利用 MATLAB 解數獨

By Cleve Moler, MATLAB 發明人

人腦和電腦程式兩者是使用非常不同的求解方法來解數獨(Sudoku)問題。用人工手算找出數獨答案的魅力，在於玩家能夠自行發現以及掌握無數的微妙組合，和提供最後解答的線索模式；而這種人腦的模式識別能力和解題過程，是很不容易用電腦程式複製的。由於這個原因，大多數解數獨的電腦程式用的是另一種不同的方式：仰賴電腦近乎無限的運算能力去進行反覆試驗、不斷摸索(trial and error)；在這邊我用的方法，就是用 MATLAB®解題。

解數獨的挑戰

正如你所可能知道，解一則數獨需要填滿 9x9 的方格，而每一行、每一列、以及所分出的 9 個主要 3x3 區塊，必須包含 1 到 9 所有數字。題目一開始有些方格會先填有數字，這些數字為解題的線索。相較於魔方陣(magic squares)和其他數字謎題，數獨並不需要算術，格子中所要填的內容，也可以改成是英文字母或是其他符號。

圖一為題目開始的方格。我特別喜歡這個例子中的對稱性，這個題目是由 University of Western Australia 的 Gordon Royle 所設計的。圖二則是這個例子的解答。

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

圖一：數獨題目範例，線索數字以藍字標示。這個例子中有討人喜歡的對稱性。

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

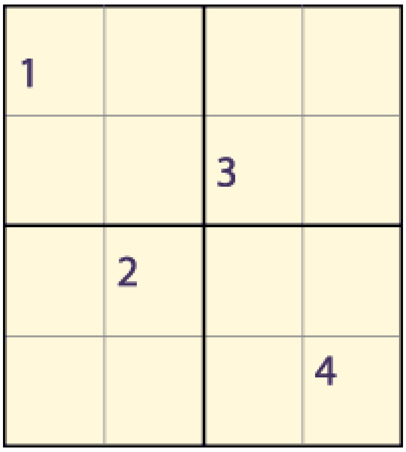
圖二：完整解答。其他空格都已填滿數字，使每行、每列、和主要 3×3 的區塊都包含數字 1 到 9。

用遞迴回溯法(Recursive Backtracking)解數獨

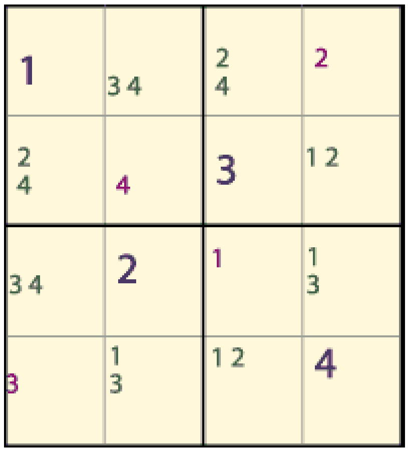
我們的 MATLAB 程式只使用一個樣本(pattern)---單一物件(singletons)，連同基本的電腦科學技術---遞迴回溯法。

要了解該程式如何運行，我們可以先用一個簡單的 4x4 方格(含有 2x2 區塊)為例說明。這類的謎題另被稱作 Shidoku，而不叫 Sudoku(數獨)，因為「Shi」在日文是「四」的意思。

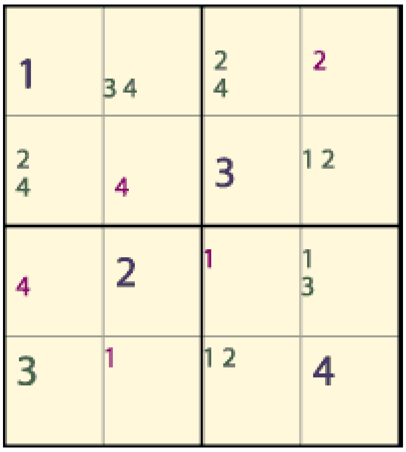
圖三為我們第一個 Shidoku 謎題，圖四到六顯示其解題過程。在圖四中，可能填入的項目或候選數字以小字表示。例如，第二列包含一個「3」，第一行有一個「1」，所以在位置(2,1)的候選數字是「2」和「4」。而所有的空格中，有四個方格僅有唯一的數字答案，這些方格就叫單一物件(singletons)。這個例子可以藉由找出僅有唯一解的單一物件方格，很快地解出答案。在圖五，我們填入其中一個單一物件方格後(位置(4,1)填入「3」)，再重新考量所有候選數字。圖六，我們填滿其他剩餘的單一物件方格後，最後的解答就出現了。



圖三：Shidoku 是只有 4x4 的數獨。



圖一：填入所有的候選數字。紅色字體的數字即為單一物件。



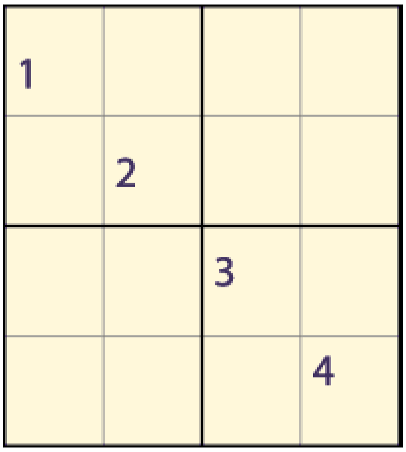
圖五：填入單一物件「3」後，再重新考量所有候選數字。

圖六：填入其他剩餘的單一物件方格，解題完成。

一個簡單的數獨謎題可能被認定為用插入單一物件的方式就能解決。若以這個說法來看，我們的第一個例子是簡單的，但下面的例子並非如此。

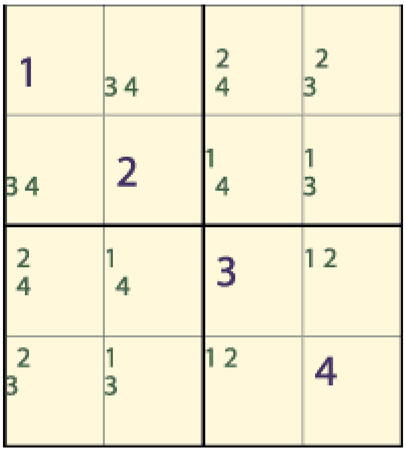
圖七題目所顯示的數字組合可由以下 MATLAB 的宣告產生：

```
X = diag(1:4)
```

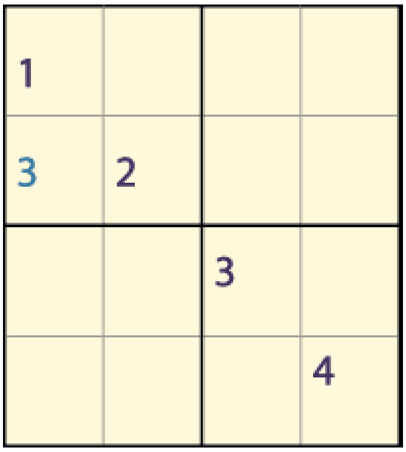


圖七：shidoku(diag(1:4))

因為圖八的例子中，無法找到單一物件，我們將使用遞迴回溯法來解題。我們挑選一個空格，一開始先填入它的其中一個候選數字。而每個候選數字在 MATLAB 裡我們用 X(:)表示(例如：X(1)=1, X(2)=2…)，並根據數字大小的順序去嘗試每一個可能數字。我們在位置(2,1)方格填入「3」之後，變成一個新的謎題(圖九)，接著遞迴呼叫這個程序。



圖八：所有的候選數字。完全沒有單一物件的存在。



圖九：開始先填入「3」後，變成一個新題目，接著開始回溯。

新的謎題很容易解，結果如圖十所示。但是，這個答案取決於我們呼叫遞迴之前的選擇，其他的選擇，可能產生不同的結果。對於這個簡單的對角線初始條件，有兩個可能的答案，恰好是矩陣的互相轉置。既然這題的解答不只唯一解，圖七的範例就不是有效的題目。

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

圖十：範例解答。但這個答案並非唯一解，它的轉置為另一解。

.

存在性和獨特性

作為數學家，我們努力追求去證明每個問題都有一個答案，而且是獨一無二的。在數獨的問題中，從最初的線索不容易去確定答案的存在性與獨特性。例如，圖一的題目，如果我們要填入一個「1」、「5」或「7」在(1,1)位置的方格，行、列和區塊的條件將可以滿足，但是接著形成的新謎題就找不到解。若是在報紙上出現這樣的題目，將會是十分令人沮喪。

回溯產生了許多不可能的答案配置。當遇到方格已沒有任何候選數字時，執行程序就會終止遞迴，而這樣的謎題就無解。

獨特性是一個可望而不可及的屬性。大部分數獨的描述並沒有規定只能有一種解，所以再次令人沮喪的，有可能會發現還有和所提供的解答不一樣的答案存在。有一些在 MATLAB Central 上分享的謎題產生程式並沒有檢查獨特性；而我唯一知道檢查獨特性的方式，是徹底詳盡列舉出所有可能的解。

求解數獨的演算法

我們的 MATLAB 程式只需要四個步驟：

1、填入所有的單一物件

2、如果方格內沒有任何候選數字就退出

3、空的方格內先填入暫定值

4、遞迴地呼叫程序

關鍵的內部函式是候選數字。每個空格以 z = 1:9 開始，並把有關的行、列和區塊已出現的數字，改成 0；剩下不為 0 的即為候選數字。例如，考慮圖一中位置(1,1)方格的解，一開始：

z = 1 2 3 4 5 6 7 8 9

第一列的值將 z 改為

z = 1 0 0 0 5 6 7 8 9　(因為第一列出現 2,3,4 的數值，所以 2,3,4 數值位置的地方改為 0)

然後第一行的值再將 z 改為

z = 1 0 0 0 5 0 7 0 9　(因為第一行出現 4, 6, 8 的數值，所以 4, 6, 8 數值位置的地方改為 0)

分成的 3x3 區塊中，(1,1)區塊條件並沒有再造成更多的改變，所以這個空格的候選數字為

C{1,1} = [1 5 7 9]

難解的謎題

圖一的謎題其實是很難解的，無論是人工手算方式或是用電腦解題。圖十一和圖十二表示計算過程的片刻情形。一開始，由於沒有單一物件，所以第一個遞迴的步驟立即發生。我們嘗試在(1,1)方格填入「1」；圖十一顯示在經過 22 個步驟後，第一行的所有空格都被填滿，但是離最後答案還有一段很長的路。而經過 3,114 的步驟後，遞迴試著將「5」填入(1,1)空格；到第 8,172 個步驟，嘗試填入「7」。

圖十一：求解過程到第 22 個步驟的情形。回溯中暫時的數字選擇以淺藍色字體顯示，而在這些選擇下出現的單一物件用綠色字體表示。(1,1)方格「1」是錯誤的選擇。

圖十二：在 14,781 個步驟後，出現了接近最後答案的解，但是卻不可能再繼續，因為在(1,9)方格已找不到候選數字。「7」也是(1,1)方格的錯誤選擇。

圖十二顯示經過 14,781 步驟後的情況，似乎已接近正確答案，81 個格子中有 73 個已分配好數字。但是將第一列和最後一行放在一起來看，包含了所有從 1 到 9 的數字，所以沒有數字可留

給右上角(1,9)空格；由於這個空格沒有任何候選數字存在，遞迴就終止。最後，到了 19,229 個步驟，我們嘗試了「9」在第一個方格；填入「9」是一個好主意，因為再經過不到 200 個步驟，也就是到第 19,422 步驟時，程式已找到如圖二所示的答案。這比大多數謎題需要更多的步驟。

Solving Sudoku Using Recursive Backtracking

```
function X = sudoku(X)
% SUDOKU    Solve Sudoku using recursive backtracking.
%      sudoku(X), expects a 9-by-9 array X.
  % Fill in all “singletons”.
  % C is a cell array of candidate vectors for each cell.
  % s is the first cell, if any, with one candidate.
  % e is the first cell, if any, with no candidates.
  [C,s,e] = candidates(X);
  while ~isempty(s) && isempty(e)
       X(s) = C{s};
       [C,s,e] = candidates(X);
  end
  % Return for impossible puzzles.
  if ~isempty(e)
       return
  end
  % Recursive backtracking.
  if any(X(:) == 0)
       Y = X;
       z = find(X(:) == 0,1);       % The first unfilled cell.
       for r = [C{z}]                   % Iterate over candidates.
            X = Y;
            X(z) = r;                        % Insert a tentative value.
            X = sudoku(X);                % Recursive call.
            if all(X(:) > 0)            % Found a solution.
                 return
            end
         end
     end
% -----------------------------
     function [C,s,e] = candidates(X)
          C = cell(9,9);
          tri = @(k) 3*ceil(k/3-1) + (1:3);
```

```
        for j = 1:9
            for i = 1:9
                if X(i,j)==0
                    z = 1:9;
                    z(nonzeros(X(i,:))) = 0;
                    z(nonzeros(X(:,j))) = 0;
                    z(nonzeros(X(tri(i),tri(j)))) = 0;
                    C{i,j} = nonzeros(z)';
                end
            end
        end
 L = cellfun(@length,C);     % Number of candidates.
 s = find(X==0 & L==1,1);
 e = find(X==0 & L==0,1);
 end % candidates
end % sudoku
```

數獨的起源

大多數的人都以為數獨起源於日本，事實上是美國人的發明。第一次出現在 1979 年的 Dell Puzzle Magazine 雜誌上，當時叫做 Number Place；1984 年一家日本出版商 Nikoli 將它介紹到日本，並給了一個新名字--數獨(Sudoku)。到了 2004 年英國泰晤士報開始刊載數獨謎題，很快地就流行到美國和世界各地。

Products Used

- MATLAB

Resources

- Experiments with MATLAB
- Sudoku Squares and Chromatic Polynomials
- Strategy Families