Using the Linear Algebra App

August 12, 2019[1]

Contents

[1] This document was prepared using the LaTeX module in NCLab

1 About This Document

This document describes how to work with the Linear Algebra App in NCLab. This app is located in the Math section of the Creative Suite, and it can be used to:

- Define your own matrices and augmented matrices.
- Practice elementary row operations.
- Create the echelon and reduced echelon forms of matrices.
- Solve systems of linear algebraic equations using Gauss elimination.
- Invert matrices using Gauss elimination.
- Add and subtract matrices, calculate matrix-vector and matrix-matrix products.
- Use a vast amount of linear algebra functionality provided by Numpy.
- Write Python code - so you can do anything.

2 Defining Matrices

The app makes it possible to define a matrix, an augmented matrix with a single right-hand side, and an augmented matrix with multiple right-hand sides.

2.1 Single Matrix

The app launches with a default demo script:

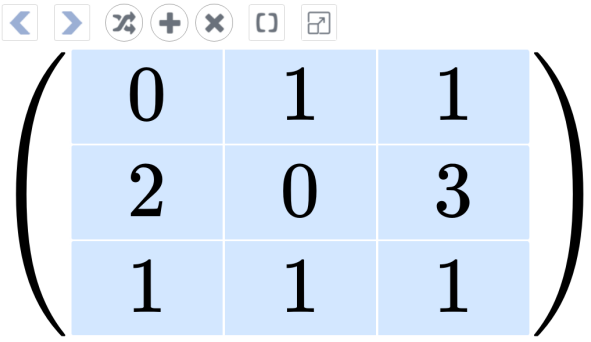
```
M = NCLabMatrixSolver([[0, 1, 1], [2, 0, 3], [1, 1, 1]], [1, 11, 2])
M.eliminator()
```

Here, the first three triplets `[0, 1, 1]`, `[2, 0, 3]`, `[1, 1, 1]` define a 3x3 matrix (each of them represents one row) and the last triplet `[1, 11, 2]` defines a right-hand side for the augmented matrix.

Because we are only interested in defining a matrix, remove the last triplet including the enclosing square brackets, and the comma which precedes it:

```
M = NCLabMatrixSolver([[0, 1, 1], [2, 0, 3], [1, 1, 1]])
M.eliminator()
```

Now, press the Play button and you should see the matrix:



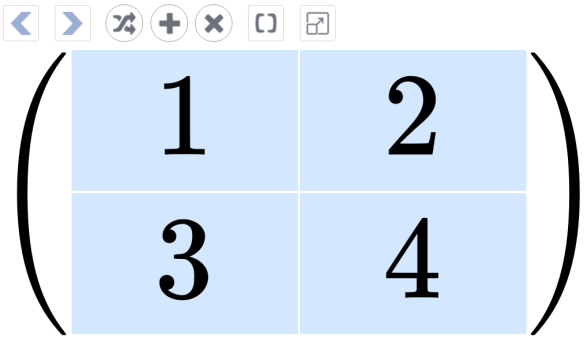
$$\begin{pmatrix} 0 & 1 & 1 \\ 2 & 0 & 3 \\ 1 & 1 & 1 \end{pmatrix}$$

Notice that the rows exactly correspond to the triplets above. Now you can modify the demo script to define your own matrix. Removing or adding triplets will remove or add rows, and removing or adding values to the rows will make the rows shorter or longer.

The following sample code,

```
M = NCLabMatrixSolver([[1, 2], [3, 4]])
M.eliminator()
```

defines a 2×2 matrix:

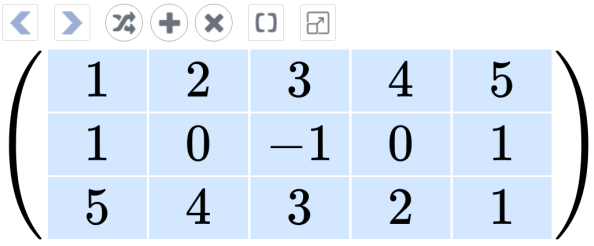


$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

And the sample code

```
M = NCLabMatrixSolver([[1, 2, 3, 4, 5], [1, 0, -1, 0, 1], [5, 4, 3, 2, 1]])
M.eliminator()
```

defines a 3×5 matrix:



$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 0 & -1 & 0 & 1 \\ 5 & 4 & 3 & 2 & 1 \end{pmatrix}$$

2.2 Augmented Matrix with One Right-Hand Side

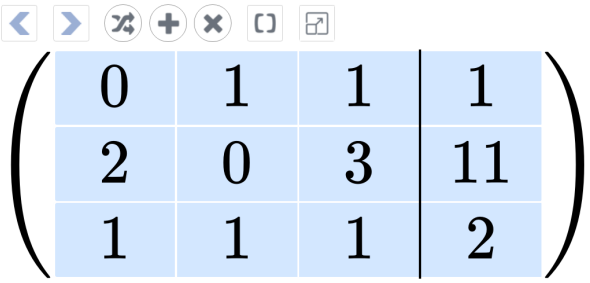
The most common application of an augmented matrix with a single right-hand side vector is to solve a system of linear algebraic equations.

The app launches with a default demo script:

```
M = NCLabMatrixSolver([[0, 1, 1], [2, 0, 3], [1, 1, 1]], [1, 11, 2])
M.eliminator()
```

Here, the first three triplets `[0, 1, 1]`, `[2, 0, 3]`, `[1, 1, 1]` define a 3x3 matrix (each of them represents one row) and the last triplet `[1, 11, 2]` defines a right-hand side for the augmented matrix.

Now, press the Play button and you should see the augmented matrix:



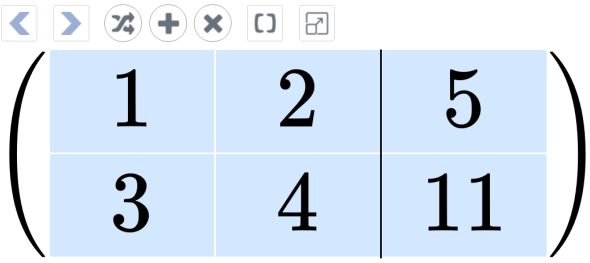
$$\left(\begin{array}{ccc|c} 0 & 1 & 1 & 1 \\ 2 & 0 & 3 & 11 \\ 1 & 1 & 1 & 2 \end{array}\right)$$

Notice that the rows and the right-hand side exactly correspond to the triplets above. Now you can modify the demo script to define your own matrix and right-hand side. Removing or adding triplets will remove or add rows, and removing or adding values to the rows will make the rows shorter or longer. Make sure that the length of the right-hand side vector matches the number of rows.

As another example, the following code,

```
M = NCLabMatrixSolver([[1, 2], [3, 4]], [5, 11])
M.eliminator()
```

defines a 2×2 matrix with the right-hand side $(5, 11)^T$:

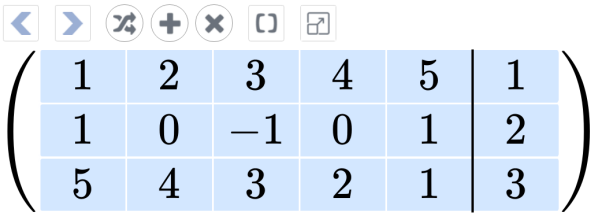


$$\left(\begin{array}{cc|c} 1 & 2 & 5 \\ 3 & 4 & 11 \end{array}\right)$$

And, as a last example, the code

```
M = NCLabMatrixSolver([[1, 2, 3, 4, 5], [1, 0, -1, 0, 1], [5, 4, 3, 2, 1]],
[1, 2, 3])
M.eliminator()
```

defines a 3×5 matrix with the right-hand side $(1, 2, 3)^T$:



$$\left(\begin{array}{ccccc|c} 1 & 2 & 3 & 4 & 5 & 1 \\ 1 & 0 & -1 & 0 & 1 & 2 \\ 5 & 4 & 3 & 2 & 1 & 3 \end{array}\right)$$

2.3 Augmented Matrix with Multiple Right-Hand Sides

Typically, this functionality is used to invert a matrix. The app launches with a default demo script:

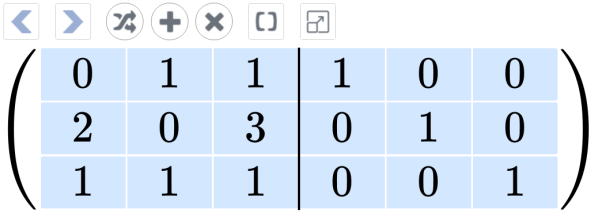
```
M = NCLabMatrixSolver([[0, 1, 1], [2, 0, 3], [1, 1, 1]], [1, 11, 2])
M.eliminator()
```

Here, the first three triplets `[0, 1, 1]`, `[2, 0, 3]`, `[1, 1, 1]` define a 3x3 matrix (each of them represents one row) and the last triplet `[1, 11, 2]` defines a right-hand side for the augmented matrix.

If you need to invert a matrix, replace the last triplet with an identity matrix `[[1, 0, 0], [0, 1, 0], [0, 0, 1]]`. Hence your code will look like this:

```
M = NCLabMatrixSolver([[0, 1, 1], [2, 0, 3], [1, 1, 1]], [[1, 0, 0], [0, 1, 0],
[0, 0, 1]])
M.eliminator()
```

Now, press the Play button and you should see the augmented matrix:

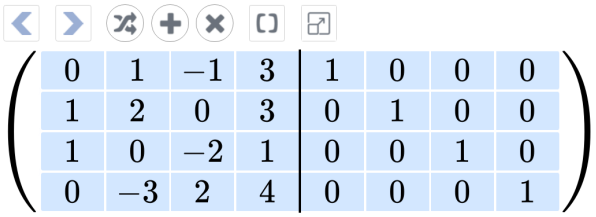


$$\left(\begin{array}{ccc|ccc} 0 & 1 & 1 & 1 & 0 & 0 \\ 2 & 0 & 3 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right)$$

Here is a 4×4 example:

```
M = NCLabMatrixSolver([[0, 1, -1, 3], [1, 2, 0, 3], [1, 0, -2, 1],
[0, -3, 2, 4]], [[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]])
M.eliminator()
```

Corresponding augmented matrix:

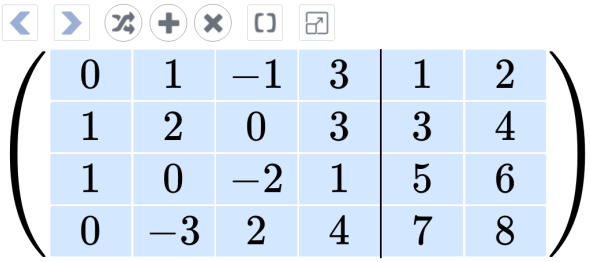


$$\left(\begin{array}{cccc|cccc} 0 & 1 & -1 & 3 & 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 3 & 0 & 1 & 0 & 0 \\ 1 & 0 & -2 & 1 & 0 & 0 & 1 & 0 \\ 0 & -3 & 2 & 4 & 0 & 0 & 0 & 1 \end{array}\right)$$

If fewer or more right-hand sides are needed, that's possible too:

```
M = NCLabMatrixSolver([[0, 1, -1, 3], [1, 2, 0, 3], [1, 0, -2, 1],
[0, -3, 2, 4]], [[1, 2], [3, 4], [5, 6], [7, 8]])
M.eliminator()
```

Corresponding augmented matrix:



$$\left(\begin{array}{cccc|cc} 0 & 1 & -1 & 3 & 1 & 2 \\ 1 & 2 & 0 & 3 & 3 & 4 \\ 1 & 0 & -2 & 1 & 5 & 6 \\ 0 & -3 & 2 & 4 & 7 & 8 \end{array}\right)$$

Just make sure that all the right-hand side vectors have the same length, and that their length is equal to the number of rows in the matrix.

3 Performing Elementary Row Operations

The app supports the three usual elementary row operations: swapping (sometimes called switching or interchange), scaling, and replacement.

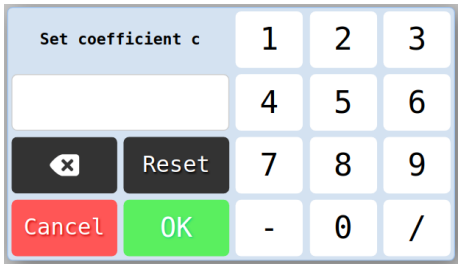
3.1 Swapping

The swapping operation swaps two rows. To swap two rows, left-click (or tap) on one of them, and drag the row to the left. This is how the app will know that you want to do swapping. Then drag the row up or down on the target row, and release. Here is a short video which demonstrates the process:

https://youtu.be/ZQu152f0hrY

3.2 Scaling

The scaling operation multiplies a row with a nonzero number. To scale a row, just click or tap on it, and release. A keypad will appear:

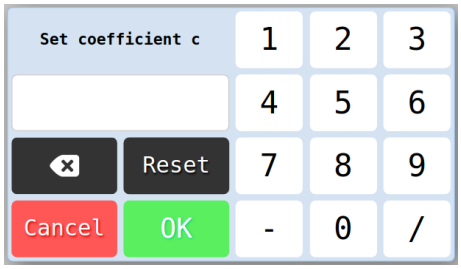


There, specify the nonzero constant to multiply the row, and click or tap OK. The following video demonstrates the process:

https://youtu.be/7vG_P1oxdz0

3.3 Replacement

The replacement operation multiplies a row with a nonzero coefficient, and adds the result to another row. To perform a replacement operation, click or tap on a row, drag it directly up or down (not sideways) and release upon the target row. Then a keypad appears:



There, specify the nonzero constant to multiply the row which was moved, and click or tap OK. The following video demonstrates the process:

https://youtu.be/201uixuJDgg

4 Controls

When the matrix or augmented matrix is defined successfully, it will render along with seven control buttons above it:

From left to right, the buttons have the following functions:

1. Undo
2. Redo
3. Indicator of swapping operation
4. Indicator of replacement operation
5. Indicator of scaling operation
6. Use square brackets instead of parentheses and vice versa.
7. Full screen mode. In full screen mode, the same button can be used to come back.

5 Getting an Error?

If you make a mistake defining the matrix or the augmented matrix, you will receive an error message generated by the Python engine on the server. For example, forgetting a closing square bracket,

```
M = NCLabMatrixSolver([[1, 2, 3, 4, 5], [1, 0, -1, 0, 1], [5, 4, 3, 2, 1])
M.eliminator()
```

will cause an error:

```
on line 1:
    M = NCLabMatrixSolver([[1, 2, 3, 4, 5], [1, 0, -1, 0, 1], [5, 4, 3, 2, 1])
                          ^
SyntaxError: The bracket [ is closed by bracket ) on line 1.
```

Usually, the error messages are fairly self-explanatory, and they should help you. If you get stuck, send us an email to `support@nclab.com` and we will try to help.

Last but not least, there always is a possibility that the app itself has a bug. If you come across one, **please report it to us!** We are genuinely interested in improving the app.

6 Miscellaneous

The code input cell allows you to enter Python code and, in particular, to import Numpy. Then, you have a vast amount of linear algebra functionality at your disposal. Check the documentation and examples to the Numpy Linalg module at

`https://docs.scipy.org/doc/numpy/reference/routines.linalg.html`

We will show you just three simple examples - matrix addition, matrix-vector multiplication, and matrix-matrix multiplication.

6.1 Matrix Addition

The following example illustrates how to add matrices:

```
import numpy as np
A = np.array([[5, 1, 3], [1, 1, 1], [1, 2, 1]])
B = np.array([[-1, 0, 2], [2, 3, -4], [0, 2, -2]])
A+B
```

Output:

```
array([[ 4,  1,  5],
       [ 3,  4, -3],
       [ 1,  4, -1]])
```

6.2 Matrix-Vector Multiplication

Matrix-vector multiplication can be done as follows:

```
import numpy as np
A = np.array([[5, 1, 3], [1, 1, 1], [1, 2, 1]])
b = np.array([1, 2, 3])
A.dot(b)
```

Output:

```
array([16,  6,  8])
```

6.3 Matrix-Matrix Multiplication

And this example shows how to do matrix-matrix multiplication:

```
import numpy as np
A = np.array([[5, 1, 3], [1, 1, 1], [1, 2, 1]])
B = np.array([[-1, 0, 2], [2, 3, -4], [0, 2, -2]])
A.dot(B)
```

Output:

```
array([[-3,  9,  0],
       [ 1,  5, -4],
       [ 3,  8, -8]])
```