Matrix Calculator

March 24, 2025 2:56 PM

Vector Class:

```
vector<double> entries;
sizet n; // the number of rows in the vector

public:
  Vector(sizet n); // Constructor

// also copy and move constructors
// as copy and move operators

  void print(); // prints the vector

  + operator(other vector);
  * operator(double c)//scalar multiplication

  dot product
};
```

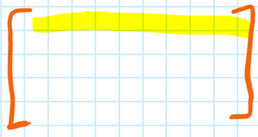
A×B	A^T	rref
A+B	print	inverse
cA	det	eigen

Matrix Class:

```
vector<Vector> entries
sizet n; // # of rows
sizet m; // # of cols

public:
  //regular constructors and assignment operators
  Matrix operator+(const Matrix & other)
  Matrix& operator+=(const Matrix & other)

  Matrix operator*(const Matrix & other)
  Matrix& operator*=(const Matrix & other)

  friend Matrix operator*(const double c, const Matrix & m);
  friend Matrix& operator*=(const double c, Matrix & m);

  void transpose();
  double det();
  void print();
  void rref();
  Matrix inverse();

  Void eigen();

  int getRows();
  int getCols();
};
```

I'm going to make the matrix implementation such that each Vector in the array of Vectors is a row vector instead of a column vector (It makes my life easier)

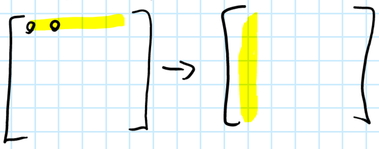
i.e.
m[0] refers to



m[0] does not refer to

Matrix Multiplication Algorithm

Transpose Algorithm



Note:

The Exception Handling is pretty messy, when you get the chance, clean it up.