APPM 2460
VECTORS & MATRICES II

1. Introduction

This week we're going to spend more time working on our ability to slice matrices (to "slice" a matrix or array means to grab certain portions of the matrix via indexing). In particular, we'll focus on performing row exchanges. This will allows us to perform the same type of Gaussian elimination we do on APPM 2360 homework.

2. Permuting Rows of a Matrix

We're going to first work on the example of permuting the rows of a matrix. "Permute" is a fancy word for "rearrange." We'll first work on the special case of swapping two rows. Eventually, we'll learn how to perform arbitrary permutations (i.e. make a whole bunch of row swaps at once.)

We'll need some matrices to play with. Let's use the command `magic(4)` to build a 4×4 matrix:

```
>> A = magic(4)

A =

    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

To figure out what a "magic" matrix is, we can use `help magic`:

```
>> help magic
 magic  Magic square.
    magic(N) is an N-by-N matrix constructed from the integers
    1 through N^2 with equal row, column, and diagonal sums.
    Produces valid magic squares for all N > 0 except N = 2.
```

In our case, it's just some matrix we're going to play around with.

Now, suppose we wanted to interchange the first and third rows of `A`. There are a few ways we could do this. An effective but somewhat clunky way is shown in the script below.

```
A = magic(4);
% make a temporary variable that holds the first row
temp_row = A(1,:);
% replace first row with third row
A(1,:) = A(3,:);
% and put variable holding first row back into the third row
A(3,:) = temp_row;
```

After running this code, try displaying the matrix `A`. You will see that we have, in fact, successfully swapped these two rows. However, this method is slow, because we need to store our first row so that it doesn't get lost when we overwrite. Try to determine your own method to do this more efficiently!

3. An example with a linear system

Let's consider the linear system $\begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix} \vec{x} = \begin{pmatrix} 3 \\ 2 \end{pmatrix}$ which is equivalent to the 2×3 augmented matrix

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

Using standard row operations, $R_1 \to R_1 - 2R_2$ we can put the augmented matrix in rref

$$\left(\begin{array}{cc|c} 1 & 0 & -1 \\ 0 & 1 & 2 \end{array}\right),$$

which in turn gives the solution to the linear system. This process is a bit tedious and would become extremely complicated quite quickly. Fortunately, Matalb has a built in function for computing the rref of a matrix. The rref of the matrix is computed in the following way for the above example

```
>> B = [1 2 3; 0 1 2]; %define augmented matrix
>> rref(B)

ans =

     1     0    -1
     0     1     2
```

4. Solving Linear Systems via Inverses

We have explored various ways to determine the way to compute the solution to a linear system of equations in APPM 2360. One way that we can use is to compute the inverse of the coefficient matrix, A, and left-multiply both sides of the equations by A^{-1} (supposing that A^{-1} exists). Let's consider the same previous linear system from before

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

The solution to the linear system via matrix inversion is

$$\begin{aligned} \vec{x} &= \begin{pmatrix} 1 & -2 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 3 \\ 2 \end{pmatrix} \\ &= \begin{pmatrix} -1 \\ 2 \end{pmatrix} \end{aligned}$$

Let's see how we can do this in Matlab

```
>> A = [1 2; 0 1]; %coefficient matrix
>> b = [3;2]; %RHS
>> inv(A)*b

ans =

    -1
     2
```

5. Homework

For the homework this week we will practice matrix operations on the following linear system

$$\begin{aligned} 2x + \ \ y + 3z &= 10 \\ x + \ \ y + \ \ z &= \ \ 6 \\ x + 3y + 2z &= 13 \end{aligned}$$

- Write the system in the form $A\vec{x} = b$. Compute the rref of the matrix by hand. Include the result in your submission.
- Verify the rref with Matlab.
- Compute A^{-1}.
- Use A^{-1} to verify the solution of the linear system.
- Compute the determinant of the coefficient matrix A. Now, swap two rows of A (you can use the example code above or try your own version). What should happen to your determinant.
- Use Matlab to swap the first and last rows of A and compute A^{-1}. How is this different from your original inverse?