Linear Algebra

Iterative methods

If we are looking for a simpler way to solve linear equations, then we can try for an iteration. Take again our example equations:

$$x+y-z = 4,\ \ 2x-y+3z = -3,\ \ -x-2y+2z = -7.$$

We can 'solve' these equations for x, y and z respectively:

$$\begin{aligned} x &= 4-y+z, \\ y &= 3+2x+3z, \\ z &= -\tfrac{7}{2}+\tfrac{1}{2}x+y, \end{aligned}$$

and we can substitute trial values into the RHS to get new values on the LHS. Sadly it doesn't work; at least, not in this case. For example, if we start from all zeros, then we get, successively, $x = 4$, $y = 11$, $z = 9\frac{1}{2}$, $x = 2\frac{1}{2}$, $y = 36\frac{1}{2}$, $z = \ldots$, and there is no sign of convergence. Even if you use Aitken's device, the values are so random that you will have to be very lucky to get it to work.

Does this idea *ever* work? Yes! It works if the initial matrix is *diagonally dominant*. We'll make this more precise soon, but for the time being, it means that the coefficients of the variables we're iterating towards are large compared with the others. For example, try multiplying some of the coefficients in our example by 10:

$$10x+y-z = 4,\ \ 2x-10y+3z = -3,\ \ -x-2y+20z = -7,$$

so that

$$\begin{aligned} x &= (4-y+z)/10, \\ y &= (3+2x+3z)/10, \\ z &= (-7+x+2y)/20, \end{aligned}$$

and now starting from zero gives successively $x = 0.4$, $y = 0.38$, $z = -0.292$, $x = 0.3328$, $y = 0.27896$, $z = -0.305464$, $x = 0.3415576$, $y = \ldots$, and we see that the process now seems to be working.

Diagonal domination

The requirement for diagonal domination is that each diagonal element [the 'pivoting' coefficients for x, y, ...] must be greater in absolute value than the sum of the absolute values of all the other elements in the same row of the matrix of the LHS. If so, then when we solve, we will get equations of the form $x = p + qy + rz + st + \ldots$, with a guarantee that $|q| + |r| + |s| + \ldots < 1$. So, whatever the errors in y, z, t, ..., and whatever their signs, we have a guarantee that the resulting error in x will be strictly smaller than the largest of the other errors. As that will happen to every new substitution we make, we can be sure that after a complete cycle, the largest error has been made smaller by at least a known ratio. That is, we have a guaranteed first-order convergence.

Don't be tempted, ever, to use iterative techniques without this guarantee. You may expect/hope to be lucky, but you won't be.

If you just write down a random matrix, then you cannot expect it to be diagonally dominant. But *(a)* some numerical techniques generate large matrices that are, of their nature, DD; and *(b)* it may sometimes simply be a matter of re-ordering or otherwise tweaking the equations, especially if a few elements are a lot larger than the others. For example,

$$10x + y - 10z = 4, \quad 20x - y + 30z = -3, \quad -x - 20y + 2z = -7$$

[another tweak from the original example!] is far from DD, but the first two equations are almost equations for x and z, so we can eliminate x [to get $-3y + 50z = -11$] and z [to get $50x + 2y = 9$], and then re-order the equations, to get

$$50x + 2y = 9, \quad -x - 20y + 2z = -7, \quad -3y + 50z = -11,$$

and now all is well. [We didn't need to eliminate x and z *exactly* between the first two equations, as long as we get one large and one small coefficient.]

Gauss–Jacobi

In this method, we start from a DD matrix, solve for x, y, *etc*., as above, then start from some initial x_0, y_0, … [often either zero, or else the result of putting all the x, y, … to zero on the RHS], and substitute in to the RHS to get new values x_1, y_1, …, and so on. This differs only slightly from:

Gauss–Seidel

In this method, we proceed as above, but as soon as the new value x_1 is available, we use it on the RHS [whereas G–J sticks to the same old x_0 throughout the first cycle], and similar for y, …. So, by the time we are nearing the end of the list of equations, we are using mostly the new values and only a few of the old ones.

This is more convenient for computer programming, as we don't need to keep copies of the old values, and might well seem more obvious. It usually converges slightly more quickly. But it is harder to do theoretical work with, and in the first cycle or two is slightly less convenient for hand calculation.

As usual in an iterative process, you do not need to stick to the rules. G–J and G–S will, *for a DD matrix,* converge from *any* starting values, so you can work to only a few significant figures in the early stages, or you can 'guess' where the values are heading towards.

In the case of a DD matrix, each cycle of the iteration in [say] G–S will use each element of the matrix once; if the process converges [to whatever accuracy we need] after, say, 10 cycles, this will have used each element 10 times. This is little or no better than direct elimination if the matrix is smaller than 10×10, but is *much* better if the matrix is large. So consider G–S *(a) if the matrix is DD* [or can easily be made so], and *(b)* if either the matrix is very large, or else the matrix is strongly DD [diagonal elements *much* larger than the others] and only limited accuracy is needed.

Ill-conditioned matrices

Consider, for example

$$\begin{pmatrix} \frac{1}{5} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{7} \end{pmatrix}^{-1} = \begin{pmatrix} 180 & -210 \\ -210 & 252 \end{pmatrix};$$

but [using 2sf on the LHS]

$$\begin{pmatrix} 0.20 & 0.17 \\ 0.17 & 0.14 \end{pmatrix}^{-1} \approx \begin{pmatrix} -156 & 189 \\ 189 & -222 \end{pmatrix};$$

and even

$$\begin{pmatrix} 0.200 & 0.167 \\ 0.167 & 0.142 \end{pmatrix}^{-1} \approx \begin{pmatrix} 201 & -135 \\ -135 & 281 \end{pmatrix}.$$

You need to use 5dp in the LHS to get the inverse correct even to the nearest integer—and this is a diddy 2×2 matrix with simple coefficients. Things can get rapidly much, much worse with, say, 5×5 matrices.

These 'ill-conditioned' matrices are characterised by having determinants much smaller than you might expect from the sizes of the elements: here $\frac{1}{35} - \frac{1}{36} = \frac{1}{1260} \approx 0.0008$, when you might have expected, say, 0.01 or so.

The problem is severe cancellation errors in the Gaussian elimination, and the same will happen no matter what you do to invert the matrix or to solve equations using it. The two rows are *almost* the same. This phenomenon is annoyingly common in real life. Sometimes there is no alternative to just doing things to great accuracy [like using Maple with *Digits* := 100; or more]. If you are lucky, then you can step back a little to where the equations came from, and subtract [or whatever] the equations *exactly* so as to get a decently-conditioned version.

Note that ill-conditioning is not the same as A having large or small elements;

$$\begin{pmatrix} 200 & 100 \\ 100 & 200 \end{pmatrix}, \begin{pmatrix} 0.02 & 0.01 \\ 0.01 & 0.02 \end{pmatrix}, \text{ even } \begin{pmatrix} 200 & 100 \\ 0.01 & 0.02 \end{pmatrix}$$

are no harder to invert or otherwise use than

$$\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}.$$

[Except that if the elements are particularly large/small, then for large matrices you may have to deal with extremely large/small numbers.]

Various matrix *norms* [beyond the present scope, in any detail] may be used to give warning signs of the problem. These include:

- Largest element of A times the largest element of its inverse numerically much larger than the size of the matrix.
  In the example, $0.2 \times 252 \approx 50$ is much greater than 2.

- Some eigenvalues much larger [numerically] than others.
  In the example, these are [as it happens] roughly 0.34 and 0.0023.