Row Reduction and Inverse Calculator

Andrew Ye and James Ross

May 2024

Table of Contents

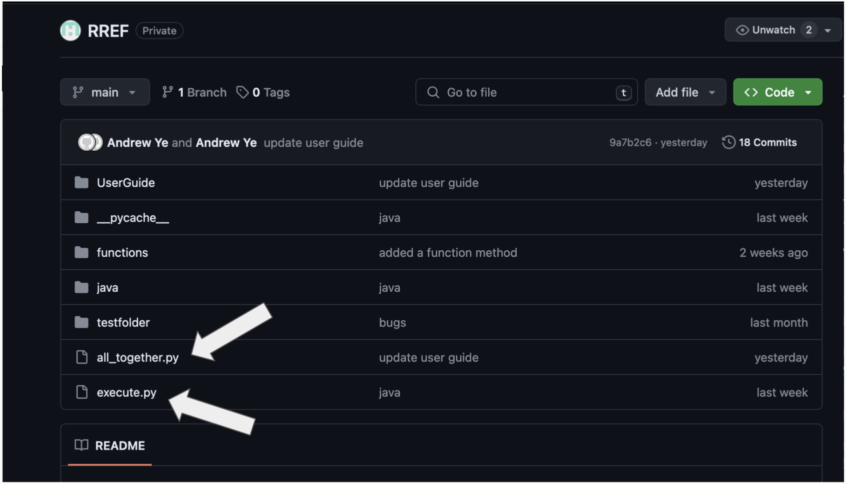
Objective

Our objective for this project was to create a program that would allow the user to perform matrix operations found in a linear algebra class.

Accessing the Calculator

Github

Go to this link `https://github.com/AndrewYe12/RREF`.
Download the *alltogether.py* and *execute.py* files.



Using the Calculator

Creating Your Matrix

To encode your matrix in a way the program can act upon you must format it as a list of lists of integers, with each list of numbers representing a row.

```
x = matrix([[1,2],[2,1]])
```

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$$

Matrix Operations as Methods

Our Methods

We wanted to program an algorithm to row reduce a matrix, find the inverse of a matrix, add/multiply two matrices, and find the determinant of a matrix.

Intro to Matrix Operation Methods

$.rref()$: Reduces the matrix to reduced echelon form

$.inverse()$: Finds and returns the inverse of the matrix

$+$: Adds two matrices together

$*$: Multiplies two matrices together

$.fraction()$: Returns the matrix as a string of fractions

$.det()$: Calculates the determinant of a matrix.

Row Reduction

Transforms a matrix into row reduced echelon form.

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

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

Inverse

Outputs the inverse of a matrix.

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

$A = \begin{bmatrix} -3 & 3 & 4 & 5 \\ 2 & 2 & 3 & 3 \\ 2 & 2 & 3 & 22 \end{bmatrix}$. This code finds the matrix $P = \begin{bmatrix} -0.166666666 & 0.24561403 & 0.00438596491 \\ 0.16666666666 & 0.333333333 & -0.08333333 \\ 0 & -0.0526315789 & 0.0526315789 \end{bmatrix}$ so that

$$PA = rref(A)$$

Addition

Inputs two matrices and returns one with the added terms.

```
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)
```

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

Multiplication

Inputs two matrices and outputs the matrix resulting from performing matrix multiplication.

```
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)
```

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

Fraction

Inputs a matrix and returns a copy of it with the entries as fractions.

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

$$\begin{bmatrix} -3 & 3 & 4 & 5 \\ 2 & 2 & 3 & 3 \\ 2 & 2 & 3 & 22 \end{bmatrix} \to \begin{bmatrix} `\frac{-3}{1}' & `\frac{3}{1}' & `\frac{4}{1}' & `\frac{5}{1}' \\ `\frac{2}{1}' & `\frac{2}{1}' & `\frac{3}{1}' & `\frac{3}{1}' \\ `\frac{2}{1}' & `\frac{2}{1}' & `\frac{3}{1}' & `\frac{22}{1}' \end{bmatrix}$$

Determinant

Inputs a matrix and returns the determinant.

```
from all_together import matrix
x = [[1,0,0],[0,1,0],[0,0,1]]
print(x.det())
```

$$det \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = 1$$

Reflection

What Went Well

The matrix multiplication and addition methods were relatively easy to implement because they had an explicit formula. For example, if A is a $m \times n$ matrix and B is a $n \times p$ matrix, then

$$(AB)_{ij} = \sum_{r=1}^{n} A_{ir} * B_{rj} \tag{1}$$

Problems We Faced

The inverse and row reduction methods were difficult to implement. The hardest part was swapping the rows. The rows only have a position relative to each other.

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Results

This program has already helped us in our linear algebra class by allowing us to compute these steps in an efficient way.

Results

This program has already helped us in our linear algebra class by allowing us to compute these steps in an efficient way.

Are there any questions?