Linear Algebra Semester Project

By Tim VanAusdal and Erik Martinez

## Linear Algebra and Sudoku

For our project, Tim and I set off to see if we could solve Sudoku puzzles using what we learned in Linear Algebra. Particularly, what we learned about matrices and row reduction.

## Method 1: 81 x 82 Matrix

To provide a little context the game sudoku is a puzzle game in which you're given a collection of numbers and you're trying to fill in every box. It's typically in the form of a 9x9 puzzle where there's a box in every location and the box may or may not have a value. It's also divided into 9 smaller boxes with three boxes on each level and 3 levels total meaning that there are 9 large boxes with 9 entries in each of them. The puzzle must follow the constraint that within each large box, column, and row there will only be one occurrence of each number from 1-9 where all of them are used in each column, box, and/ or row but they're used exactly once. Knowing this we know that all columns, rows, and large boxes will add up to 45 as 1+2+3+4+5+6+7+8+9=45. Knowing this information, we will attempt to solve a sudoku puzzle using only linear algebra and these relationships. As one of the people in our group (Tim) is a computer science major we aimed to write a program that would solve a sudoku puzzle using only linear algebra.

Our first objective was to find if we could solve an entire 9 x 9 or 4x4 sudoku puzzle by row reducing a matrix with entries corresponding to the direct entries of the puzzle, and relationships between the entries, such as how a row needs to add up to equal 45, and the other constraints that make a sudoku puzzle what it is. We set up a full matrix and had each variable in

the matrix correspond to an entry in the puzzle, so that totaled at 81 variables to work with. To set up 82 different columns and 81 different rows we had 81 of the columns represent a certain variable, for example, column 1 represented the item in row 1 column 1 of the puzzle, and we had 81 rows where from the 27 equations that we could gather from the constraints of the sudoku puzzle we would use relationship equation 3 times and would represent the members of the equation in the appropriate column. In the 82$^{nd}$ column we represented what each equation was equal to. For there to be a unique solution, which is what makes a sudoku puzzle, we would need all 81 pivot columns, which would require our matrix to be 81 rows tall, and 82 columns wide. For such a large matrix to produce a unique solution, that means it would need 81 unique constraints to produce 81 pivot columns. Despite our best efforts of trying many combinations of constraints, including having many of them be the numbers of what the finished puzzle is supposed to look like, we weren't able to get a matrix with a determinant that wasn't 0. In addition to this, we were unable to consistently predict the number of nonzero valid equations that we would have so, though we may end up with 81 for one puzzle, we may end up with 80, 79, or any other number of non-zero equations that we could use which led us to the conclusion that this wouldn't be viable. In addition, the way that we even divided up which equations would be used to represent each variable would be difficult. After this failed, we realized that if we wanted to have something to show for this project, we would need to shoot slightly smaller and take a different approach to our goal.

## Method 2: a simplification of the Matrix and a 27x82 Matrix

Though we originally attempted to solve the sudoku puzzle using an 81x82 matrix that we could also attempt to calculate a determinant with, after determining that we wouldn't be able to consistently produce a useful determinant or the desired 81 pivot columns consistently we

decided to simplify the problem to use a 27 by 82 matrix and to attempt to solve it using the relationships we could derive from reducing the matrix with 82 columns and 27 rows rather than it being 82x81. We also decided to attempt to make it so that you could solve a sudoku puzzle of any size (we only experimented with 4x4 and 9x9 though) where the dimensions of the matrix could be represented as having dimension*dimension+1 columns and dimension*3 rows because there are 3 different constraints that you can use to gather information. Though the 82 by 81 matrix may technically give more information all information within it can be derived from other equations as it would include 3 copies of each equation so because of the potential issues with maintainability and the lack of a practical application we decided to simplify it. In doing this we would change the goal of this project from solving the sudoku puzzle using only linear algebra and the inherent relationships within the puzzle to determining if the relationships could be applied using principles from linear algebra. We wrote a computer program based on this.

## Method 3: Computer Programing and Simple Relationships

After exploring these ideas, we decided to change our approach in a few ways that could potentially assist us. The biggest way was that we would stop doing these matrices completely by hand and would instead create a computer program to solve these matrices for us. We also decided to experiment with smaller matrices. Instead of just using a giant matrix to solve a 9 x 9 sudoku puzzle, we also explored what we could do with a 4x4 sized sudoku puzzle. Because we maintained the concept of having each variable we were solving for be an entry in the puzzle, these changes resulted in smaller matrices and fewer equations to worry about. We found, however, that our solver and row reduction revealed, not direct answers, but relationships between the entries and that these relationships could be used to solve for a specified location assuming that you knew the

other values. For example, the equation r1c1-r4c4=-1 could be used to solve for r1c1 (row 1 column 1) or r4c4 assuming that you had the value of the other one.

Method 4: Applying the information from the matrix and from the puzzle to solve it

After successfully completing this step we would then attempt to apply this information to solve it. We configured the project to print out the inherent relationships between elements but what we found was that most of the variables ended up being in terms of free variables which would mean that we would have to find a unique solution to the problem using a matrix with infinitely many solutions. At this point in time it would print out a lot of different relationships between elements that, while valid, weren't an actual solution. For example, while it may write out relationships among several variables it would rarely if ever find the actual value of a variable using this method. From here we would attempt to solve it in order to find that unique solution by only using the relationships that we gathered and the constraints on the puzzle itself. For example, if r1c4+r2c3=5 and we can see that r1c4 can't be 1,2,or 3 then it means that r1c4=4 and that r2c3=1 . We had some success using this approach with 4x4 matrices which are much less complex and easier to work with but while trying to program it we ran into the issue that while it was able to find valid combinations to the equation r1c4+r2c3=5 it would sometime find multiple valid entries for r1c4 and r2c3 where they would add up to 5 and there were multiple combinations where they wouldn't violate the constraints of sudoku. This is one of several potential challenges that, while it may be possible to overcome, we weren't able to address.

Information about the Program:

The program was written in the programming language of java and all of the necessary information is contained within one class. The package for the assignment is Sudoku and the

class name is SudokuMain. It required user input on the dimensions of the puzzle (4 if 4x4 and 9 if 9x9) and it would then require the user to input the puzzle on one line where they would either input the value itself or they would put a 0 to represent that it was unsolved. This puzzle

000260701
680070090
190004500
820100040
004602900
050003028
009300074
040050036
703018000

Would have to be inputted by specifying a dimension size of 9 and by inputting the line: 0 0 0 2 6 0 7 0 1 6 8 0 0 7 0 0 9 0 1 9 0 0 0 4 5 0 0 8 2 0 1 0 0 0 4 0 0 0 4 6 0 2 9 0 0 0 5 0 0 0 3 0 2 8 0 0 9 3 0 0 0 7 4 0 4 0 0 5 0 0 3 6 7 0 3 0 1 8 0 0 0

We would try printing out a lot of information based on this and after this is inputted we would print out what that line of code would look like as a sudoku puzzle, what it looked like as a matrix in its base form, what a reduced form of the matrix may look like by using the standard algorithms for row reduction(though not necessarily in echelon form), a slightly more concise version of that, and then it would print out the relationships themselves. For example, for the puzzle given in the examples the exact process to input and what would be printed out would look like this.

```
input the desired size (if 2x2 for each box put 4 if 3x3 input 9
4
4
input the sudoku puzzle
1 0 4 0 0 0 0 3 0 0 0 4 0 2 0 0
1040
0003
0004
0200
```

Matrix

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 7 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 6 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 8 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 9 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 8 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 6 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 3 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 9 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 8 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 6 |

Maple command:

```
interface(rtablesize=18);
Q:=<<0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5>|<0, 0, 0, 0,
1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7>|<0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
1, 0, 0, 0, 0, 0, 6>|<0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
8>|<0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 9>|<0, 1, 0, 0, 0,
1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 8>|<0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
0, 0, 0, 1, 0, 6>|<0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
3>|<0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9>|<0, 0, 0, 1, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3>|<0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
0, 1, 0, 0, 0, 8>|<0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
6>>;
ReducedRowEchelonForm(Q);
```

| r1c1 | r1c2 | r1c3 | r1c4 | r2c1 | r2c2 | r2c3 | r2c4 | r3c1 | r3c2 | r3c3 | r3c4 | r4c1 | r4c2 | r4c3 | r4c4 | ans |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 2 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | -1 | -1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 8 |
| 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | -1 | -6 |
| 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | -3 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | -1 | -1 | -6 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

```
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 2
0 0 0 0 1 0 0 0 0 -1 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 1 1 0 0 0 0 -1 -1 0
```

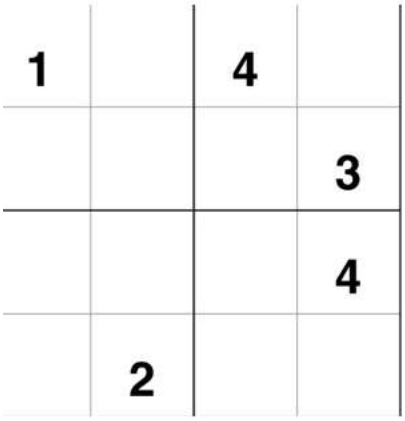
```
 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 8
 0 0 0 0 0 -1 0 0 0 -1 0 0 0 0 0 -1 -6
 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 -1 -3
 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 -1 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 -1 -1 -6
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0


 r1c2-r4c4= 2
 r2c1-r3c2= 1
 r3c1+r3c2-r4c3-r4c4= 0
 r4c1+r4c3+r4c4= 8
-r2c2-r3c2-r4c4=-6
-r1c4-r4c4=-3
 r2c3-r4c4= 0
-r3c3-r4c3-r4c4=-6
```

We used a 4x4 because this can be displayed much easier in word though a 9x9 matrix would print something that looks very similar.

## Examples and Findings

For the first example, we will be referring to this simple 4x4 sudoku puzzle:




We solved this by hand in order to determine if the approach of using the constraints and relationships was valid and we were successful in our attempt to solve it. We did most of the

analysis by hand using the relationships we were able to infer from the puzzle. Within the program we configured it to do several things: take in an inputted puzzle optimized for 4x4 and 9x9 puzzles and then to create a matrix that's the dimension*3 by dimension*dimension+1 where dimension is the size of the array (it ended up being 82 columns with 27 rows for a 9x9 and 17 columns with 12 rows for a 4x4 matrix) where the 27 (or 12) rows will represent the relationships we can derive from the puzzle, the first 81 (or 16) columns of the matrix will be used to represent each number and where last column (either the 82nd or the 17th) will represent what the relationships evaluate to. From there we would reduce the matrix and then print out the relationships.

We would input it in this form and receive the information below it.

1 0 4 0 0 0 0 3 0 0 0 4 0 2 0 0

input the desired size (if 2x2 for each box put 4 if 3x3 input 9

4

4

input the sudoku puzzle

1 0 4 0 0 0 0 3 0 0 0 4 0 2 0 0

1040

0003

0004

0200

```
  r1c1 r1c2 r1c3 r1c4 r2c1 r2c2 r2c3 r2c4 r3c1 r3c2 r3c3 r3c4 r4c1 r4c2 r4c3 r4c4
ans
    0    1    0    0    0    0    0    0    0    0    0    0    0    0    0   -1    2
```

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | -1 | -1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 8 |
| 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | -1 | -6 |
| 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | -3 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | -1 | -1 | -6 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

r1c2-r4c4= 2

r2c1-r3c2= 1

r3c1+r3c2-r4c3-r4c4= 0

r4c1+r4c3+r4c4= 8

-r2c2-r3c2-r4c4=-6

-r1c4-r4c4=-3

r2c3-r4c4= 0

-r3c3-r4c3-r4c4=-6

From here we decided to solve it on our own. For r1c2 it can’t be 1 or 4 and it’s greater than 2 so that means it’s 3 and r4c4 is 1.

| r1c1 | r1c2 | r1c3 | r1c4 | r2c1 | r2c2 | r2c3 | r2c4 | r3c1 | r3c2 | r3c3 | r3c4 | r4c1 | r4c2 | r4c3 | r4c4 | ans |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | -1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 7 |
| 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | -5 |
| 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -2 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | -1 | 0 | -5 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Now for row 4 column 3 it’s one of the free variables so if we look at it we can see that it and r4c1 add up to 7 which means that one of them is 3 and the other is 4. There’s already a 4 in c3 which means that r4c3 is 3 while r4c1 is 4. Knowing that r4c3 is 3 means that r3c3 is 2 and that r4c1 is 4

| r1c1 | r1c2 | r1c3 | r1c4 | r2c1 | r2c2 | r2c3 | r2c4 | r3c1 | r3c2 | r3c3 | r3c4 | r4c1 | r4c2 | r4c3 | r4c4 | ans |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 4 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 4 |
| 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | -5 |
| 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -2 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | -2 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Now we have 1 free variable (r3c2) and 3 items reliant on it among those that haven’t been solved. R2c1-r3c2 is 1 but that won’t help us. R3ce1+r3c2 equals 4 which means neither of them are 2 so one of them is 3 and the other is 1. R3c1 can’t be 1 because there’s a 1 in c1 so that means it’s 3 and that r3c2 is 1. Knowing that r3c2 is 1 it also means that r3c1 is 3, r2c1 is 2 and that r2c2 is 4 . All of this is correct.

| r1c1 | r1c2 | r1c3 | r1c4 | r2c1 | r2c2 | r2c3 | r2c4 | r3c1 | r3c2 | r3c3 | r3c4 | r4c1 | r4c2 | r4c3 | r4c4 | ans |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 4 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -4 |
| 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -2 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | -2 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

And thus, it’s actually solved and we were able to verify this with the actual solution to the puzzle and everything was accurate.

## Findings:

The most important finding that we were able to find is that in addition to the relationships that are always given you can also derive valid relationships between a variety of other parts of the puzzle and that you can use these to assist in solving the sudoku puzzle. If someone were to attempt to use our project to solve a sudoku puzzle it could potentially reduce the problem from finding 70 correct numbers to finding 10 correct numbers which could be much easier. Thus, we found that linear algebra has definite potential applications in solving it and that you can use it to assist you in solving the puzzle. Though we did program it all of the information we gained could also be calculated by hand.

An unexpected finding that we were able to find was actually with something that we didn’t think of originally is how linear algebra can be used in forming a sudoku puzzle. With many sudoku puzzles they will give you the absolute minimum amount necessary to solve it but that brings up the question of what would be a valid puzzle if one of the numbers was gone. An

unexpected application of our program is that it could also be used in determining all possible valid puzzles from that position because, as long as all equations are upheld, it should still be valid.

## Conclusion

In the end, we were unable to solve a complete 9x9 matrix using only linear algebra, however, we were able to gain valuable information about the nature of a sudoku puzzle and we were able to solve a sudoku puzzle by using these relationships and the typical relationships from the puzzle itself. There are certainly certain aspects that we didn't play with, such as which entries we decide are given, what needs to be found, and looking further into what makes an independent constraint. We were also unable to produce an actual solver that would solve it using only the relationships derived from the equations and the constraints of the original puzzle.

Regardless, we certainly made progress on the idea of using linear algebra concepts to assist in solving a sudoku puzzle. Using this information we were able to solve a 4x4 puzzle and, through other experimenting with various puzzles, we were able to find that the relationships derived were accurate. We found that while using this concept, may not give direct answers of puzzle entries, it can reveal patterns and relationships between them which can be useful as one goes along attempting to solve various sudoku puzzles. Overall, though linear algebra cannot be used to solve all sudoku puzzles it can be a very useful tool in solving them and it could be implemented to drastically reduce the number of possibilities and help with solving a Sudoku puzzle.