User Guide

Andrew Ye and James Ross

April 2024

Contents

1 Introduction

Our overall goal for this project was to create a program that would perform matrix operations on a given matrix. We wanted to create an algorithm that would put a matrix into Row Reduced Echelon Form (RREF), transform a matrix into its inverse, and multiply and add matrices together. Our programs takes the input of a matrix (in the form of a list of lists of numbers) and outputs the matrix resulting from the specified operation. These operations are some of those studied in linear algebra classrooms, and our program can help as a multi-use calculator for the class.

2 Accessing the Project

To utilize our program, first navigate to the RREF repository and download the files *all_together.py* and *execute.py*. Create an instance of the matrix class by assigning a matrix to *matrix*(your matrix goes here) inside the *execute.py* file. Now you can apply methods to this instance.

```
from all_together import matrix
x = matrix([[-3,3,4,5], [2,2,3,3],[2,2,3,22]])
print(x.matrix)
```

In this example, x is an instance of the matrix class with the 2-d list representing a matrix. But since x is an instance of the matrix class, printing out x will have no meaning since the matrix itself is defined as one of its instance variables. To print the instance variable add $.matrix$.

3 Class Methods

To meet our goal, we created a matrix class to house our matrix operation methods.

Operation	Function Name
Row Reduced Echelon Form	.rref()
Inverse Matrix	.inverse()
Represent each entry as a fraction	.fraction()
Multiply Matrics	$*$
Add Matrics	$+$

Note that each method returns an instance of the matrix class, so to access the matrix itself you will have to add $.matrix$ to the end of the instance. Also, the $.fraction()$ method returns a list of list of strings, so $.fraction()$ should be the last method applied to whatever operation you are applying to the matrix.

4 .rref()

This row reduces the matrix to echelon form. This method is a bijection between $\mathbb{R}^{m\times n}$ and $\mathbb{R}^{m\times n}$

```
from all_together import matrix
x = matrix([[-3,3,4,5], [2,2,3,3],[2,2,3,22]])
print(x.rref().matrix)
```

This code row reduces this matrix

$$\begin{bmatrix} -3 & 3 & 4 & 5 \\ 2 & 2 & 3 & 3 \\ 2 & 2 & 3 & 22 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0.08333333333333326 & 0 \\ 0 & 1 & 1.4166666666666665 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

5 .inverse()

This method returns the matrix B such that $rref(A) = BA$ where A is the input matrix. Thus when A is square and invertible, it will return the inverse of A where $I = rref(A) = BA$. This method is a bijection between $\mathbb{R}^{m\times n}$ and $\mathbb{R}^{m\times m}$

```
from all_together import matrix
x = matrix([[-3,3,4,5], [2,2,3,3],[2,2,3,22]])
print(x.inverse().matrix)
```

This code finds the matrix

$P = \begin{bmatrix} -0.166666666 & 0.24561403 & 0.00438596491 \\ 0.16666666666 & 0.333333333 & -0.08333333 \\ 0 & -0.0526315789 & 0.0526315789 \end{bmatrix}$ for the matrix $A = \begin{bmatrix} -3 & 3 & 4 & 5 \\ 2 & 2 & 3 & 3 \\ 2 & 2 & 3 & 22 \end{bmatrix}$. $PA = rref(A)$. When A is square and invertible, $P = A^{-1}$

6 .fraction()

This method returns the matrix as another matrix except each entry is a string in fraction form.

```
from all_together import matrix
x = matrix([[-3,3,4,5], [2,2,3,3],[2,2,3,22]])
print(x.inverse().fraction().matrix)
```

Remember to use this method last as converting the matrix items into a string makes them inoperable.

7 +

Matrix addition between two instances of matrix class. $f : \mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n} \rightarrow \mathbb{R}^{m\times n}$.

```
from all_together import matrix
x = matrix([[-3,3,4,5], [2,2,3,3],[2,2,3,22]])
y = matrix([[-3,3,4,10], [2,22,3,3],[2,11,3,22]])
print((x + y).matrix)
```

This adds together two matrices.

$$\begin{bmatrix} -3 & 3 & 4 & 5 \\ 2 & 2 & 3 & 3 \\ 2 & 2 & 3 & 22 \end{bmatrix} + \begin{bmatrix} -3 & 3 & 4 & 10 \\ 2 & 22 & 3 & 3 \\ 2 & 11 & 3 & 22 \end{bmatrix} = \begin{bmatrix} -6 & 6 & 8 & 15 \\ 4 & 24 & 6 & 6 \\ 4 & 13 & 6 & 44 \end{bmatrix} \tag{2}$$

8 *

Matrix multiplication between two instances of matrix class. $f : \mathbb{R}^{m\times n} \times \mathbb{R}^{n\times p} \to \mathbb{R}^{m\times p}$.

```
from all_together import matrix
x = matrix([[-3,3,4,5], [2,2,3,3],[2,2,3,22]])
y = matrix([[-3,3,4,10,9],
    [2,22,3,3,8],[2,11,3,22,7],[4,2,1,34,5]])
print((x * y).matrix)
```

This multiplies two matrices.

$$\begin{bmatrix} -3 & 3 & 4 & 5 \\ 2 & 2 & 3 & 3 \\ 2 & 2 & 3 & 22 \end{bmatrix} \begin{bmatrix} -3 & 3 & 4 & 10 & 9 \\ 2 & 22 & 3 & 3 & 8 \\ 2 & 11 & 3 & 22 & 7 \\ 4 & 2 & 1 & 34 & 5 \end{bmatrix} = \begin{bmatrix} 43 & 111 & 14 & 237 & 50 \\ 16 & 89 & 26 & 194 & 70 \\ 92 & 127 & 45 & 840 & 165 \end{bmatrix} \tag{3}$$