Project 1, Transformations

Tejas Patel

25 March, 2025

1

Matrices defined: $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \ B = \begin{bmatrix} 0 & 2 & -1 \\ 1 & 1 & 10 \end{bmatrix}, \ V = \begin{bmatrix} 1 & 4 & 0 \end{bmatrix}$

When `show(A), show(B), show(V)` was input into the python terminal, the output from SageMath was $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \ \begin{bmatrix} 0 & 2 & -1 \\ 1 & 1 & 10 \end{bmatrix}, \ \begin{bmatrix} 1 & 4 & 0 \end{bmatrix}$ respectively. The matrices were repeated back to me as I originially entered them

$$\text{For } QQ \to RR \Rightarrow \begin{bmatrix} 1.00000000000000 & 3.00000000000000 \\ 2.00000000000000 & 4.00000000000000 \end{bmatrix}$$

$$\text{For } QQ \to RDF \Rightarrow \begin{bmatrix} 1.0 & 2.0 \\ 3.0 & 4.0 \end{bmatrix}$$

RR and RDF seem to be data types for the matrices. RR appears to be an aribtrary precision floating point, outputting the maximum number of decimal places allowed. It won't be as precise as QQ, which is symbolic math, nor will it be as fast as RDF, but it will get you a decimal answer when youre looking for one.

$$\text{RREF}(B) = \begin{bmatrix} 1 & 0 & \frac{21}{2} \\ 0 & 1 & -\frac{1}{2} \end{bmatrix}$$

The command creates an identity matrix of the degree of the input.

For the original command: $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ For input = 2: $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ For input = 5: $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

2

New matrices defined: $C = \begin{bmatrix} 3 & 7 \\ 1 & 6 \end{bmatrix}$

In SageMath, to add: `A+C` results in $\begin{bmatrix} 4 & 9 \\ 4 & 10 \end{bmatrix}$, to subtract `A-C` results in $\begin{bmatrix} -2 & -5 \\ 2 & -2 \end{bmatrix}$

Lastly, to multiply: `A*C` results in $\begin{bmatrix} 5 & 19 \\ 13 & 45 \end{bmatrix}$, bonus: `det(A*C)` results in -22

$$B^T = \begin{bmatrix} 0 & 1 \\ 2 & 1 \\ -1 & 10 \end{bmatrix}$$

`A*C==C*A` returned `false`, meaning $AC \neq CA$

`(A*C).transpose()==A.transpose()*C.transpose()` returned `false`, meaning $(AC)^T \neq A^T C^T$

`(A+C).transpose()==A.transpose()+C.transpose()` returned `true`, meaning $(A+C)^T = A^T + C^T$

`(c*A).transpose()==c*A.transpose()` returned `true` for $c = 6$ and $c = 10$, meaning $(cA)^T = c * A^T$

3

4

5