Another Way to Solve Systems

- Array commands in Julia
- Gaussian Elimination
- Reduced Row-Echelon Form

Useful Array Commands

First, type this array into Julia: A = $\begin{bmatrix} 2 & -1 & 4 \\ 6 & 0 & -3 \end{bmatrix}$

Then try these commands:

`A[1]`	first element (upper left)
`A[3]`	third element (counting across)
`A[4]`	fourth element
`A[1, 3]`	first row, third column
`A[2, 2]`	second row, second column

Useful Array Commands

Now try these:

`A[1, :]`	first row of A
`A[2, :]`	second row
`A[:, 3]`	third column
`B = A[2, :]`	B is now the second row of A
`vcat(A[1, :], B)`	re-creates A from A row 1 and B
`hcat(A, [3; 5])`	augments A on the right

Useful Array Commands

Finally, try these:

```
A[1,:]=[A[1,:]*A[2,1]/A[1,1]]
```

…what did that do?

Next,

```
A[2,:]= [A[1,:] - A[2,:]]
```

…what did that do?

A Note About Types

Now, try typing this in:

```
A[1, :] = A[1, :]/5
            ERROR: InexactError()...
```

… so why can't you divide by 5?

The answer is that Julia thought A was an integer only array (Int64), not all numbers (Float64). Dividing by 5 would give non-integer results which is considered illegal in A.

A Note About Types

There are multiple ways around this.

One is to be specific about the desired type of an array when you originally enter it:

```
A = Float64[3 3 -2; 4 1 0]
```

Another is to enter one of the original numbers as a decimal:

```
A = [3.0 3 -2; 4 1 0]
```

If the matrix has already been entered, you can convert it:

```
A = float64(A)
```

Practice Problem 1

1. Let A = $\begin{bmatrix} 3 & 1 & -2 \\ 2 & -2 & 5 \end{bmatrix}$. Use Julia to get a 1 in the first row, first column (location [1, 1]) and 0 in the second row, first column (location [2, 1]) by

- dividing the entire first row by 3, then
- replacing the second row with a sum of the second row and a multiple of the first row

Try to make your code as general as possible (ie, use locations rather than actual numbers from the array)

Gaussian Elimination

Let's say you were solving this system of equations using *elimination*:

$$2x_1 + 3x_2 = 4$$
$$3x_1 - 5x_2 = 5$$

You might choose to multiply the first row by 3 and the second row by -2, then add the two rows like this:

$$\begin{array}{r} 6x_1 + 9x_2 = 12 \\ \underline{-6x_1 + 10x_2 = -10} \\ 19x_2 = 2 \end{array}$$

After dividing, you might plug the answer back in to the first equation to find x_1.

Gaussian Elimination

Without actually solving it, the implication is:

The solution to $\begin{cases} 2x_1 + 3x_2 = 4 \\ 3x_1 - 5x_2 = 5 \end{cases}$

is the same as the solution to $\begin{cases} 2x_1 + 3x_2 = 4 \\ \qquad\; 19x_2 = 2 \end{cases}$

but the second is easier to solve.

Gaussian Elimination

In matrix form, we could say that the solution to

$$\begin{bmatrix} 2 & 3 & 4 \\ 3 & -5 & 5 \end{bmatrix}$$

is the same as the solution to

$$\begin{bmatrix} 2 & 3 & 4 \\ 0 & 19 & 2 \end{bmatrix}$$

but the second is easier to solve.

Practice Problem 2

Write a program that, given a 2x3 matrix A,

a) returns a matrix with 0 in the lower corner
b) reports the value of x_2
c) reports the value of x_1.

Test your code!

Reduced Row-Echelon Form

As long as we're building equivalent (but simpler) matrices, this is the ideal form:

$$\begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \end{bmatrix}$$

With reduced row-echelon form, this is exactly the goal. It's painful to do by hand, but with computers, it's not so bad.

Reduced Row-Echelon Form

The rules for creating equivalent matrices are as follows:

1. You may always, anytime, multiply or divide a row by a constant.
2. You may replace any row with the sum or difference of that row and another row.
3. You may combine these operations by combining multiples of rows.

Reduced Row-Echelon Form

In creating the ideal (row-reduced) matrix, the simplest way to progress is as follows:

$$\begin{bmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \end{bmatrix}$$

original problem

$$\begin{bmatrix} 1 & c_{12} & d_1 \\ 0 & c_{22} & d_2 \end{bmatrix}$$

first column complete

$$\begin{bmatrix} 1 & 0 & x_1 \\ 0 & 1 & x_2 \end{bmatrix}$$

second column complete, matrix solved

Practice Problem 3

Write a program that will return a 2x3 matrix A in reduced row-echelon form, and the answers (x_1, x_2) as an array B.

Then modify your program so it only returns the answers.

Reduced Row-Echelon: Moving Up

Next, we'll move on to solving 3x3 systems, like

this one:
$$\left[\begin{array}{ccc|c} 4 & -2 & 1 & 12 \\ 3 & 0 & -1 & 5 \\ -2 & 1 & 3 & -8 \end{array}\right]$$

As we do so, here is some useful vocabulary:

The current row of focus (the row where you divide to get 1 and make the rest of the column 0) is called the pivot row.

The location that becomes = 1 is called the pivot.

The process of getting 0's in the rest of the column is called pivoting.

Reduced Row-Echelon Form

In words, the process is summarized like this:

1. Divide row 1 by the number in [1, 1].
2. Pivot around [1, 1].
3. Divide row 2 by the number in [2, 2].
4. Pivot around [2, 2].
5. (repeat for row 3 and [3, 3])

Reduced Row-Echelon Form

You could summarize even more by saying:

For rows k = 1-3, divide row k by [k, k], then pivot around [k, k].

And, you could deal with even larger matrices by saying:

For rows k = 1-n, divide row k by [k, k], then pivot around [k, k].

Practice Problems 4-6

4. Write a program that solves 3x3 matrices using reduced row-echelon solving without using loops.

5. Modify your program so it uses "for" loops to solve 3x3 matrices.

6. Write a program that solves a matrix of any size (use `size(A,1)`) to find the number of rows) using reduced row-echelon solving.

Test your code! Then document and save this program!