulb. Only about 10% of those neurons respond to a given stimulus (Mombaerts et al., 1996). A similar motif is found in rat's cerebellum and hippocampus (Dasgupta et al., 2017).

From the computational perspective, expansion is helpful

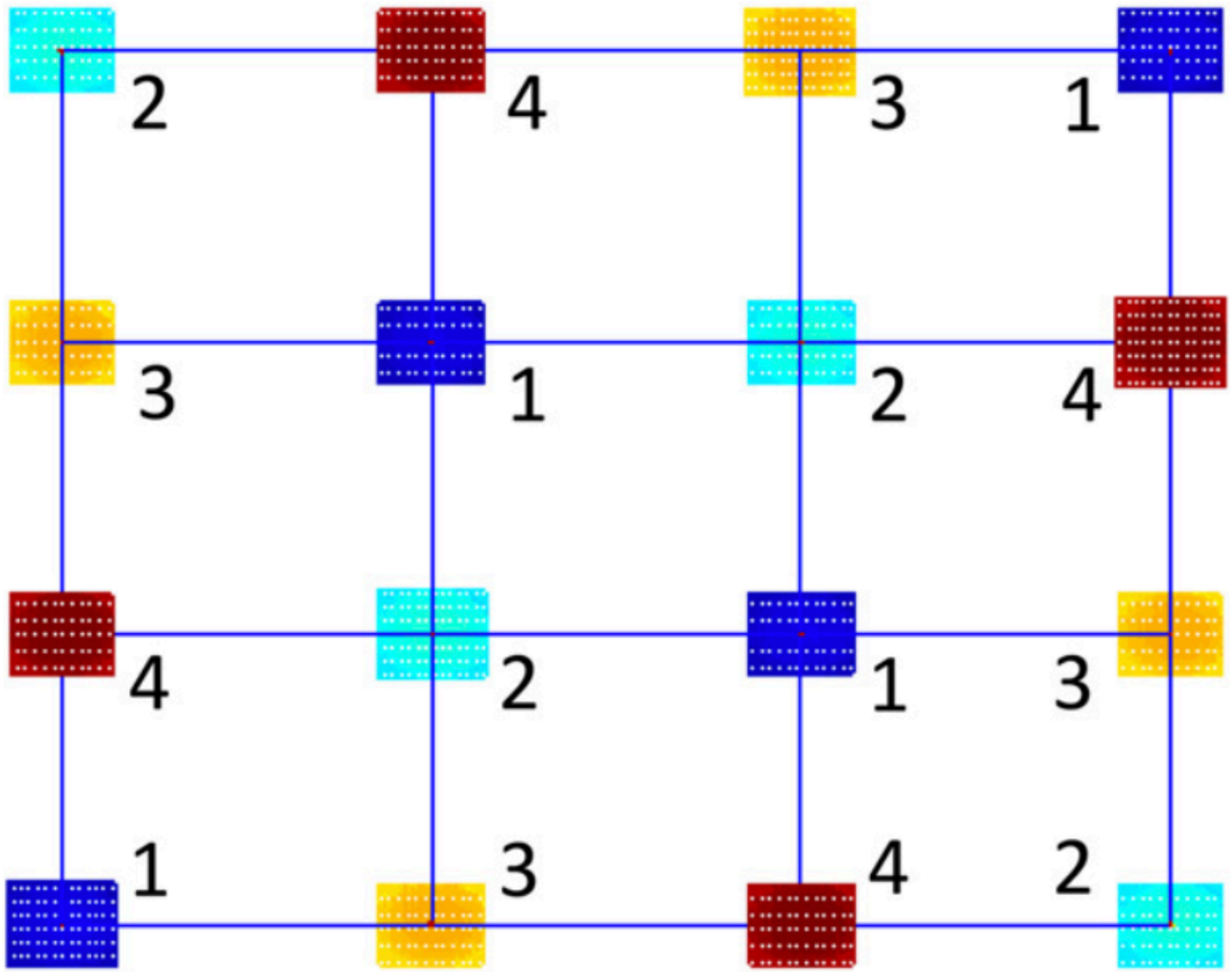

9 4 1 6
2 5 6 7
8 7 1 4 2
6 3 1 5
8
7 3 4 9
7 6 2 4 8
8 3 9 4
3 9 7 5
Sudoku puzzle resolution
9 8 7 4 2 3 5 1 6
2 4 1 9 5 6 7 8 3
5 6 3 8 7 1 4 9 2
6 3 9 7 1 4 2 5 8
4 1 2 5 8 9 3 6 7
8 7 5 6 3 2 1 4 9
7 5 6 2 4 8 9 3 1
1 2 8 3 9 5 6 7 4
3 9 4 1 6 7 8 2 5

Relaxation of Neural Dynamics

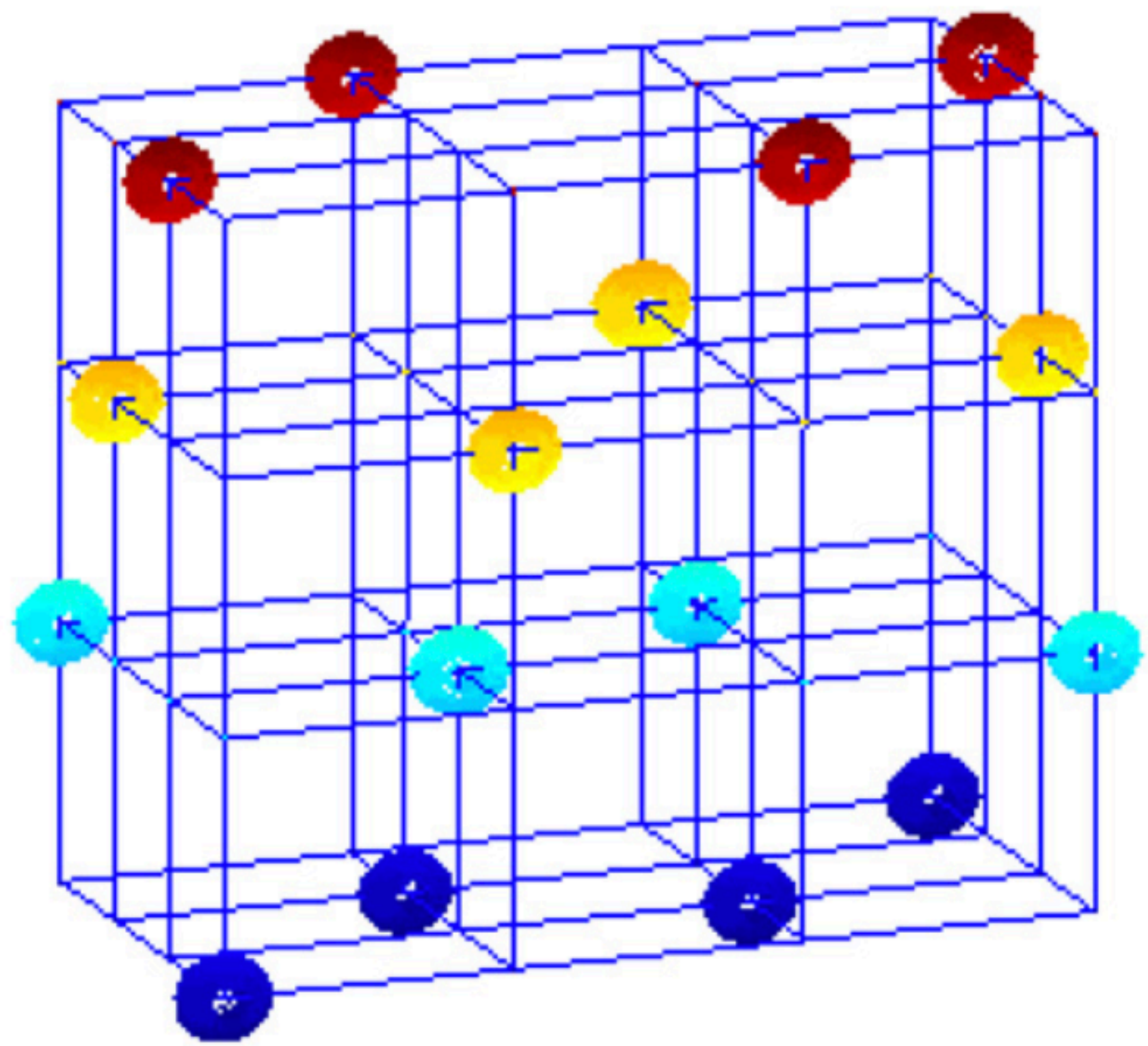


2	4	3	1
3	1	2	4
4	2	1	3
1	3	4	2

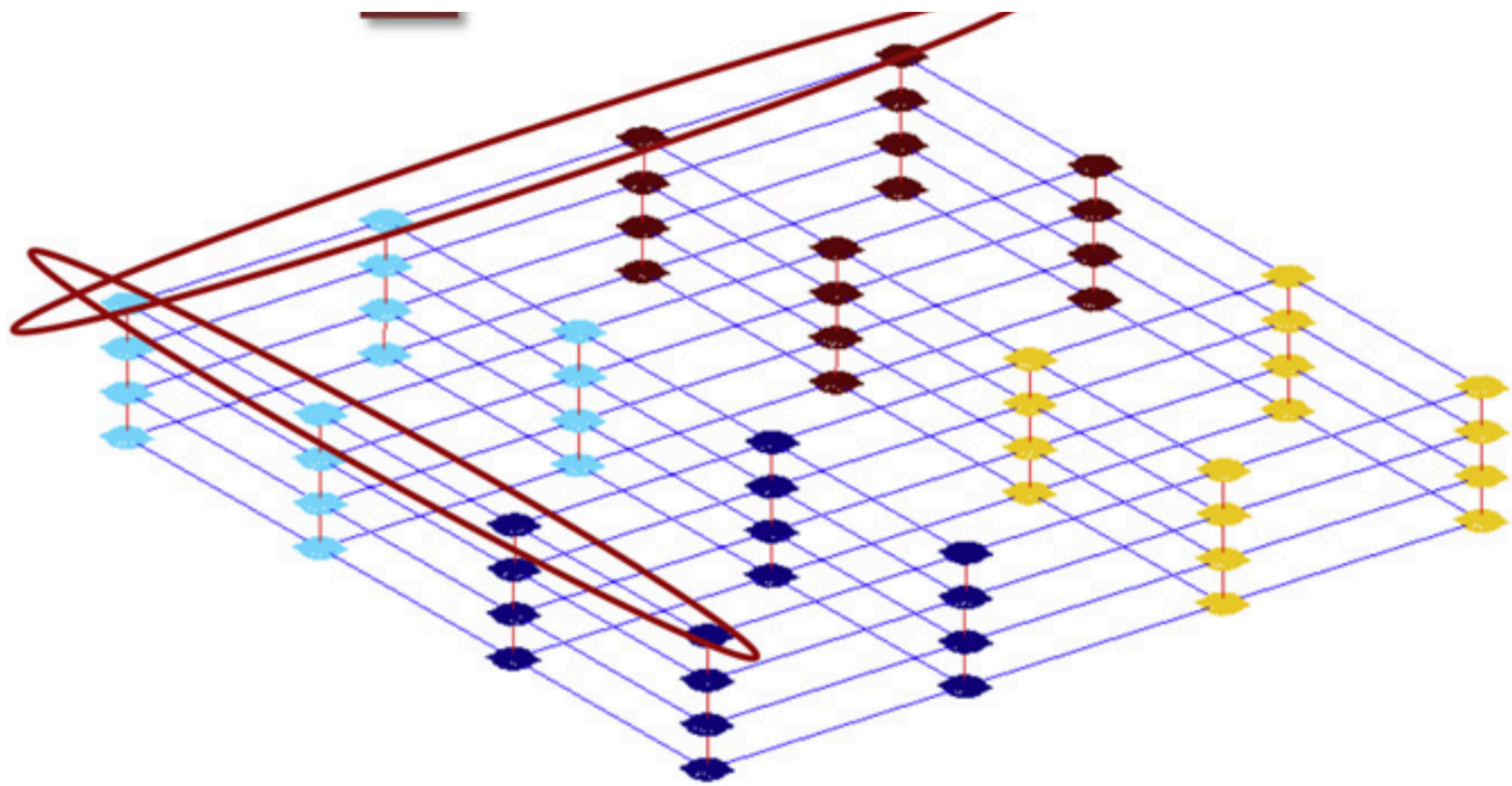
Convergence to solve 4-by-4 Sudoku

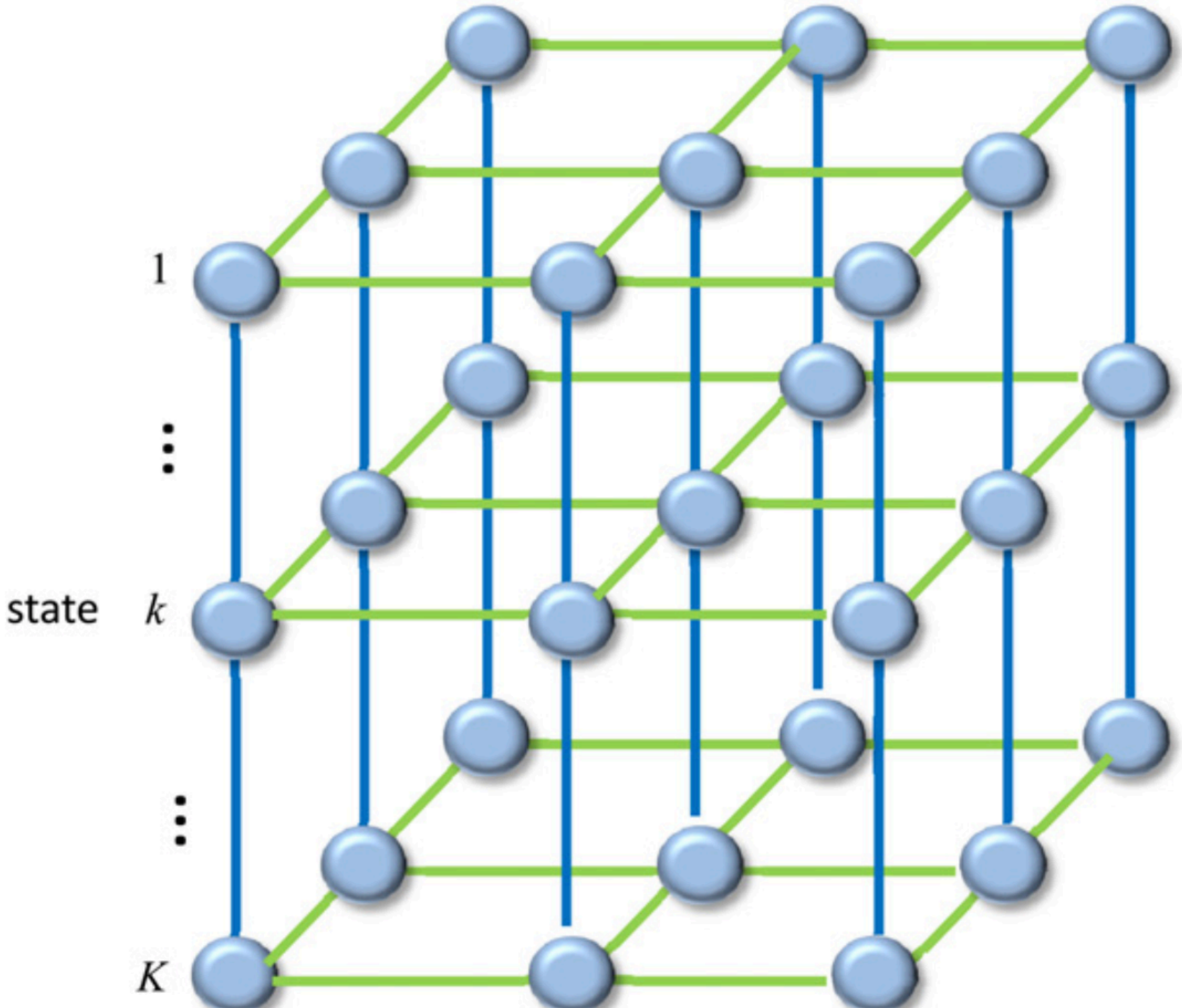


Three Dimensional Neural Circuits



Inhibition constraints





729 unkowns

$x(i, j, k) \in \{0,1\}$

$i \in \{1...9\}$

$j \in \{1...9\}$

$k \in \{1...9\}$

Three constraints of binary variables for solving 9×9 Sudoku

```
B = [1,2,2;
    1,5,3;
    1,8,4;
    2,1,6;
    2,9,3;
    3,3,4;
    3,7,5;
    4,4,8;
    4,6,6;
    5,1,8;
    5,5,1;
    5,9,6;
    6,4,7;
    6,6,5;
    7,3,7;
    7,7,6;
    8,1,4;
    8,9,8;
    9,2,3;
    9,5,4;
    9,8,2];
drawSudoku(B)
```

Constraint I: Clues

$j = 4$

$i = 6$

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

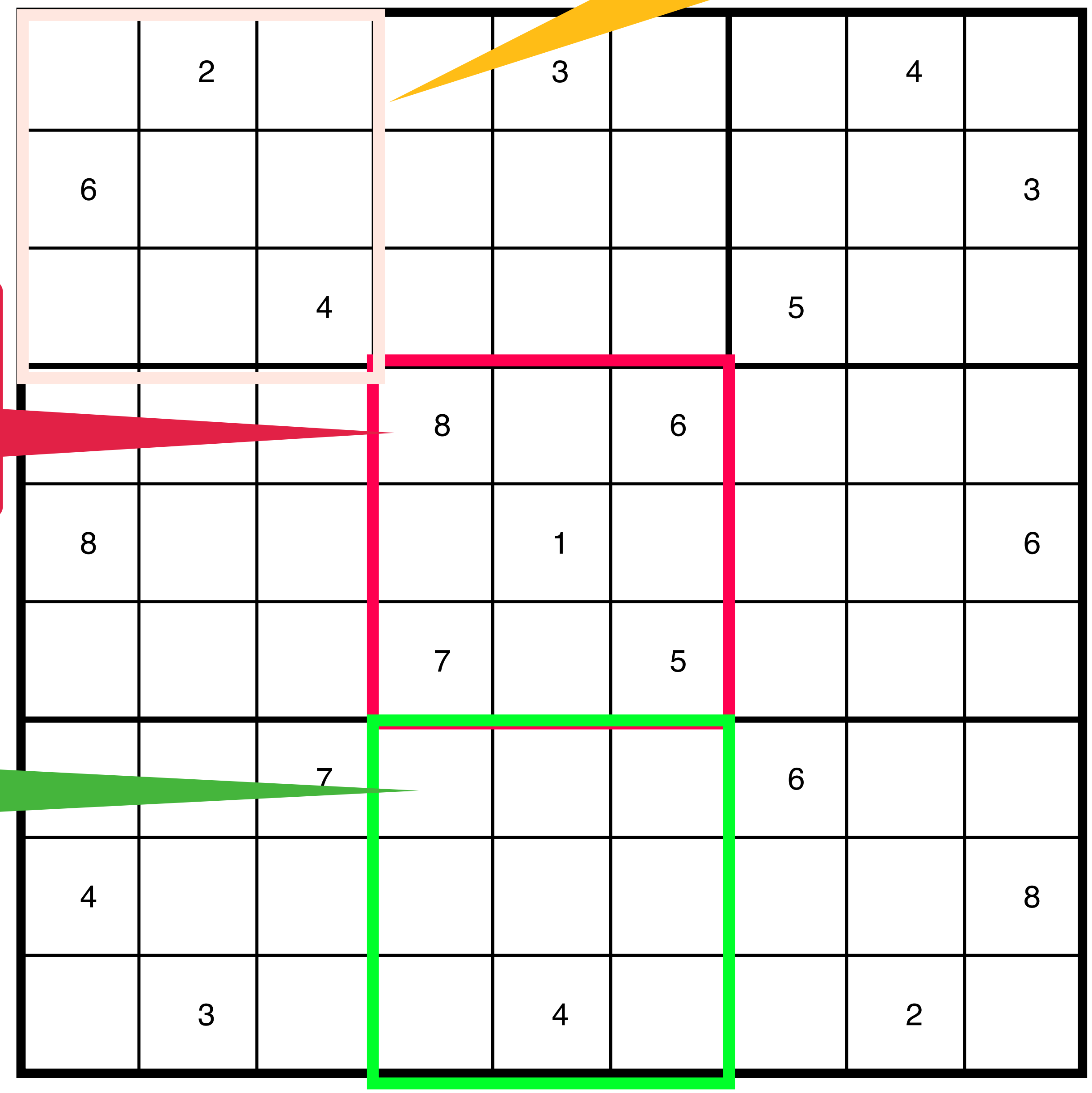
$$x(6,4,7) = 1$$

Binary variables

$$x(i,j,k) \in \{0,1\}$$

81 Constraint II: nine blocks

$$U = o, V = o, \sum_{i=1}^{3} \sum_{j=1}^{3} x(i+0, j+0, k) = 1$$



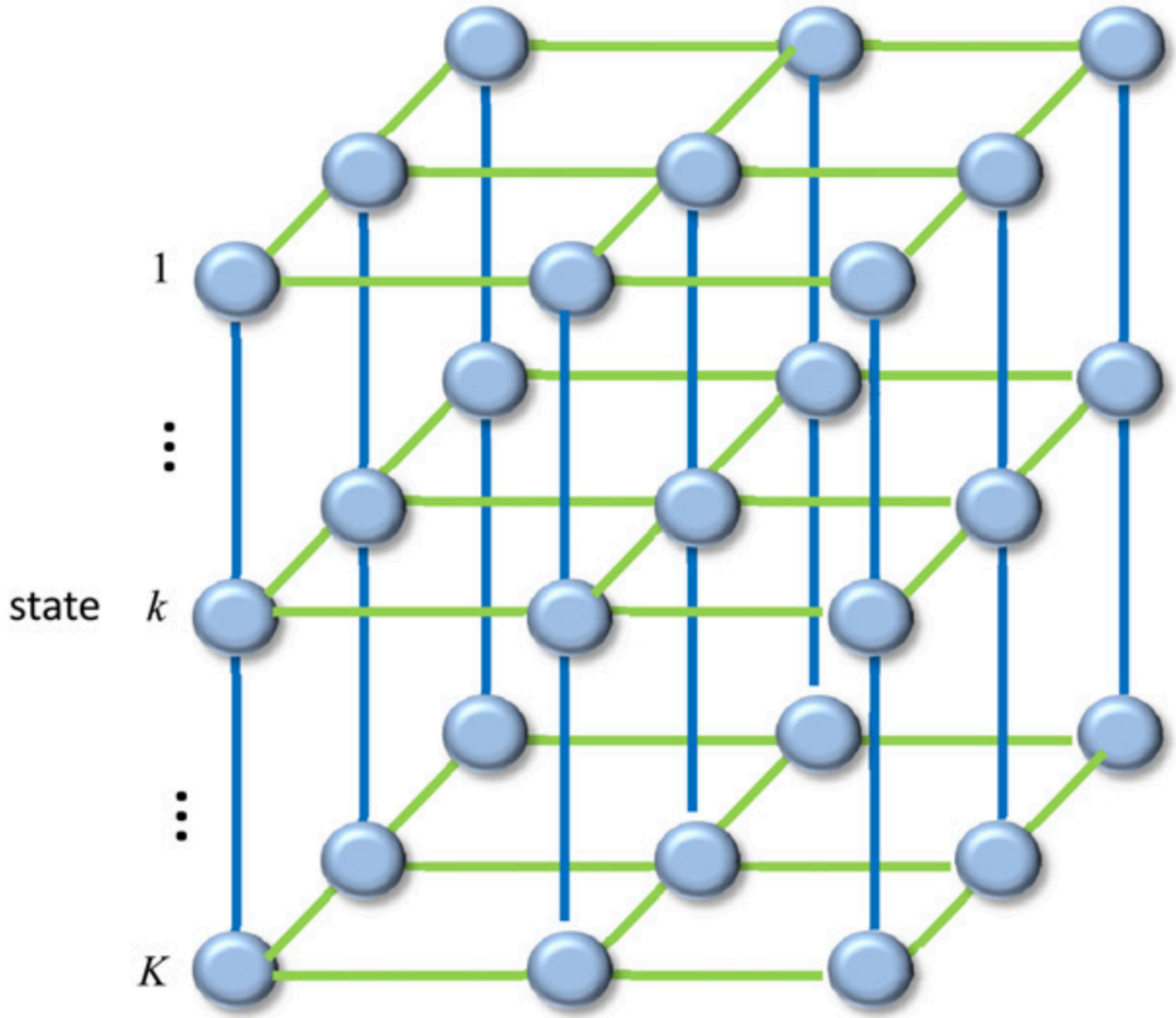
$$x(3+3, 1+3, 7) = 1$$

$$\sum_{i=1}^{3} \sum_{j=1}^{3} x(i+3, j+3, 1) = 1$$

$$x(2+3, 2+3, 1) = 1$$

$$\sum_{i=1}^{3} \sum_{j=1}^{3} x(i+U, j+V, k) = 1, \text{ where } U, V \in \{0,3,6\}$$

$$k \in \{1 \ldots 9\}$$



$$\sum_{k=1}^{9} x(i,j,k) = 1$$

$$\sum_{j=1}^{9} x(i,j,k) = 1$$

$$\sum_{i=1}^{9} x(i,j,k) = 1$$

Constraints III: Row, column, unitary conditions

$j = 4$

$i = 6$

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

$$\sum_{k} x(i,j,k) = 1$$

81 constraints for different i and j

$$\sum_{j} x(i,j,k) = 1$$

$$\sum_{i} x(i,j,k) = 1$$

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

$$\sum_{k} x(1,8,k) = 1$$

$$\sum_{k} x(1,9,k) = 1$$

$$\sum_k x(i,j,k) = 1$$

$$\boxed{\sum_j x(i,j,k) = 1}$$

81 constraints for different i and k

$$\sum_i x(i,j,k) = 1$$

$$\sum_j x(1,j,4) = 1$$

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

Nine-alphabet Latin Square

$$\sum_{k} x(i,j,k) = 1$$

$$\sum_{j} x(i,j,k) = 1$$

$$\sum_{i} x(i,j,k) = 1$$

$$\sum_{i} x(i,8,4) = 1$$

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

Matlab programming

$$Ax = b$$
$$x : 729 \times 1$$
$$A : 324 \times 729$$
$$b : 324 \times 1$$

```
%% The rules of Sudoku:
N = 9^3; % number of independent variables in ... 9 array
M = 4*9^2; % number of constraints, see the c... eq
Aeq = zeros(M,N); % allocate equality constraint matrix Aeq*x = beq
beq = ones(M,1); % allocate constant vector beq
f = (1:N)'; % the objective can be anything, but having nonconstant f can speed the solver
lb = zeros(9,9,9); % an initial zero array
ub = lb+1; % upper bound array to give binary variables
```

Equality constraint

$$\text{Aeq} * x = \text{beq}$$

Aeq: 324×729

x(:) : 729×1

b: 324×1

```
B = [1,2,2;
    1,5,3;
    1,8,4;
    2,1,6;
    2,9,3;
    3,3,4;
    3,7,5;
    4,4,8;
    4,6,6;
    5,1,8;
    5,5,1;
    5,9,6;
    6,4,7;
    6,6,5;
    7,3,7;
    7,7,6;
    8,1,4;
    8,9,8;
    9,2,3;
    9,5,4;
    9,8,2];
drawSudoku(B)
```

```
lb = zeros(9,9,9); % an initial zero array
ub = lb+1; % upper bound array to give binary variables
```

Upper bound of of $x(6,4,7)$ is one

```
for i = 1:size(B,1)
    lb(B(i,1),B(i,2),B(i,3)) = 1;
end
```

Lower bound of $x(6,4,7)$ is set to one

It is implied that $x(6,4,7)$ is one

Equality constraint

$\text{Aeq} * x = \text{beq}$

$$\sum_k x(i,j,k) = 1$$

x is a $9 \times 9 \times 9$ matrix.
For some j and k, sum of 9 elements with different i is one

$$\sum_j x(i,j,k) = 1$$

$$\sum_i x(i,j,k) = 1$$

$$[a_{111} \quad a_{112} \ldots a_{119} \quad a_{121} \quad a_{122} \ldots a_{129} \ldots a_{191} \quad a_{192} \ldots a_{199} \quad a_{211} \ldots a_{299} \quad a_{311} \ldots a_{399} \ldots a_{911} \ldots a_{999}] x(:) = 1$$

Specify nine elements in a to one such that sum of 9 elements with different i is one for some j and k

$a_{184} = 1, a_{284} = 1, \ldots, a_{984} = 1$ And remaining 720 a_{i84} terms are zero

$$[a_{111} \quad a_{112} \ldots a_{119} \quad a_{121} \quad a_{122} \ldots a_{129} \ldots a_{191} \quad a_{192} \ldots a_{199} \quad a_{211} \ldots a_{299} \quad a_{311} \ldots a_{399} \ldots a_{911} \ldots a_{999}] x(:) = 1$$

729 elements

$$\sum_i x(i,8,4) = a_{184}x_{184} + a_{284}x_{284} + \ldots + a_{984}x_{984} = 1$$

$$\sum_i x(i,8,4) = 1$$

$$\sum_i x(i,j,k) = 1$$

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

$$\sum_k x(i,j,k) = 1$$

$$\sum_j x(i,j,k) = 1$$

$$\sum_i x(i,j,k) = 1$$

lb is a $9 \times 9 \times 9$ matrix.
For some j and k, sum of 9 elements with different i is one

```
counter = 1;
for j = 1:9 % one in each row
    for k = 1:9
        Astuff = lb; % clear Astuff
        Astuff(1:end,j,k) = 1; % one row in Aeq*x = beq
        Aeq(counter,:) = Astuff(:)'; % put Astuff in a row of Aeq
        counter = counter + 1;
    end
end
```

Specify nine elements to one such that sum of 9 elements with different i is one for some j and k

$$\sum_k x(i,j,k) = 1$$

lb is a $9 \times 9 \times 9$ matrix.
For some i and k, sum of 9 elements with different j is one

$$\sum_j x(i,j,k) = 1$$

$$\sum_i x(i,j,k) = 1$$

```
for i = 1:9 % one in each column
    for k = 1:9
        Astuff = lb;
        Astuff(i,1:end,k) = 1;
        Aeq(counter,:) = Astuff(:)';
        counter = counter + 1;
    end
end
```

Specify nine elements to one such that sum of 9 elements with different j is one for some i and k

lb is a $9 \times 9 \times 9$ matrix.
For some i and k, sum of 9 elements with different i is one

$$\sum_k x(i,j,k) = 1$$

$$\sum_j x(i,j,k) = 1$$

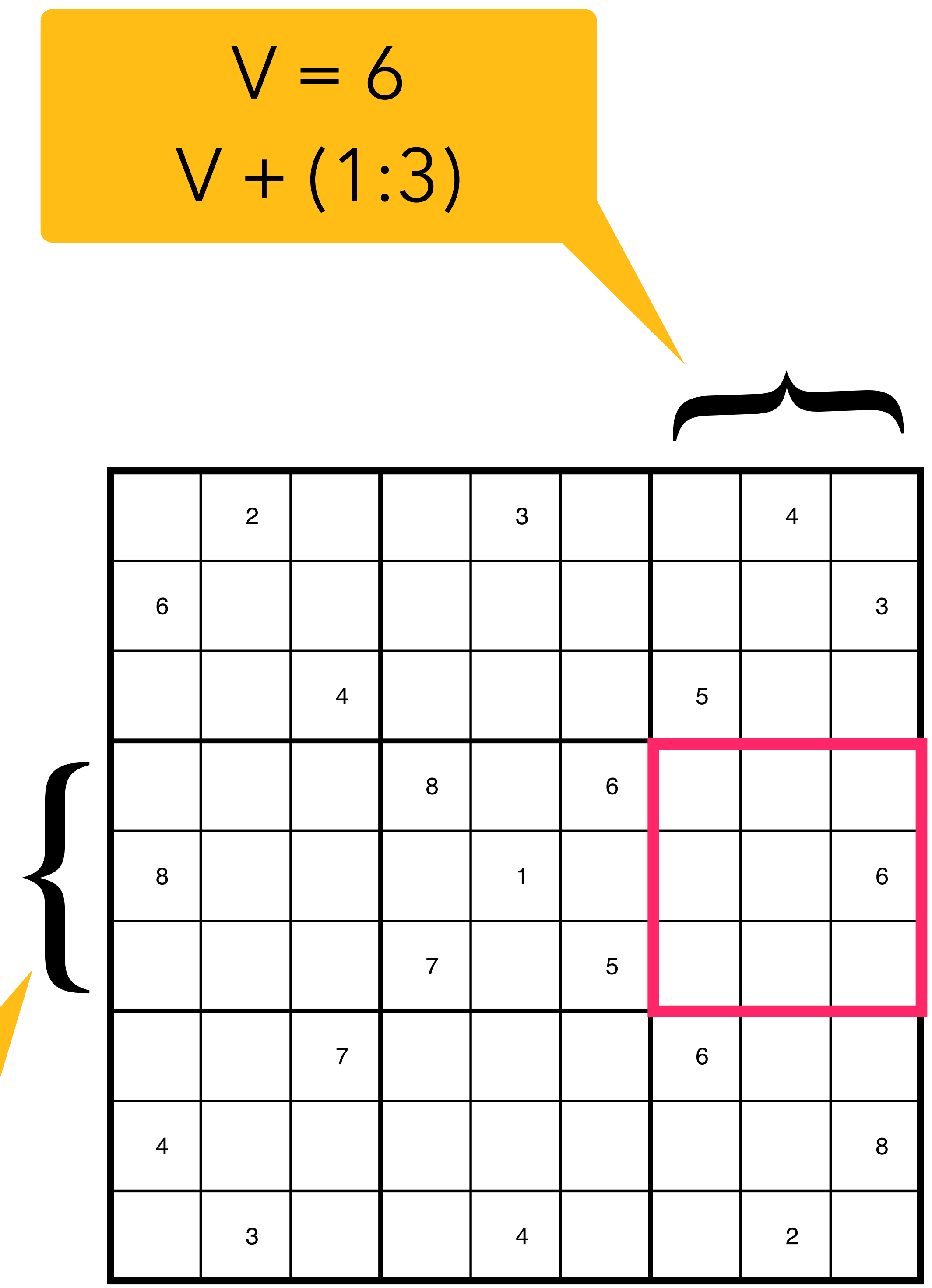
$$\sum_i x(i,j,k) = 1$$

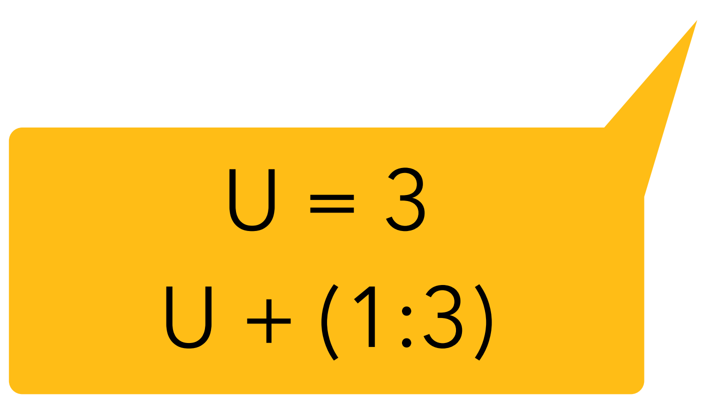
```
for i = 1:9 % one in each depth
    for j = 1:9
        Astuff = lb;
        Astuff(i,j,1:end) = 1;
        Aeq(counter,:) = Astuff(:)';
        counter = counter + 1;
    end
end
```

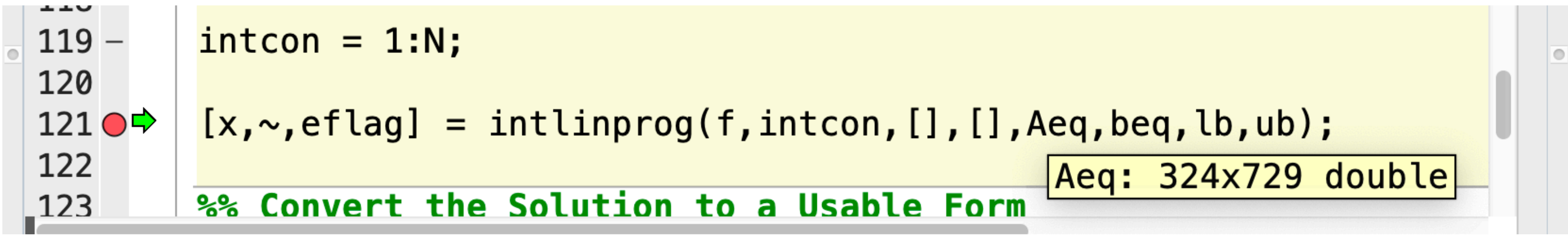
Specify nine elements to one such that sum of 9 elements with different k is one for some i and j

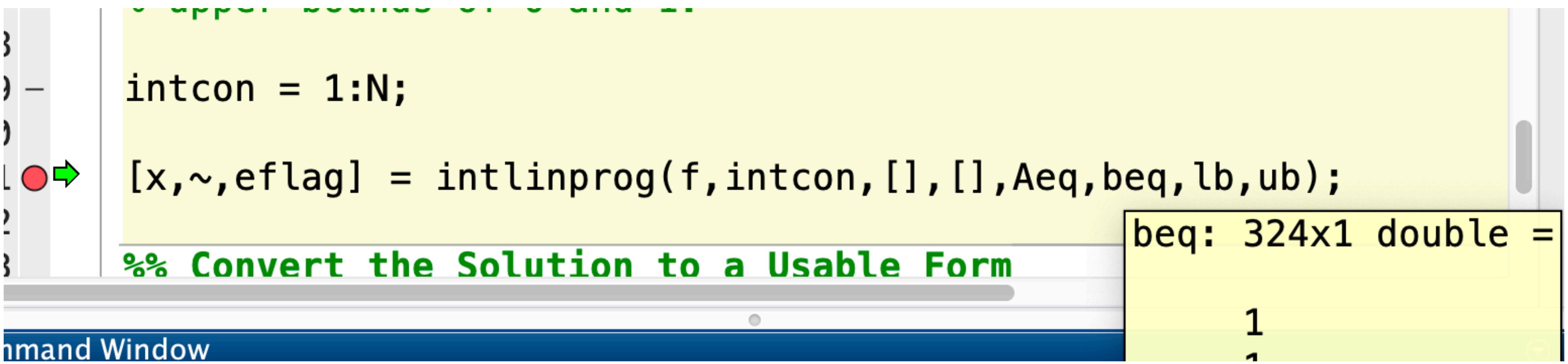
$$\sum_{i=1}^{3} \sum_{j=1}^{3} x(i+U, j+V, k) = 1, \text{ where } U, V \in \{0,3,6\}$$

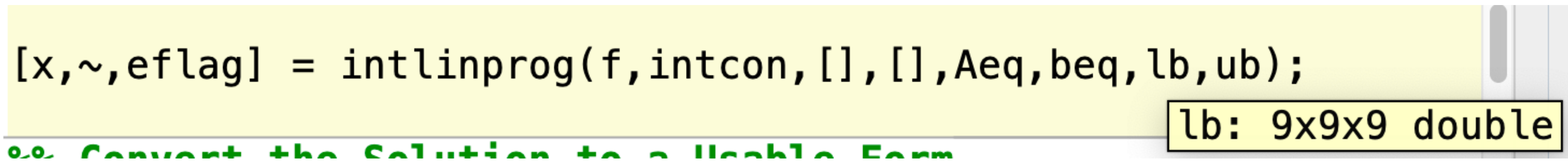
```
for U = 0:3:6 % one in each square
    for V = 0:3:6
        for k = 1:9
            Astuff = lb;
            Astuff(U+(1:3),V+(1:3),k) = 1;
            Aeq(counter,:) = Astuff(:)';
            counter = counter + 1;
        end
    end
end
```



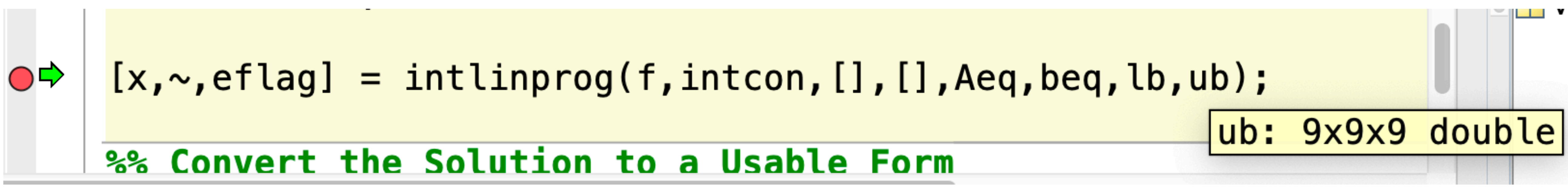



119 intcon = 1:N;
120
121 [x,~,eflag] = intlinprog(f,intcon,[],[],Aeq,beq,lb,ub);
122
123 %% Convert the Solution to a Usable Form
Aeq: 324x729 double


intcon = 1:N;
[x,~,eflag] = intlinprog(f,intcon,[],[],Aeq,beq,lb,ub);
%% Convert the Solution to a Usable Form
beq: 324x1 double =
1



[x,~,eflag] = intlinprog(f,intcon,[],[],Aeq,beq,lb,ub);
lb: 9x9x9 double



[x,~,eflag] = intlinprog(f,intcon,[],[],Aeq,beq,lb,ub);
ub: 9x9x9 double
%% Convert the Solution to a Usable Form

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

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

```
B = [1, 7, 8;
    1, 9,7;
    2, 1, 3;
    2, 2, 2;
    2, 4, 7;
    2, 7, 5;
    3, 3, 7;
    3, 4, 8;
    3, 5, 1;
    3, 7, 6;
    3, 8, 2;
    4, 5, 4;
    5, 1, 1;
    5, 3, 9;
    5, 7, 7;
    5, 9, 4;
    6, 5, 8;
    7, 2, 9;
    7, 3, 4; 7, 5, 7; 7, 6, 5; 7, 7, 2;
    8, 3, 1; 8, 6, 4; 8, 8, 7; 8, 9, 3;
    9, 1, 7; 9, 3, 2]
```

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

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

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

Solve Sudoku Puzzles Via Integer Programming: Problem-Based

Open Live Script

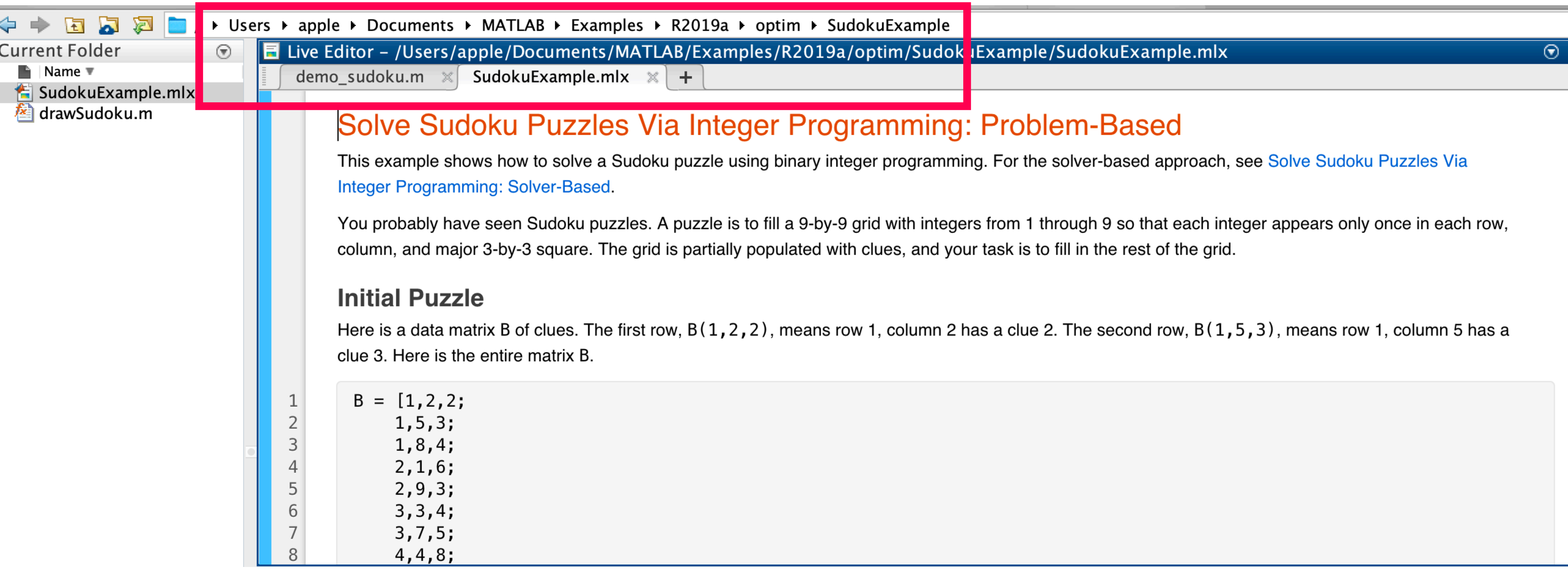
This example shows how to solve a Sudoku puzzle using binary integer programming. For the solver-based approach, see Solve Sudoku Puzzles Via Integer Programming: Solver-Based.

You probably have seen Sudoku puzzles. A puzzle is to fill a 9-by-9 grid with integers from 1 through 9 so that each integer appears only once in each row, column, and major 3-by-3 square. The grid is partially populated with clues, and your task is to fill in the rest of the grid.

Initial Puzzle

Here is a data matrix B of clues. The first row, `B(1,2,2)`, means row 1, column 2 has a clue 2. The second row, `B(1,5,3)`, means row 1, column 5 has a clue 3. Here is the entire matrix B.

```
B = [1,2,2;
    1,5,3;
    1,8,4;
    2,1,6;
    2,9,3;
    3,3,4;
    3,7,5;
    4,4,8;
    4,6,6;
    5,1,8;
    5,5,1;
    5,9,6;
    6,4,7;
    6,6,5;
    7,3,7;
    7,7,6;
    8,1,4;
    8,9,8;
    9,2,3;
    9,5,4;
    9,8,2];

drawSudoku(B) % For the listing of this program, see the end of this example.
```


Users ▸ apple ▸ Documents ▸ MATLAB ▸ Examples ▸ R2019a ▸ optim ▸ SudokuExample
Current Folder
Name
SudokuExample.mlx
drawSudoku.m
Live Editor – /Users/apple/Documents/MATLAB/Examples/R2019a/optim/SudokuExample/SudokuExample.mlx
demo_sudoku.m
SudokuExample.mlx
Solve Sudoku Puzzles Via Integer Programming: Problem-Based
This example shows how to solve a Sudoku puzzle using binary integer programming. For the solver-based approach, see Solve Sudoku Puzzles Via Integer Programming: Solver-Based.
You probably have seen Sudoku puzzles. A puzzle is to fill a 9-by-9 grid with integers from 1 through 9 so that each integer appears only once in each row, column, and major 3-by-3 square. The grid is partially populated with clues, and your task is to fill in the rest of the grid.
Initial Puzzle
Here is a data matrix B of clues. The first row, B(1,2,2), means row 1, column 2 has a clue 2. The second row, B(1,5,3), means row 1, column 5 has a clue 3. Here is the entire matrix B.
B = [1,2,2;
1,5,3;
1,8,4;
2,1,6;
2,9,3;
3,3,4;
3,7,5;
4,4,8;

Stephen P. Boyd

Home
Teaching
Biography

Research

Books
Papers
Software
People

Classes

ENGR108
EE104
EE263
EE363
EE364a
EE364b
EE365

Stephen P. Boyd – Software

Department of Electrical Engineering, Stanford University

You can find source for many of our group's projects at our github site.

Recent software

- CVX, matlab software for convex optimization
- CVXPY, a convex optimization modeling layer for Python
- CVXR, a convex optimization modeling layer for R
- Convex.jl, a convex optimization modeling layer for Julia
- DCCP, a CVXPY extension for difference of convex programming
- QCQP, a CVXPY extension for nonconvex QCQP
- CVXPortfolio, a Python package for multi-period trading
- GLRM, generalized low rank models
- OSQP, first-order general-purpose QP solver

CVXPY

Star 3,621

Navigation

Install

Tutorial

Examples

API Documentation

FAQ

Citing CVXPY

Contributing

Related Projects

Changes to CVXPY

CVXPY Short Course

License

Welcome to CVXPY 1.1

Convex optimization, for everyone.

We are building a CVXPY community on Discord. Join the conversation!

CVXPY is an open source Python-embedded modeling language for convex optimization problems. It lets you express your problem in a natural way that follows the math, rather than in the restrictive standard form required by solvers.

For example, the following code solves a least-squares problem with box constraints:

```
import cvxpy as cp
import numpy as np

# Problem data.
m = 30
n = 20
np.random.seed(1)
A = np.random.randn(m, n)
b = np.random.randn(m)

# Construct the problem.
x = cp.Variable(n)
objective = cp.Minimize(cp.sum_squares(A @ x - b))
constraints = [0 <= x, x <= 1]
```

```python
import cvxpy as cp
import numpy as np

# Problem data.
m = 30
n = 20
np.random.seed(1)
A = np.random.randn(m, n)
b = np.random.randn(m)

# Construct the problem.
x = cp.Variable(n)
objective = cp.Minimize(cp.sum_squares(A @ x - b))
constraints = [0 <= x, x <= 1]
prob = cp.Problem(objective, constraints)

# The optimal objective value is returned by `prob.solve()`.
result = prob.solve()
# The optimal value for x is stored in `x.value`.
print(x.value)
# The optimal Lagrange multiplier for a constraint is stored in
# `constraint.dual_value`.
print(constraints[0].dual_value)
```

$$A_{30\times 20}x_{20\times 1} = b_{30\times 1}$$

Convex Optimization

Stephen Boyd

Department of Electrical Engineering
Stanford University

Lieven Vandenberghe

Electrical Engineering Department
University of California, Los Angeles

1. Introduction

- mathematical optimization
- least-squares and linear programming
- convex optimization
- example
- course goals and topics

Brief history of convex optimization

theory (convex analysis): ca1900–1970

algorithms

- 1947: simplex algorithm for linear programming (Dantzig)
- 1960s: early interior-point methods (Fiacco & McCormick, Dikin, . . .)
- 1970s: ellipsoid method and other subgradient methods
- 1980s: polynomial-time interior-point methods for linear programming (Karmarkar 1984)
- late 1980s–now: polynomial-time interior-point methods for nonlinear convex optimization (Nesterov & Nemirovski 1994)

applications

- before 1990: mostly in operations research; few in engineering
- since 1990: many new applications in engineering (control, signal processing, communications, circuit design, . . .); new problem classes (semidefinite and second-order cone programming, robust optimization)

Objective function

$A_{m \times n}, m = 30, n = 20$

$x = [x_1, \ldots, x_n]^T$

$min_x ||Ax - b||^2,$

Subject to

$0 \le x_i \le 1, \quad \text{for all } i$

Constrained optimization

```python
# Construct the problem.
x = cp.Variable(n)
objective = cp.Minimize(cp.sum_squares(A @ x - b))
constraints = [0 <= x, x <= 1]
prob = cp.Problem(objective, constraints)

# The optimal objective value is returned by `prob.solve()`.
result = prob.solve()
```

$A_{m \times n}, m = 30, n = 20$

$x = [x_1, \ldots, x_n]^T$

- $min_x \|Ax - b\|^2,$
  Subject to
  $0 \leq x_i \leq 1, \quad \text{for all } i$

For example, the following code solves a least-squares problem with box constraints:

```python
import cvxpy as cp
import numpy as np

# Problem data.
m = 30
n = 20
np.random.seed(1)
A = np.random.randn(m, n)
b = np.random.randn(m)

# Construct the problem.
x = cp.Variable(n)
objective = cp.Minimize(cp.sum_squares(A @ x - b))
constraints = [0 <= x, x <= 1]
prob = cp.Problem(objective, constraints)

# The optimal objective value is returned by `prob.solve()`.
result = prob.solve()
# The optimal value for x is stored in `x.value`.
print(x.value)
# The optimal Lagrange multiplier for a constraint is stored in
# `constraint.dual_value`.
print(constraints[0].dual_value)
```

Polynomial time
Non-polynomial time

N: problem size

Python整數規劃與數獨求解：幫幫外送員 Knapsack

	10	10000	
nlogn	23	92103	
n^2	100	100000000	
N!	3628800	Inf	
2^N	1024	??	

Non-polynomial

幫幫外送員

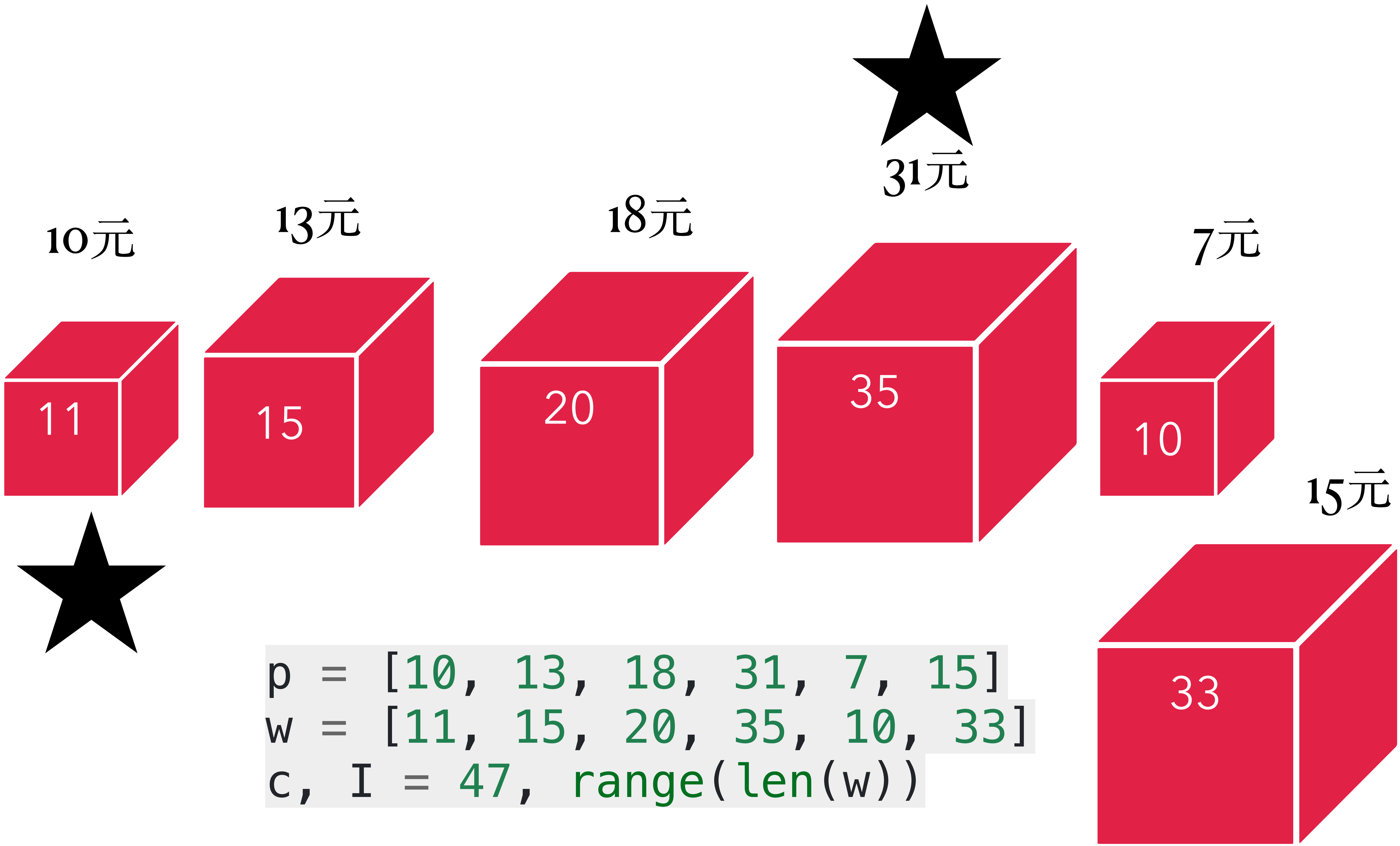
對應的利潤總和最高

```
p = [10, 13, 18, 31, 7, 15]
w = [11, 15, 20, 35, 10, 33]
c, I = 47, range(len(w))
```

從w串列中選擇外送物件，總體積或重量小於c

$x_i = 0$ 代表第i個物件不送

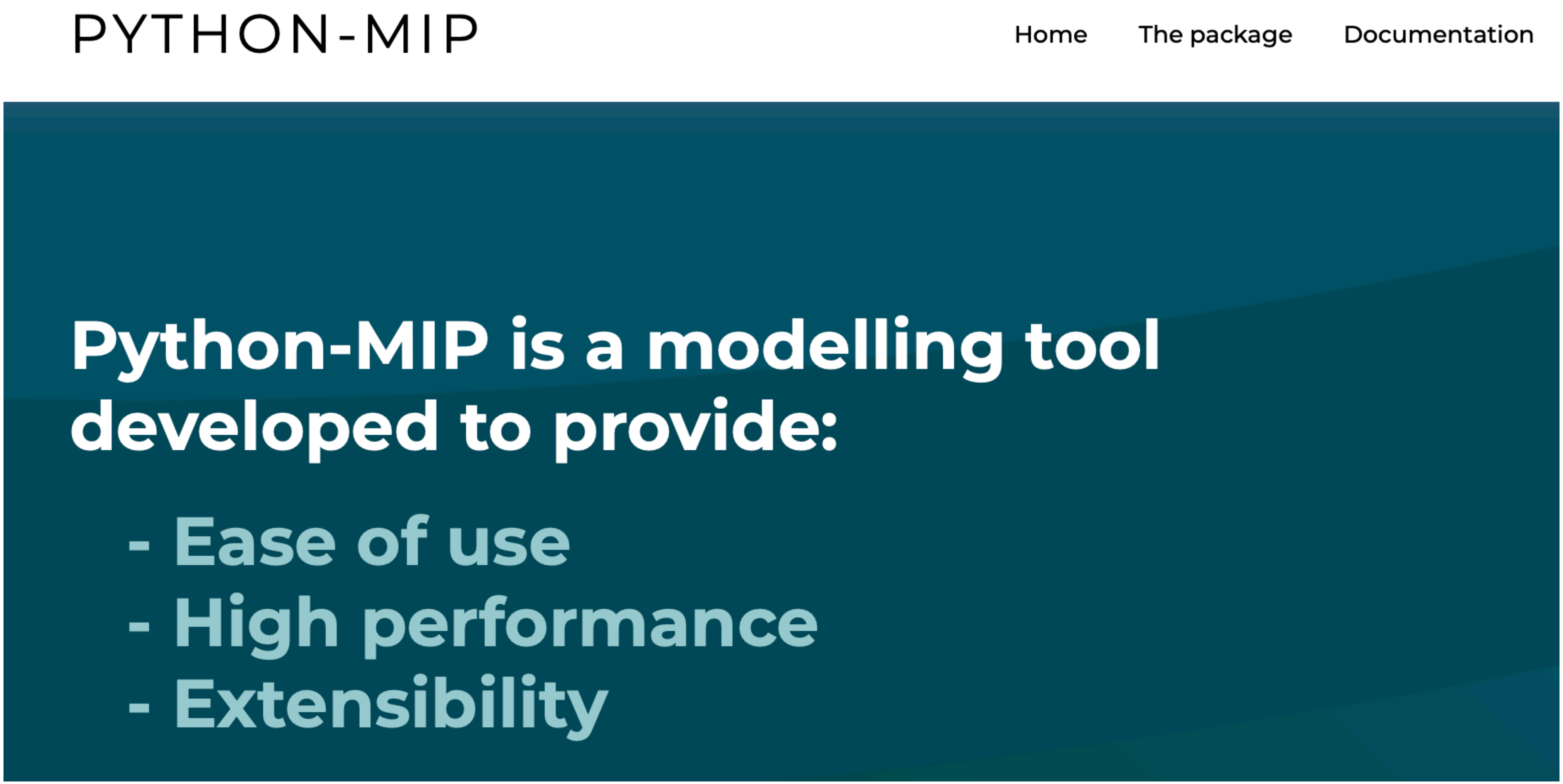
$x_i = 1$ 代表第i個物件要送, $i \in I = \{1,2,3,4,5,6\}$




```
p = [10, 13, 18, 31, 7, 15]
w = [11, 15, 20, 35, 10, 33]
c, I = 47, range(len(w))
```

總重量c = 47

Mip


PYTHON-MIP
Home
The package
Documentation
Python-MIP is a modelling tool developed to provide:
- Ease of use
- High performance
- Extensibility

The 0/1 Knapsack Problem

As a first example, consider the solution of the 0/1 knapsack problem: given a set I of items, each one with a weight w_i and estimated profit p_i, one wants to select a subset with maximum profit such that the summation of the weights of the selected items is less or equal to the knapsack capacity c. Considering a set of decision binary variables x_i that receive value 1 if the i-th item is selected, or 0 if not, the resulting mathematical programming formulation is:

```
p = [10, 13, 18, 31, 7, 15]
w = [11, 15, 20, 35, 10, 33]
c, I = 47, range(len(w))
```

$x_i = 0$ 代表第i個物件不送

$x_i = 1$ 代表第i個物件要送, $i \in I = \{1,2,3,4,5,6\}$

Maximize:

$$\sum_{i \in I} p_i \cdot x_i$$

Subject to:

$$\sum_{i \in I} w_i \cdot x_i \leq c$$

$$x_i \in \{0, 1\} \quad \forall i \in I$$

```python
from mip import Model, xsum, maximize, BINARY

p = [10, 13, 18, 31, 7, 15]
w = [11, 15, 20, 35, 10, 33]
c, I = 47, range(len(w))

m = Model("knapsack")

x = [m.add_var(var_type=BINARY) for i in I]

m.objective = maximize(xsum(p[i] * x[i] for i in I))

m += xsum(w[i] * x[i] for i in I) <= c

m.optimize()

selected = [i for i in I if x[i].x >= 0.99]
print("selected items: {}".format(selected))
```

Maximize:

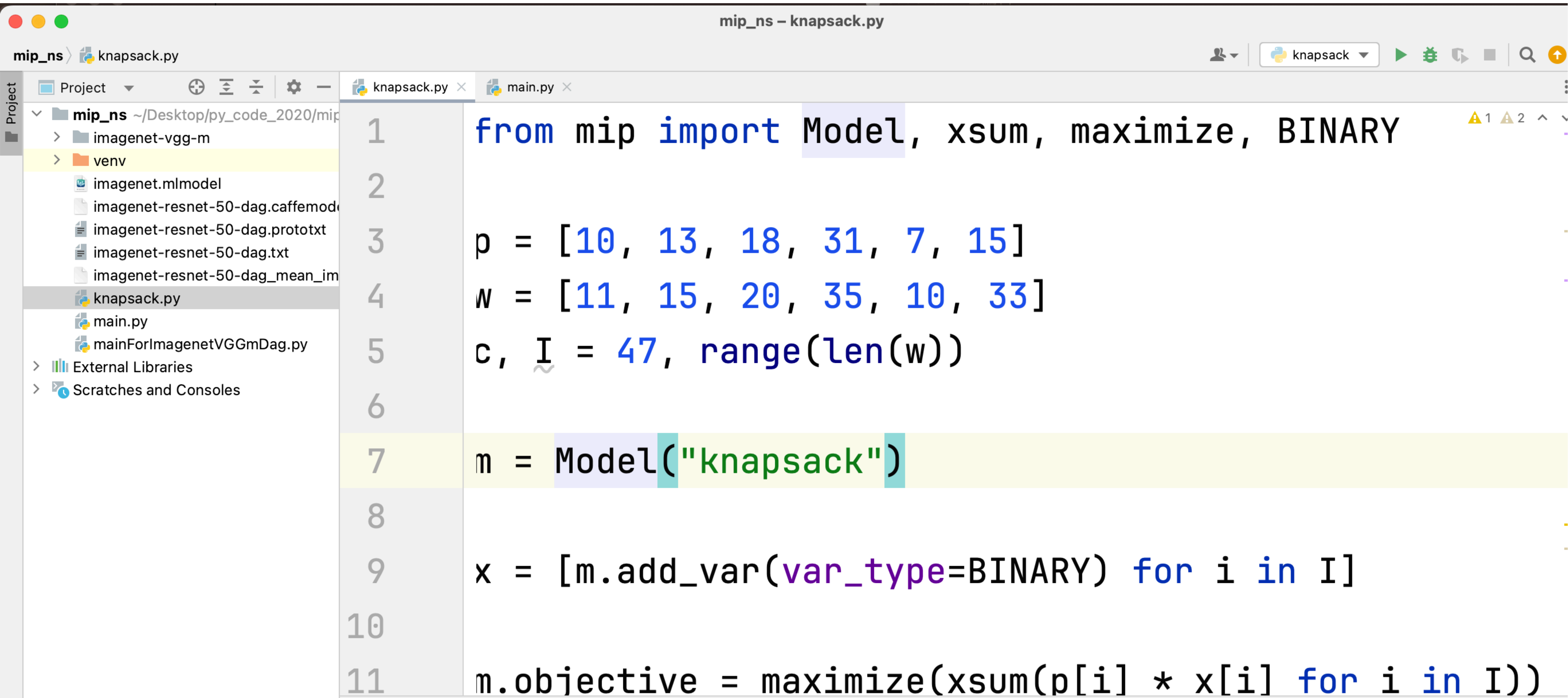
$$\sum_{i \in I} p_i \cdot x_i$$

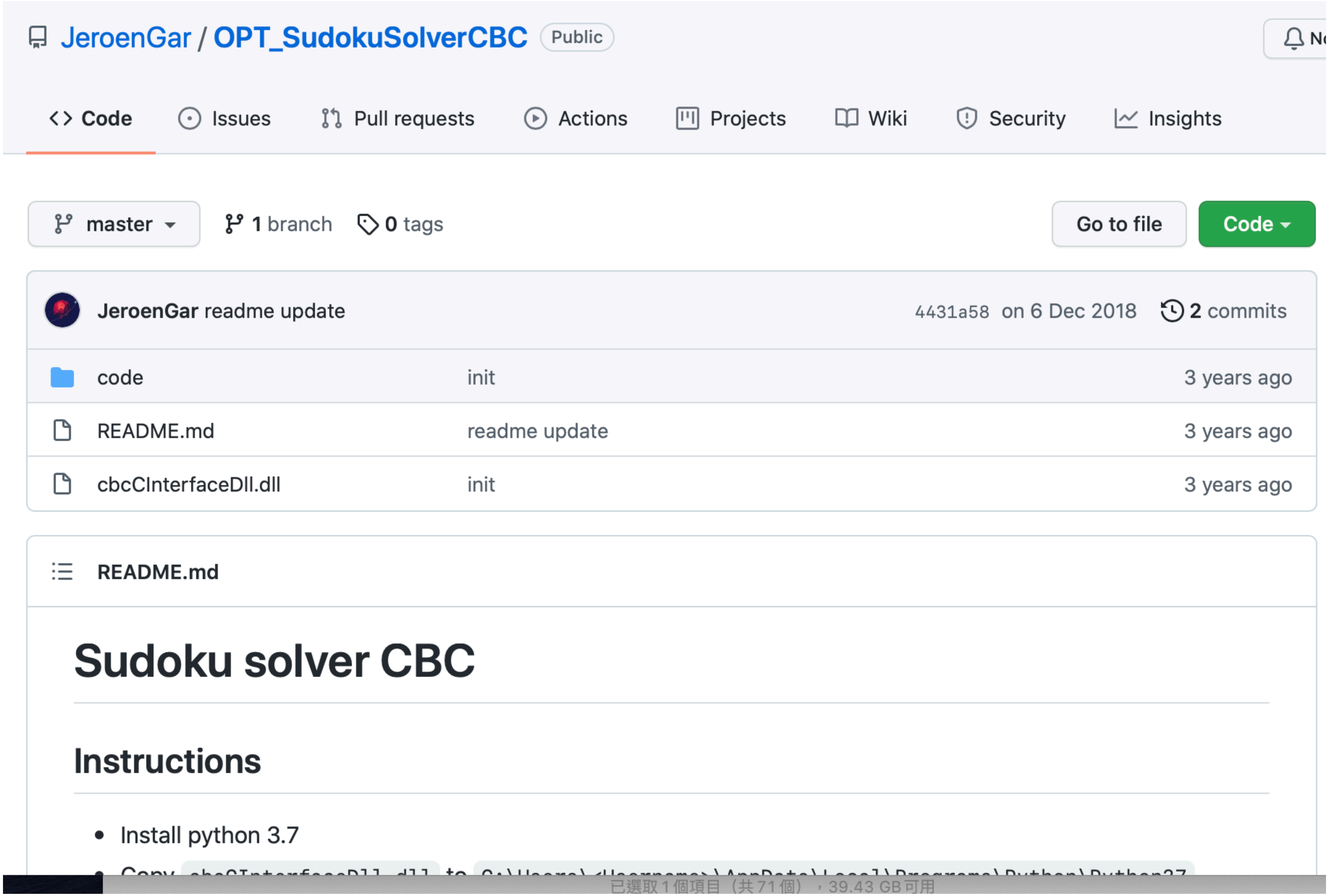
Subject to:

$$\sum_{i \in I} w_i \cdot x_i \leq c$$

$$x_i \in \{0, 1\} \quad \forall i \in I$$

```python
from mip import Model, xsum, maximize, BINARY

p = [10, 13, 18, 31, 7, 15]
w = [11, 15, 20, 35, 10, 33]
c, I = 47, range(len(w))

m = Model("knapsack")

x = [m.add_var(var_type=BINARY) for i in I]

m.objective = maximize(xsum(p[i] * x[i] for i in I))

m += xsum(w[i] * x[i] for i in I) <= c

m.optimize()

selected = [i for i in I if x[i].x >= 0.99]
print("selected items: {}".format(selected))
```


mip_ns – knapsack.py
mip_ns
knapsack.py
knapsack
Project
mip_ns ~/Desktop/py_code_2020/mip
imagenet-vgg-m
venv
imagenet.mlmodel
imagenet-resnet-50-dag.caffemod
imagenet-resnet-50-dag.prototxt
imagenet-resnet-50-dag.txt
imagenet-resnet-50-dag_mean_im
knapsack.py
main.py
mainForImagenetVGGmDag.py
External Libraries
Scratches and Consoles
knapsack.py
main.py
from mip import Model, xsum, maximize, BINARY
p = [10, 13, 18, 31, 7, 15]
w = [11, 15, 20, 35, 10, 33]
c, I = 47, range(len(w))
m = Model("knapsack")
x = [m.add_var(var_type=BINARY) for i in I]
m.objective = maximize(xsum(p[i] * x[i] for i in I))


JeroenGar / OPT_SudokuSolverCBC
Public
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
master
1 branch
0 tags
Go to file
Code
JeroenGar readme update
4431a58 on 6 Dec 2018
2 commits
code
init
3 years ago
README.md
readme update
3 years ago
cbcCInterfaceDll.dll
init
3 years ago
README.md
Sudoku solver CBC
Instructions
Install python 3.7
已選取 1 個項目（共 71 個），39.43 GB 可用

Stephen P. Boyd

Home

Teaching

Biography

Research

Books

Papers

Software

People

Classes

ENGR108

EE104

EE263

EE363

EE364a

EE364b

EE365

Stephen P. Boyd – Software

Department of Electrical Engineering, Stanford University

You can find source for many of our group's projects at our github site.

Recent software

- CVX, matlab software for convex optimization
- CVXPY, a convex optimization modeling layer for Python
- CVXR, a convex optimization modeling layer for R
- Convex.jl, a convex optimization modeling layer for Julia
- DCCP, a CVXPY extension for difference of convex programming
- QCQP, a CVXPY extension for nonconvex QCQP
- CVXPortfolio, a Python package for multi-period trading
- GLRM, generalized low rank models
- OSQP, first-order general-purpose QP solver

1. Introduction

- mathematical optimization
- least-squares and linear programming
- convex optimization
- example
- course goals and topics
- nonlinear optimization
- brief history of convex optimization

$$Ax = b$$

$$A^T A x = A^T b$$

$$\hat{x} = (A^T A)^{-1} A^T b$$

(mathematical) optimization problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq b_i, \quad i = 1, \ldots, m \end{array}$$

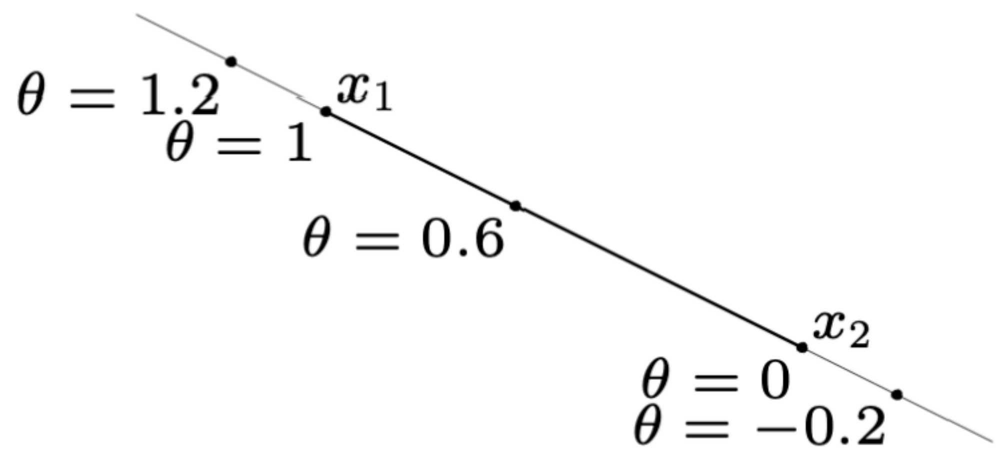
Convex optimization problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq b_i, \quad i = 1, \ldots, m \end{array}$$

Affine set

line through x_1, x_2: all points

$$x = \theta x_1 + (1 - \theta) x_2 \qquad (\theta \in \mathbf{R})$$



affine set: contains the line through any two distinct points in the set

example: solution set of linear equations $\{x \mid Ax = b\}$

(conversely, every affine set can be expressed as solution set of system of linear equations)

Convex set

line segment between x_1 and x_2: all points

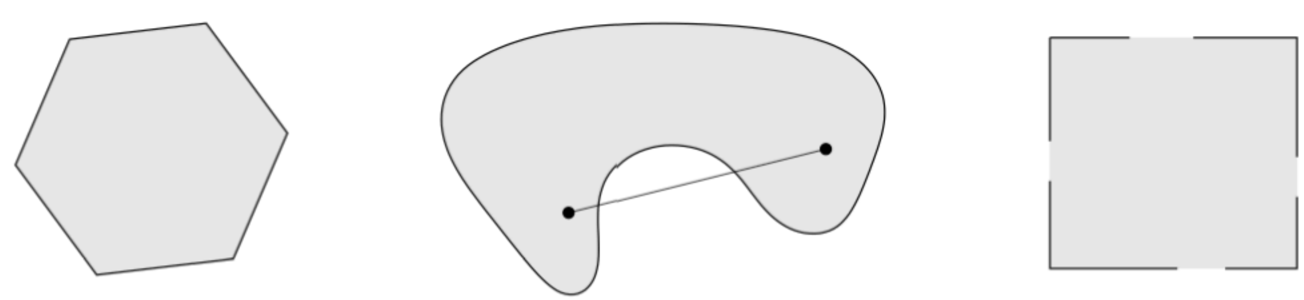
$$x = \theta x_1 + (1 - \theta) x_2$$

with $0 \le \theta \le 1$

convex set: contains line segment between any two points in the set

$$x_1, x_2 \in C, \quad 0 \le \theta \le 1 \quad \Longrightarrow \quad \theta x_1 + (1 - \theta) x_2 \in C$$

examples (one convex, two nonconvex sets)



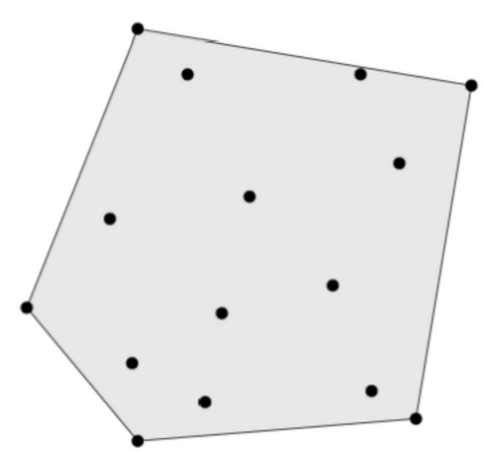
Convex combination and convex hull

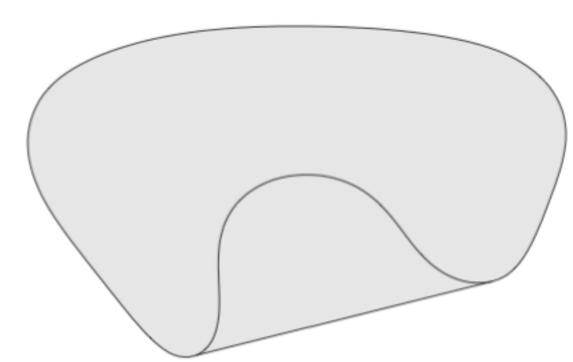
convex combination of x_1,. . . , x_k: any point x of the form

$$x = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_k x_k$$

with $\theta_1 + \cdots + \theta_k = 1$, $\theta_i \geq 0$

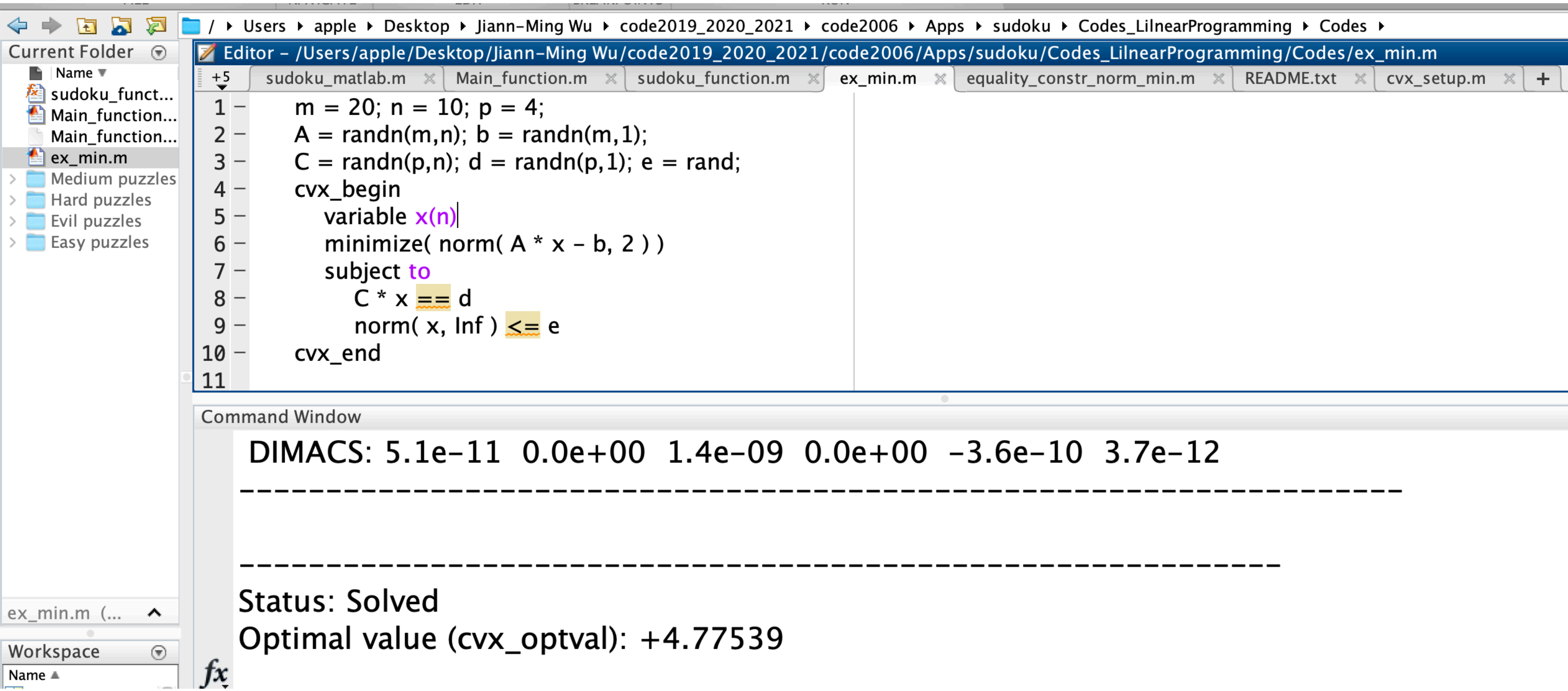
convex hull $\textbf{conv}\, S$: set of all convex combinations of points in S





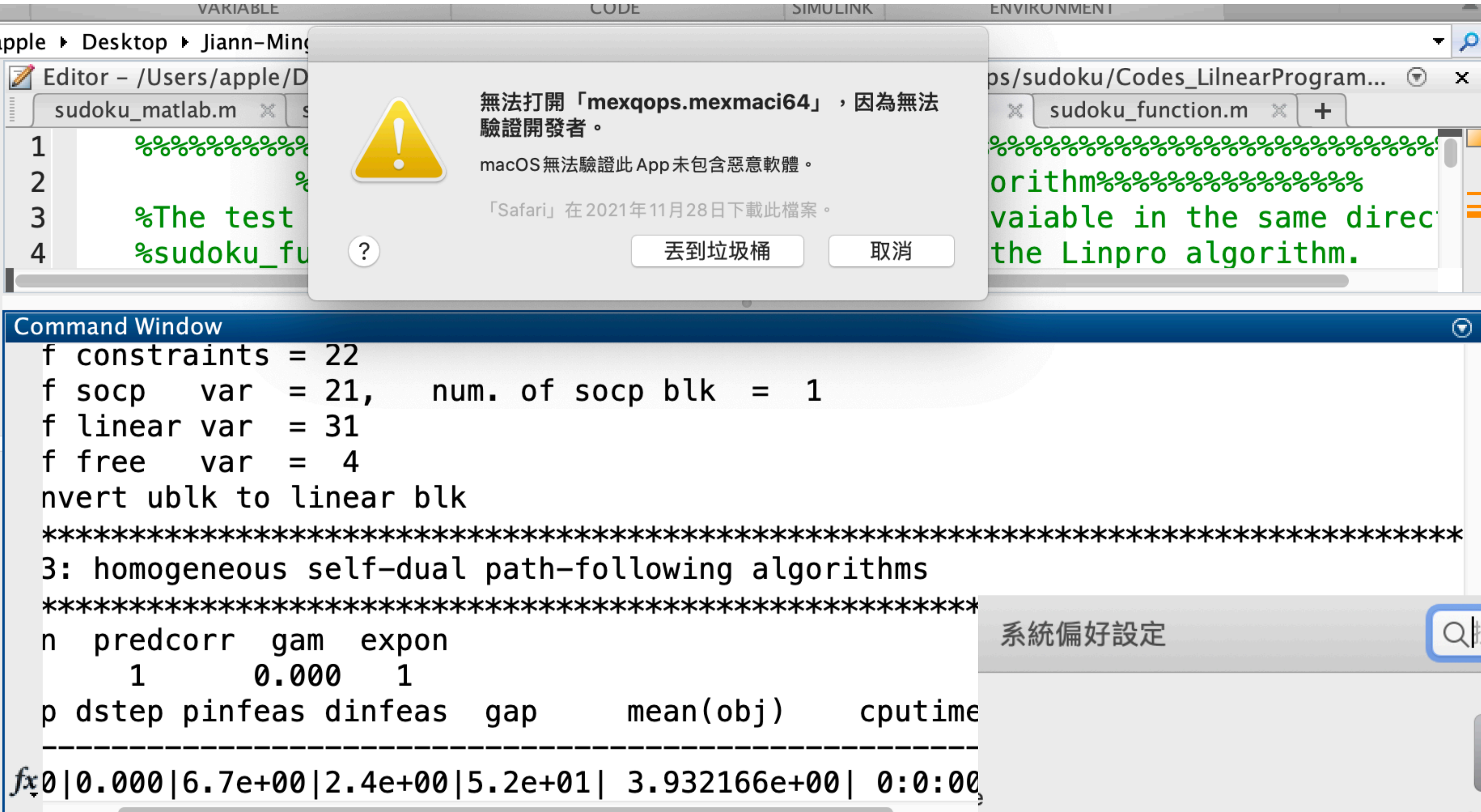
$$
\begin{array}{ll}
\text{minimize} & \|Ax - b\|_2 \\
\text{subject to} & Cx = d \\
 & \|x\|_\infty \leq e
\end{array}
$$

```
m = 20; n = 10; p = 4;
A = randn(m,n); b = randn(m,1);
C = randn(p,n); d = randn(p,1); e = rand;
cvx_begin
    variable x(n)
    minimize( norm( A * x - b, 2 ) )
    subject to
        C * x == d
        norm( x, Inf ) <= e
cvx_end
```


/ Users apple Desktop Jiann-Ming Wu code2019_2020_2021 code2006 Apps sudoku Codes_LilnearProgramming Codes
Current Folder
Name
sudoku_funct...
Main_function...
Main_function...
ex_min.m
Medium puzzles
Hard puzzles
Evil puzzles
Easy puzzles
ex_min.m (...
Workspace
Name
Editor – /Users/apple/Desktop/Jiann-Ming Wu/code2019_2020_2021/code2006/Apps/sudoku/Codes_LilnearProgramming/Codes/ex_min.m
+5
sudoku_matlab.m
Main_function.m
sudoku_function.m
ex_min.m
equality_constr_norm_min.m
README.txt
cvx_setup.m
m = 20; n = 10; p = 4;
A = randn(m,n); b = randn(m,1);
C = randn(p,n); d = randn(p,1); e = rand;
cvx_begin
variable x(n)
minimize(norm(A * x - b, 2))
subject to
C * x == d
norm(x, Inf) <= e
cvx_end
Command Window
DIMACS: 5.1e-11 0.0e+00 1.4e-09 0.0e+00 -3.6e-10 3.7e-12
Status: Solved
Optimal value (cvx_optval): +4.77539

$$
\begin{array}{ll}
\text{minimize} & \|Ax - b\|_2 \\
\text{subject to} & Cx = d \\
& \|x\|_\infty \leq e
\end{array}
$$

```
m = 20; n = 10; p = 4;
A = randn(m,n); b = randn(m,1);
C = randn(p,n); d = randn(p,1); e = rand;
cvx_begin
    variable x(n)
    minimize( norm( A * x - b, 2 ) )
    subject to
        C * x == d
        norm( x, Inf ) <= e
cvx_end
```


無法打開「mexqops.mexmaci64」，因為無法驗證開發者。
macOS 無法驗證此 App 未包含惡意軟體。
「Safari」在 2021年11月28日下載此檔案。
丟到垃圾桶
取消
Editor – /Users/apple/D
sudoku_matlab.m
sudoku_function.m
%The test
%sudoku_fu
vaiable in the same direc
the Linpro algorithm.
Command Window
f constraints = 22
f socp var = 21, num. of socp blk = 1
f linear var = 31
f free var = 4
nvert ublk to linear blk
3: homogeneous self-dual path-following algorithms
n predcorr gam expon
1 0.000 1
p dstep pinfeas dinfeas gap mean(obj) cputime
0|0.000|6.7e+00|2.4e+00|5.2e+01| 3.932166e+00| 0:0:00

系統偏好設定
搜尋
Apple ID
家人共享
Siri
Spotlight
語言與地區
通知
輔助使用
螢幕使用時間
延伸功能
安全性與隱私權
印表機與掃描器
鍵盤
觸控式軌跡板
滑鼠

安全性
設定


安全性與隱私權
搜尋
一般
檔案保險箱
防火牆
隱私權
已設定這位使用者的登入密碼
更改密碼⋯
進入睡眠或開始螢幕保護程式
立即
喚醒電腦需要輸入密碼
螢幕鎖定時顯示訊息
設定鎖定訊息⋯
停用自動登入
允許從以下來源下載的 App：
App Store
App Store 和已識別的開發者
「mexqops.mexmaci64」遭到阻擋無法使用，因為它不是來自已識別的開發者。
強制允許
按鎖頭一下，以進行更改。
進階⋯
?
強制允許

Article

Linear Systems, Sparse Solutions, and Sudoku

February 2010 · Signal Processing Letters, IEEE 17(1):40 - 42
DOI:10.1109/LSP.2009.2032489
Source · IEEE Xplore

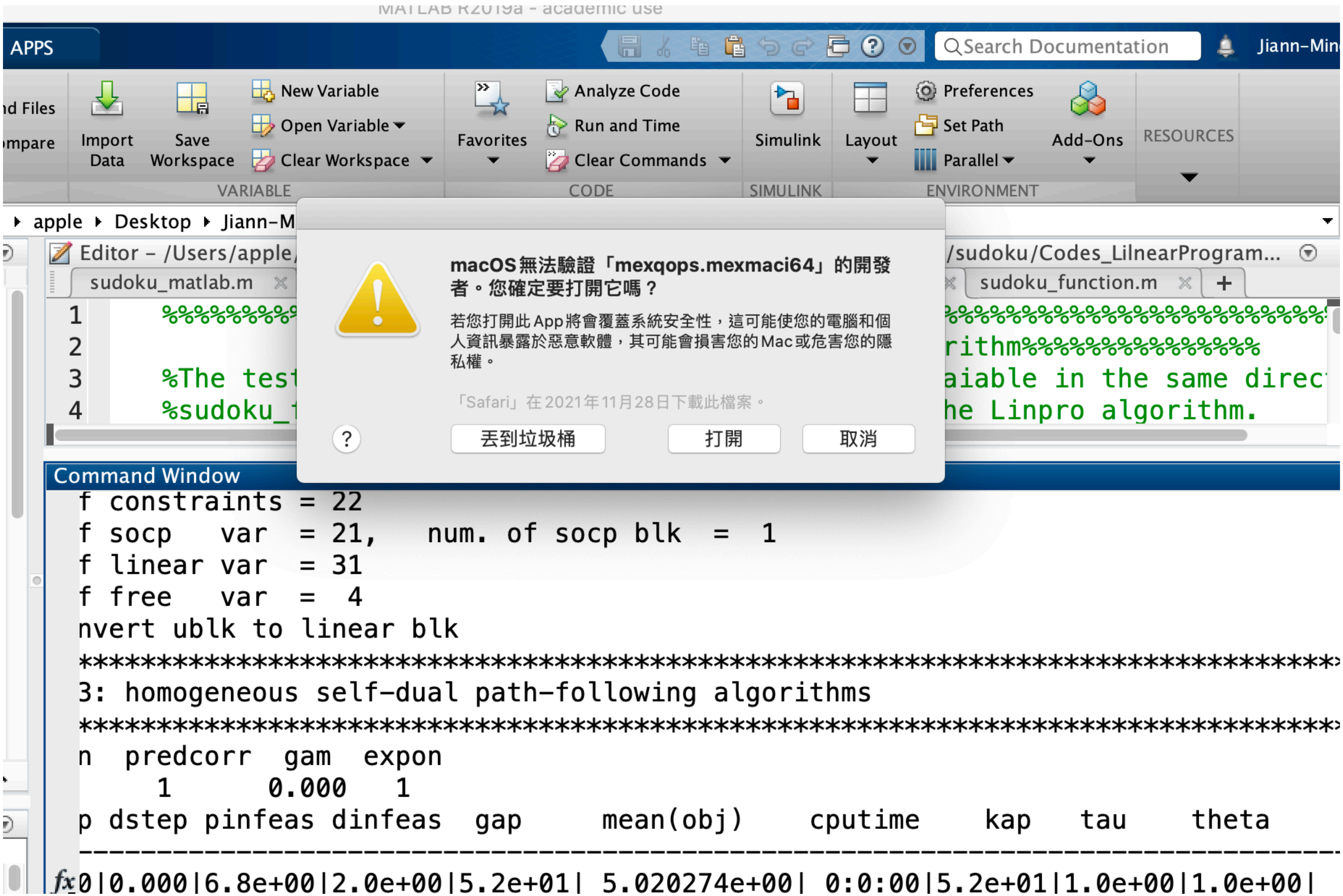
Authors:

Prabhu Babu

Kristiaan Pelckmans
Uppsala University

Petre Peter Stoica
Uppsala University

Jian Li
Beihang University (BUAA)


MATLAB R2019a - academic use
APPS
Search Documentation
Jiann-Min
nd Files
mpare
Import Data
Save Workspace
New Variable
Open Variable
Clear Workspace
VARIABLE
Favorites
Analyze Code
Run and Time
Clear Commands
CODE
Simulink
SIMULINK
Layout
Preferences
Set Path
Parallel
Add-Ons
ENVIRONMENT
RESOURCES
apple ▸ Desktop ▸ Jiann-M
Editor – /Users/apple/
/sudoku/Codes_LilnearProgram...
sudoku_matlab.m
sudoku_function.m
1 %%%%%%%%
2
3 %The test
4 %sudoku_f
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
rithm%%%%%%%%%%%%%%
aiable in the same direct
he Linpro algorithm.
macOS無法驗證「mexqops.mexmaci64」的開發者。您確定要打開它嗎？
若您打開此App將會覆蓋系統安全性，這可能使您的電腦和個人資訊暴露於惡意軟體，其可能會損害您的Mac或危害您的隱私權。
「Safari」在2021年11月28日下載此檔案。
?
丟到垃圾桶
打開
取消
Command Window
f constraints = 22
f socp var = 21, num. of socp blk = 1
f linear var = 31
f free var = 4
nvert ublk to linear blk
****
3: homogeneous self-dual path-following algorithms
****
n predcorr gam expon
1 0.000 1
p dstep pinfeas dinfeas gap mean(obj) cputime kap tau theta
----
fx 0|0.000|6.8e+00|2.0e+00|5.2e+01| 5.020274e+00| 0:0:00|5.2e+01|1.0e+00|1.0e+00|


安全性與隱私權
搜尋
一般
檔案保險箱
防火牆
隱私權
已設定這位使用者的登入密碼
更改密碼⋯
進入睡眠或開始螢幕保護程式
立即
喚醒電腦需要輸入密碼
螢幕鎖定時顯示訊息
設定鎖定訊息⋯
停用自動登入
允許從以下來源下載的 App：
App Store
App Store 和已識別的開發者
「mexexpand.mexmaci64」遭到阻擋無法使用，因為它不是來自已識別的開發者。
強制允許
按鎖頭一下，以進行更改。
進階⋯
?

```
 gap := trace(XZ)       = 6.15e-11
 relative gap           = 1.22e-11
 actual relative gap    = -1.77e-10
 rel. primal infeas     = 1.50e-11
 rel. dual   infeas     = 7.20e-10
 norm(X), norm(y), norm(Z) = 1.8e+00, 4.2e+00, 6.0e+00
 norm(A), norm(b), norm(C) = 2.1e+01, 1.0e+00, 5.2e+00
 Total CPU time (secs)  = 6.06
 CPU time per iteration = 0.36
 termination code       =  0
 DIMACS: 1.5e-11  0.0e+00  7.2e-10  0.0e+00  -1.8e-10  6.8e-12
-------------------------------------------------------------------

------------------------------------------------------------
Status: Solved
Optimal value (cvx_optval): +4.05342
```

```
m = 20; n = 10; p = 4;
A = randn(m,n); b = randn(m,1);
C = randn(p,n); d = randn(p,1); e = rand;
cvx_begin
    variable x(n)
    minimize( norm( A * x - b, 2 ) )
    subject to
        C * x == d
        norm( x, Inf ) <= e
cvx_end

norm(C*x-d)
max(abs(x))<= e
norm(A*x-b)
```

```
>> norm(C*x-d)

ans =

   1.8007e-09
```

```
>> norm(A*x-b)

ans =

    4.0534
```

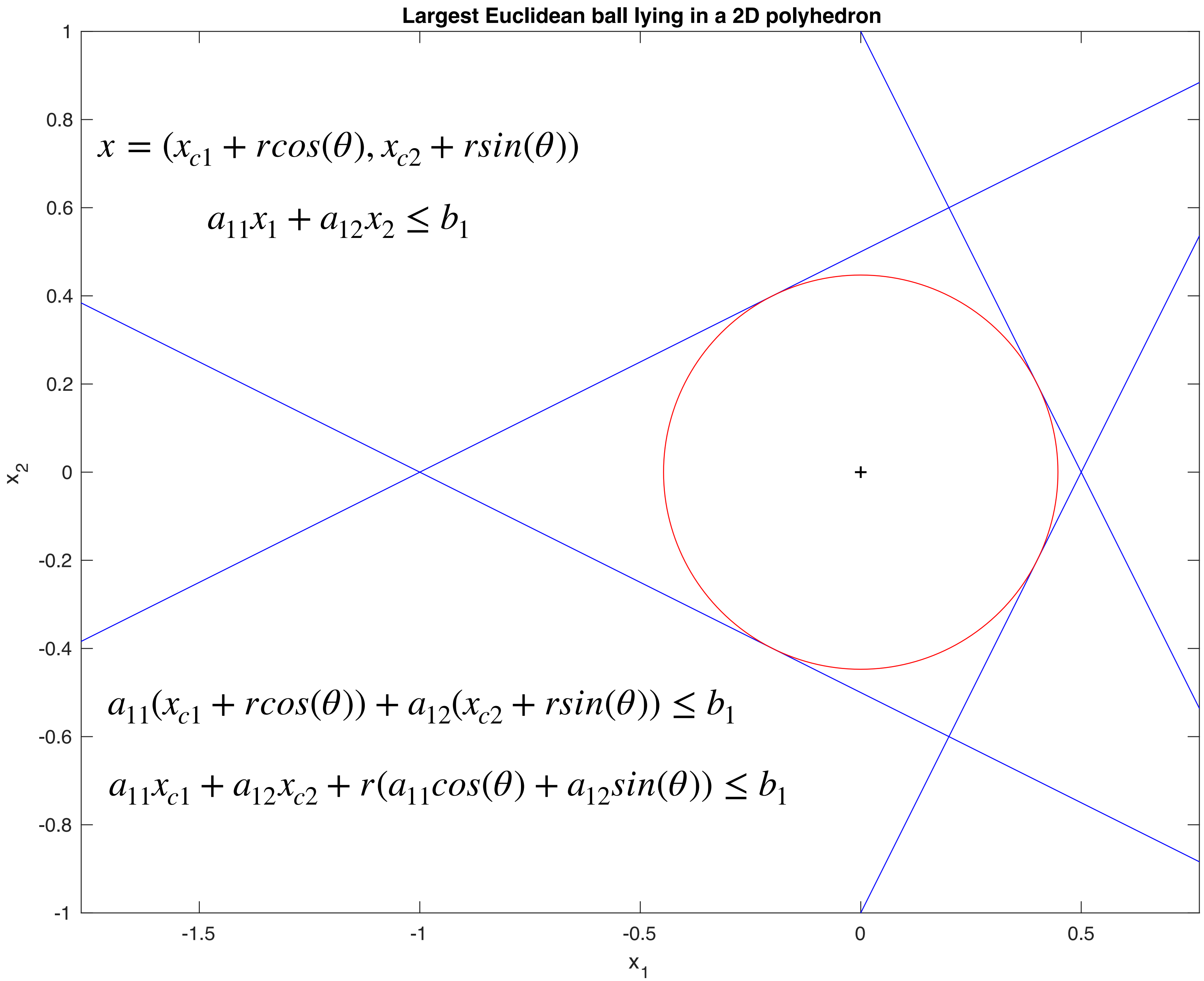
```
>> max(abs(x))<= e

ans =

  logical

   1
```

$$
\begin{aligned}
\text{minimize} \quad & \|Ax - b\|_2 \\
\text{subject to} \quad & Cx = d \\
& \|x\|_\infty \le e
\end{aligned}
$$

```
>> norm(C*x-d)

ans =

   1.8007e-09
```

```
>> norm(A*x-b)

ans =

    4.0534
```

```
>> max(abs(x))<= e

ans =

  logical

   1
```



$$x = (x_{c1} + rcos(\theta), x_{c2} + rsin(\theta))$$

$$a_{11}x_1 + a_{12}x_2 \leq b_1$$

$$a_{11}(x_{c1} + rcos(\theta)) + a_{12}(x_{c2} + rsin(\theta)) \leq b_1$$

$$a_{11}x_{c1} + a_{12}x_{c2} + r(a_{11}cos(\theta) + a_{12}sin(\theta)) \leq b_1$$

$$a_{11}cos(\theta) + a_{12}sin(\theta) = = norm(a_1,2)$$

$$\sqrt{(}a_{11}^2 + a_{12}^2)$$

Section 4.3.1: Compute and display the Chebyshev center of a 2D polyhedron

Jump to: Source code Text output Plots Library index

```
% radius) that lies in a polyhedron described by linear inequalites in this
% fashion: P = {x : a_i'*x <= b_i, i=1,...,m} where x is in R^2

% Generate the input data
a1 = [ 2;  1];
a2 = [ 2; -1];
a3 = [-1;  2];
a4 = [-1; -2];
b = ones(4,1);

% Create and solve the model
cvx_begin
    variable r(1)
    variable x_c(2)
    maximize ( r )
    a1'*x_c + r*norm(a1,2) <= b(1);
    a2'*x_c + r*norm(a2,2) <= b(2);
    a3'*x_c + r*norm(a3,2) <= b(3);
    a4'*x_c + r*norm(a4,2) <= b(4);
cvx_end

% Generate the figure
x = linspace(-2,2);
theta = 0:pi/100:2*pi;
```

$$\min \frac{1}{2} x' * P * x + q' * x + r$$

$$-1 \leq x_i \leq 1, \text{ for i} = 1,\ldots,3$$

$$x_i \in \{1, -1\}$$

$$\min \frac{1}{2} x' * P * x + q' * x + r$$

$$-1 \le x_i \le 1, \text{ for i} = 1,\ldots,3$$

Exercise 4.3: Solve a simple QP with inequality constraints

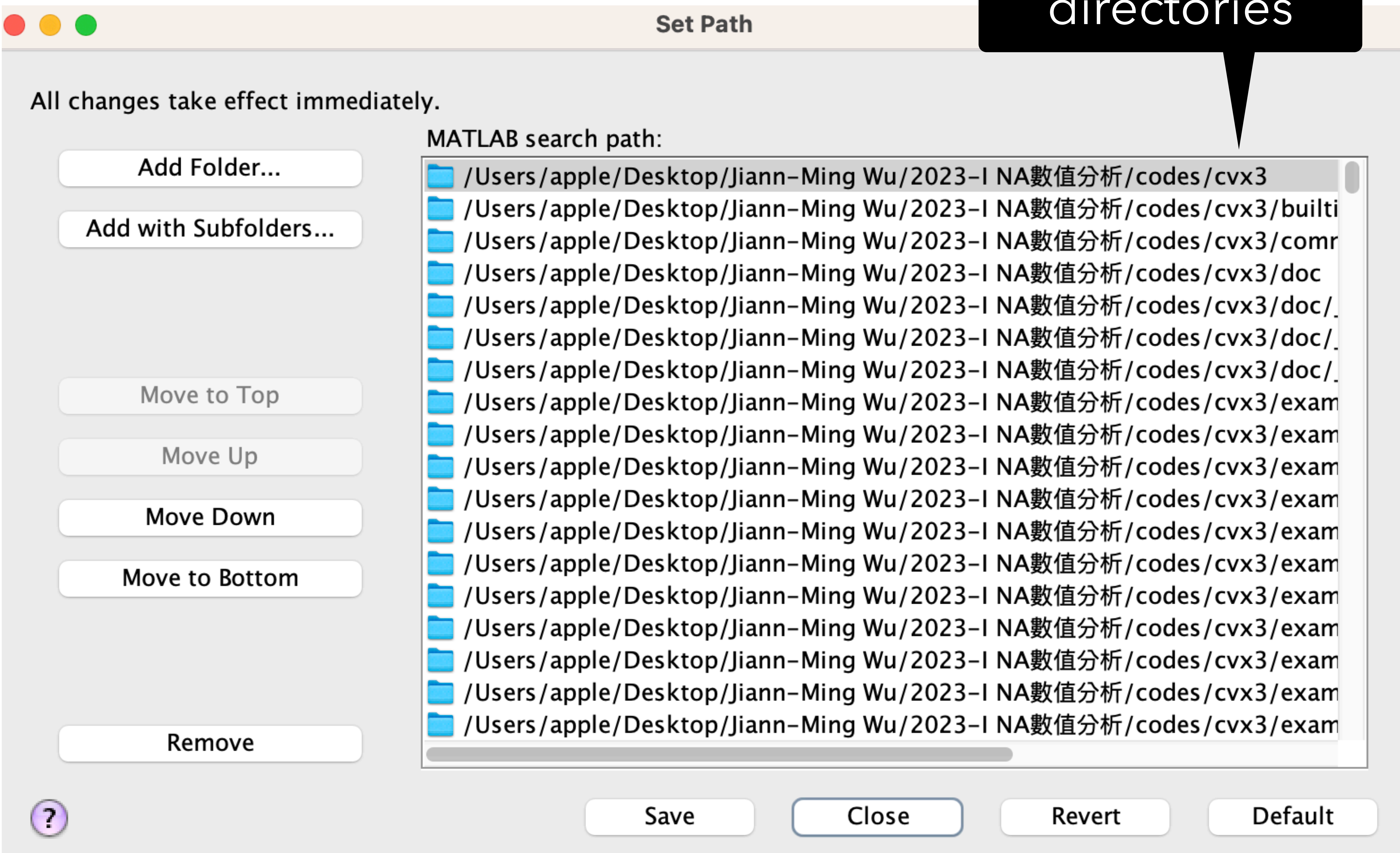
Jump to: Source code Text output Plots Library index

```
% From Boyd & Vandenberghe, "Convex Optimization"
% Joëlle Skaf - 09/26/05
%
% Solves the following QP with inequality constraints:
%           minimize    1/2x'*P*x + q'*x + r
%               s.t.    -1 <= x_i <= 1      for i = 1,2,3
% Also shows that the given x_star is indeed optimal

% Generate data
P = [13 12 -2; 12 17 6; -2 6 12];
q = [-22; -14.5; 13];
r = 1;
n = 3;
x_star = [1;1/2;-1];
```

$A_{m \times n}, m = 30, n = 20$
$x = [x_1, \ldots, x_n]^T$

$min_x \|Ax - b\|^2,$
Subject to
$0 \leq x_i \leq 1, \quad \text{for all } i$

Add with sub-directories

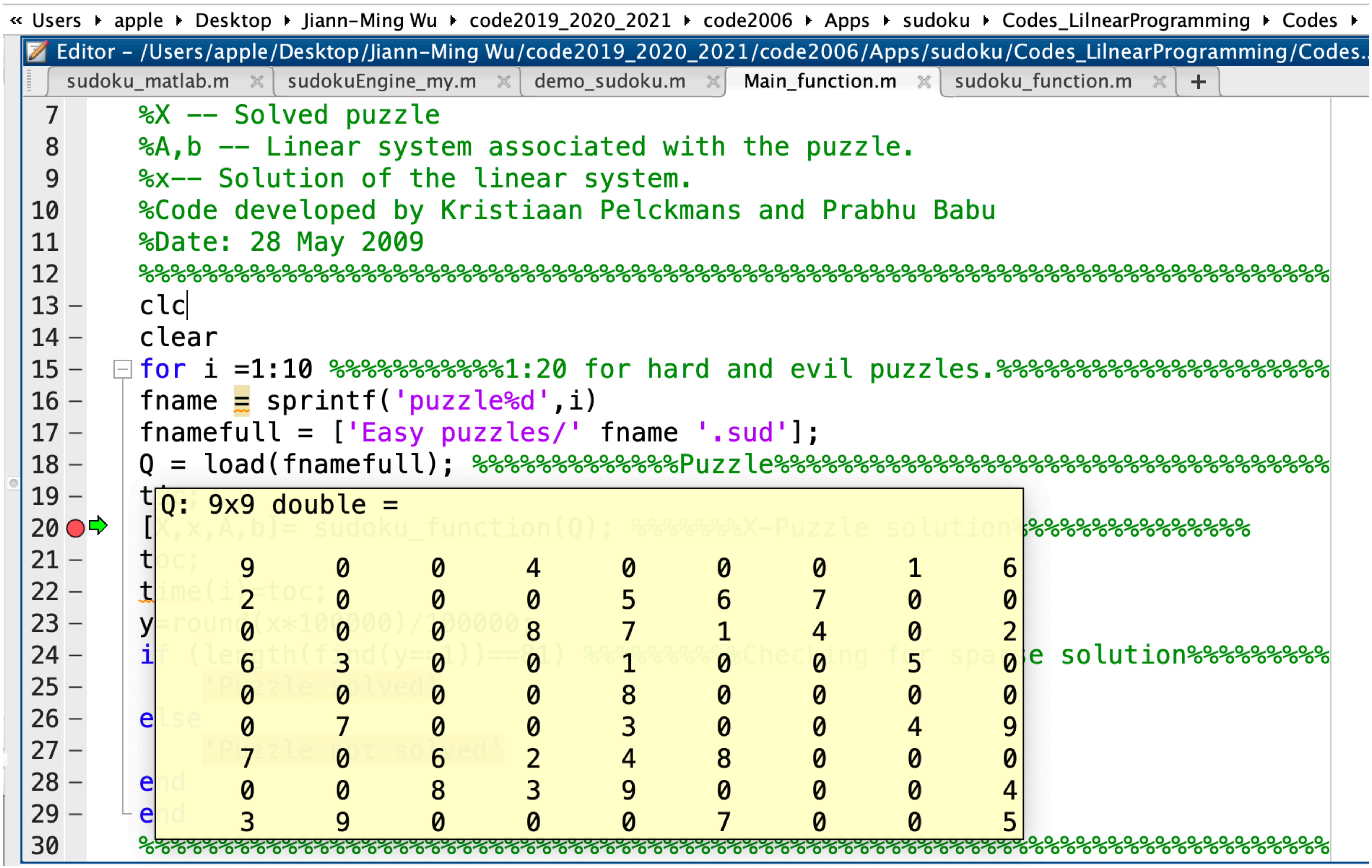


Desktop ▸ Jiann-Ming Wu ▸ code2019_2020_2021 ▸ code2006 ▸ Apps ▸ sudoku ▸ Codes_LilnearProgramming ▸ Codes ▸ Easy puzzles

Editor - /Users/apple/Desktop/Jiann-Ming Wu/code2019_2020_2021/code2006/Apps/sudoku/Codes_LilnearProgramming/Code

sudoku_matlab.m | sudokuEngine_my.m | demo_sudoku.m | Main_function.m | sudoku_function.m | puzzle1.sud

```
9 0 0 4 0 0 0 1 6
2 0 0 0 5 6 7 0 0
0 0 0 8 7 1 4 0 2
6 3 0 0 1 0 0 5 0
0 0 0 0 8 0 0 0 0
0 7 0 0 3 0 0 4 9
7 0 6 2 4 8 0 0 0
0 0 8 3 9 0 0 0 4
3 9 0 0 0 7 0 0 5
```


Users ▸ apple ▸ Desktop ▸ Jiann-Ming Wu ▸ code2019_2020_2021 ▸ code2006 ▸ Apps ▸ sudoku ▸ Codes_LilnearProgramming ▸ Codes ▸
Editor – /Users/apple/Desktop/Jiann-Ming Wu/code2019_2020_2021/code2006/Apps/sudoku/Codes_LilnearProgramming/Codes..
sudoku_matlab.m
sudokuEngine_my.m
demo_sudoku.m
Main_function.m
sudoku_function.m
%X -- Solved puzzle
%A,b -- Linear system associated with the puzzle.
%x-- Solution of the linear system.
%Code developed by Kristiaan Pelckmans and Prabhu Babu
%Date: 28 May 2009
%%%
clc
clear
for i =1:10 %%%%%%%%%%1:20 for hard and evil puzzles.%%%%%%%%%%%%%%%%%%%%
fname = sprintf('puzzle%d',i)
fnamefull = ['Easy puzzles/' fname '.sud'];
Q = load(fnamefull); %%%%%%%%%%%%%Puzzle%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Q: 9x9 double =
9 0 0 4 0 0 0 1 6
2 0 0 0 5 6 7 0 0
0 0 0 8 7 1 4 0 2
6 3 0 0 1 0 0 5 0
0 0 0 0 8 0 0 0 0
0 7 0 0 3 0 0 4 9
7 0 6 2 4 8 0 0 0
0 0 8 3 9 0 0 0 4
3 9 0 0 0 7 0 0 5

Jsers ▸ apple ▸ Desktop ▸ Jiann-Ming Wu ▸ code2019_2020_2021 ▸ code2006 ▸ Apps ▸ sudoku ▸ Codes_LilnearProgramming ▸ Codes ▸

Editor - /Users/apple/Desktop/Jiann-Ming Wu/code2019_2020_2021/code2006/Apps/sudoku/Codes_LilnearProgramming/Codes...

sudoku_matlab.m | sudokuEngine_my.m | demo_sudoku.m | Main_function.m | sudoku_function.m | puzzle1.sud | +

```
function [X,x,Aeq,beq] = sudoku_function(Q0)
m  = size(Q0,1);
m2 = m*m;
m1 = sqrt(m);
nq = m*m*m;
%%%%%%%%%%%%%%%%%%%%%Constraint Matices%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%ROWS: A_row%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Aeq = [];
beq = [];
for j=1:m;
    for k=1:m,
        sudj = [zeros(m,j-1) k*ones(m,1) zeros(m,m-j)];
        Aeq  = [Aeq; sud2vec(sudj,m)'];
        beq = [beq; 1];
    end;
end
```

```
%%%%%%%%%%%%%%%%%%%%COLUMNS:A_col%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for j=1:m;
    for k=1:m,
        sudj = [zeros(j-1,m); k*ones(1,m); zeros(m-j,m)];
        Aeq  = [Aeq; sud2vec(sudj,m)'];
        beq = [beq; 1];
    end;
end
```

```
%%%%%%%%%%%%%%%%%%%%BOXS:A_box%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for j1=1:m1;
    for j2=1:m1,
        for k=1:m,
            sudj = zeros(m);
            sudj((j1-1)*m1+(1:m1),(j2-1)*m1+(1:m1)) = ones(m1)*k;
            Aeq  = [Aeq; sud2vec(sudj,m)'];
            beq = [beq; 1];
        end;
    end;
end
```

```
%%%%%%%%%%%%%%%%%%%CELLS:A_cell%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for j=1:m2,
    con = zeros(1,m2*k);
    con(1,j:m2:end)=1;
    Aeq = [Aeq;con];
    beq=[beq;1];
end
```

```
%%%%%%%%%%%%%%%%%%CLUES:A_clue%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[i1,i2,label]=find(Q0);
for j=1:length(i1),
    sudj = zeros(m);
    sudj(i1(j),i2(j))=label(j);
    Aeq  = [Aeq; sud2vec(sudj,m)'];
    beq = [beq; 1];
end
tic;
```

```
tic;
%%%%%%%%%%%%%%%%%%%%%L1 Minimization%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%Need CVX software to execute the code%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%Download CVX from http://www.stanford.edu/~boyd/cvx/%%%%%%%%
cvx_begin
variables x(size(Aeq,2),1)
minimize norm(x,1)
subject to
Aeq*x == beq;
cvx_end
```

```
X = vec2sud(x,m);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%convert to Sudoku matrix to vector%%%%%%%%%%%%%%%%%%%%
function q=sud2vec(Q,m)
q = zeros(m*m*m,1);
for i = 1:m,
    ind=find(Q(:)==i);
    q((i-1)*m*m+ind)=1;
end
%%%%%%%%%%%%%%%%%convert vector to Sudoku matrix%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Q = vec2sud(q,m)
Qsol = zeros(m,m,m);
Q = zeros(m,m);
for k=1:m,
    Qsol(:,:,k) = reshape(q((m*m*(k-1))+(1:(m*m))),m,m);
end
```

```matlab
%%%%%%%%%%%%%%%%%convert vector to Sudoku matrix%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Q = vec2sud(q,m)
Qsol = zeros(m,m,m);
Q = zeros(m,m);
for k=1:m,
    Qsol(:,:,k) = reshape(q((m*m*(k-1))+(1:(m*m))),m,m);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%Rounding incase of fractional solution%%%%%%%%%%
for j1=1:m,
    for j2 = 1:m,
        [ff,in] = max([Qsol(j1,j2,:)]);
        if ff>.1,
            Q(j1,j2)=in;
        end;
    end;
end
```