Solving a System of n Linear Equations via Gaussian Elimination

Gaussian Elimination is typically taught in Algebra II in high school. When transferring what's taught in high school to working code, the key is to understand issues dealing with floating point error. The key is to be very careful to remove floating point error. One of the ways in which floating point error is magnified in computer programs is via division by small absolute values. Imagine a cascading sequence of dividing various values by small values, all of which are off by a little bit. The end result could be off by a great deal.

Review of Gaussian Elimination, as Taught in High School

In school, we rarely solve more than 3 linear equations in three variables. Let's go through an example by hand:

3x + 2y + z = 17
x - y + 3z = 0
2x + 3y + 2z = 21

We take the first equation, and use it to eliminate x in the next two equations. Let's call this equation, the one we use to eliminate a variable, the "pivot equation". We do this as follows:

The coefficient in front of x in the second equation is 1 and the coefficient in front of x in the pivot equation is 3. Divide these terms to get 1/3 and multiply the first equation through by this factor to get:

$$x + \frac{2}{3}y + \frac{z}{3} = \frac{17}{3}$$

Now, subtract this from the second equation to get:

$$-\frac{5}{3}y + \frac{8}{3}z = -\frac{17}{3}$$

We can repeat this process for the third equation to transform it to:

$$\frac{5}{3}y + \frac{4}{3}z = \frac{29}{3}$$

Now, we have eliminated x from all equations except the first. Now, we can pretend the first equation doesn't exist and treat the rest of the equations as two equations and two variables and repeat the process. When we do this, the third equation transforms to:

$$4z = 4$$

The idea is that no matter how many equations we have, we can utilize this process to have 1 equation in n variables, 1 in n-1 variables, etc. and 1 in 1 variable.

Then, using that last equation, we can solve for z. In this case z = 1.

Now, let's go to the previous equation: $-\frac{5}{3}y+\frac{8}{3}z=-\frac{17}{3}$. Since we know z = 1, we can plug this into the equation to get:

$$-\frac{5}{3}y+\frac{8}{3}(1)=-\frac{17}{3}$$
$$-\frac{5}{3}y=-\frac{17}{3}-\frac{8}{3}$$
$$-\frac{5}{3}y=-\frac{25}{3}$$
$$y=5$$

Finally, we can now look at the original first equation and using the known values of y and z, solve for x:

$$3x+2y+z=17$$
$$3x+2(5)+1=17$$
$$3x=6$$
$$x=2$$

Gaussian Elimination: Issues in Code

Normally, we just use the first equation as our first pivot equation, our second equation as our second pivot equation and so forth. But what if, when eliminating a variable, another coefficient we didn't think would change to 0 does? In this case, if we are trying to eliminate variable y, using the equation:

$$0x+0y+3z=6$$

would not be terribly helpful.

In code, if we tried to use this as a pivot equation, in the step where we figure out the factor to multiply our pivot equation by, we would divide by 0, causing our program to crash. ***In fact, even if this value was non-zero, but close to zero, this division would cause magnified floating point error that would cascade in future operations!!!***

So, in code, what we'd like to do is choose our pivot equation very, very carefully. Most importantly, we want our pivot equation to have a large absolute value for the coefficient of the critical term. For example, if we had an equation with 4 variables, x, y, z and w, and after eliminating x, we had these three equations (not counting the initial pivot equation):

$$.1y+5z-10w=17.3$$
$$7y-z+3w=13.2$$
$$2y+3.6z-6.1w=22.2$$

Instead of using the top equation to simplify the rest, we should first swap the first two equations and use the one with a coefficient of 7 in front of y:

$$
\begin{gathered}
7y - z + 3w = 13.2 \\
.1y + 5z - 10w = 17.3 \\
2y + 3.6z - 6.1w = 22.2
\end{gathered}
$$

The other thing that could happen is that all equations with y could disappear. When this occurs, this means there are either no solutions to the system or an infinite number of solutions. A simple way of thinking about what to do at this step is as follows (though we won't delve further into this situation):

1) You will have k+1 equations with only k variables since 2 variables dropped out in one step.
2) Just try to solve the system with the first k of these equations. If all goes well, there will be a unique solution.
3) Then, see if that unique solution works perfectly in the last equation. If so, then use this to plug into the previous equation. When you do this, what will remain is an equation with two variables and one constant, something like 2x + 3y = 7. Since this is the only equation with these two variables, any setting of values for x and y that satisfy this equation would be valid.

Unfortunately, this isn't a complete representation of all cases. In an elimination step, more than 2 variables could be eliminated. In a system of k+c equations in k variables, it might be the case that the first k equations don't have a unique solution. It could also be the case that after several elimination steps the first issue occurs, so that there are multiple equations with x and y instead of just one. This is just a guide of how to try to deal with these situations. All of them correspond to a case where the system either has no solution or an infinite number of solutions. The rest of these notes will focus on solving systems with a unique solution.

Key Components in Code for Gaussian Elimination

1) A function to figure out at an arbitrary step, which candidate equation has the largest absolute value coefficient for the pivot variable.

2) A function to swap rows, to get the best pivot equation into the proper spot.

3) A function to reduce each relevant equation based on the selected pivot equation.

4) A function to perform back substitution, once the equations are in "Upper Triangular" form.

In the posted code, these functions are performed by the following methods:

getMaxRow, swapRows, reduce and solveIt.