Numerical Methods: Jacobi and Gauss-Seidel Iteration

We can use row operations to compute a Reduced Echelon Form matrix row-equivalent to the augmented matrix of a linear system, in order to solve it exactly. For many simple systems (with few variables and integer coefficients, for example) this is an effective approach. However, for systems that come from physical situations that are interesting, it's quite likely that the resulting matrices would be too unwieldy to solve like this, with even computers prone to calculation errors.

To cope with this situation, mathematicians have developed *approximation techniques* that trade out the ability to solve a system exactly for the ability to get approximate solutions quickly and with low chances of error. The Jacobi and Gauss-Seidel Iteration techniques are two important examples, which are fairly simple to describe and carry out. As iteration techniques, the idea is to find a procedure for computing several "rounds" of approximations, each better than the last.

1 Equation Formulation

We can write out a linear system of m equations in m variables in the following way:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1m}x_m & = b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2m}x_m & = b_2, \\ \vdots & \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mm}x_n & = b_m. \end{cases}$$

We will re-write this system by solving the kth equation for x_k (assuming that the coefficient a_{kk} isn't zero—we could re-order the equations to try to avoid this). This gives us the (equivalent) system of equations

$$\begin{cases} x_1 & = \frac{1}{a_{11}}(b_1 - a_{12}x_2 - \cdots - a_{1n}x_n), \\ x_2 & = \frac{1}{a_{22}}(b_2 - a_{21}x_1 - a_{23}x_3 - \cdots - a_{2m}x_m), \\ \vdots & \vdots \\ x_m & = \frac{1}{a_{mm}}\left(b_m - a_{m1}x_1 - a_{m(m-1)}x_{m-1}\right). \end{cases}$$

That is, variable x_k is equal to $\frac{1}{a_{kk}}$ (its coefficient in the kth equation) times b_k (the constant from the kth equation) minus all of the other coefficient-and-variable terms from the kth equation.

It's good to take a moment to note that all we've done so far is find a set of equations that is equivalent to the original ones we had. This new form has the benefit that, if we were missing just one of the values in the solution, we could figure it out from the others directly (and exactly). From here, we must choose how to approximate the solution.

1.1 Jacobi Iteration

For Jacobi Iteration, we choose (through a guess or simply with a standard value) a starting value for each of x_1 through x_m, and call it the $n = 0$ value. Then, we use the right hand side of the equations we just found to compute the $n = 1$ value of each variable on the left hand side. That is, we think of n as the "stage" each variable is in, and imagine the left hand side of the equations as living in a higher "stage" than the right. For each n after 1, we use the previous "stage's" values (so we plug the $n = 1$ values into the right hand side to get the $n = 2$ values on the left, and so on).

1.2 Gauss-Seidel Iteration

For Gauss-Seidel Iteration, we do roughly the same thing as in Jacobi Iteration. However, we notice that when computing the variables in stage $n = i$ and for all the variables before x_k (x_1 through x_{k-1}), we have already computed a (theoretically) better approximation than the stage $(i-1)$ value: we already have the stage i value! So, in Gauss-Seidel Iteration, when computing stage $n = i$ we use the stage i values for all of the *previous* variables, and the stage $(i-1)$ values only for the *following* variables. Where Jacobi Iteration follows the simple rule of using each stage to compute the next, Gauss-Seidel Iteration improves on it by using the new approximations as soon as they are available.

1.3 Example

Suppose we had the system

$$\begin{cases} x_1 + x_2 &= 4, \\ x_1 - 2x_2 &= 1. \end{cases}$$

Rewriting it to the form used by the two iteration techniques, we get the equivalent system

$$\begin{cases} x_1 &= 4 - x_2, \\ x_2 &= \frac{-1}{2}(1 - x_1). \end{cases}$$

For the Jacobi Iteration method, we choose $x_1 = x_2 = 0$ as our starting ($n = 0$) value. Then, for $n = 1$, we get the new x_1 as $4 - 0 = 4$, and the new x_2 as $\frac{-1}{2}(1 - 0) = -0.5$ (plugging in the stage 0 values to the right hand sides of the equations). For $n = 2$, we get the new x_1 as $4 - (-0.5) = 4.5$, and the new x_2 as $\frac{-1}{2}(1 - 4) = 1.5$. Going a little further, we get the following table:

n	x_1	x_2
0	0	0
1	4	-0.5
2	4.5	1.5
3	2.5	1.75
4	2.25	0.75
5	3.25	0.625
6	3.375	1.125

For the Gauss-Seidel Iteration, we again start with $x_1 = x_2 = 0$ as the $n = 0$ value. Then for $n = 1$, we get the new x_1 as 4 again. The new x_2, on the other hand, depends on x_1, which we have already computed as 4 rather than 0, so we get the new x_2 as $\frac{-1}{2}(1 - 4) = 1.5$. For $n = 2$, we get the new x_1 as $4 - (1.5) = 2.5$, and the new x_2 as $\frac{-1}{2}(1 - 2.5) = 0.75$. Going further, we get the table:

n	x_1	x_2
0	0	0
1	4	1.5
2	2.5	0.75
3	3.25	1.125
4	2.875	0.9375
5	3.0625	1.03125
6	2.96875	0.984375

Since this is a system we can quickly solve exactly, we can check this against the true solution, $x_1 = 3$ and $x_2 = 1$. We can see that both appear to be approaching this, and Gauss-Seidel is somewhat faster.

It's worth noting that under certain circumstances, these approximation techniques can *diverge*; that is, fail to settle down and approach a single value. Sometimes rearranging the equations before solving for each variable can help, but other times the problem is intractable. Also, of course, neither technique can detect a solution if none exists.

2 Matrix Formulation

[If you're unfamiliar with matrix operations, feel free to skip this section (and perhaps return after it's covered later in the course.)]

The original system could be written as the matrix equation $A\vec{x} = \vec{b}$. If we *decompose* A as $A = D - U - L$, where D matches the main diagonal and has zeroes elsewhere, U matches the (negative of) the upper triangular part (above the diagonal), and L matches the (negative of) the lower triangular part (the negative signs and negated upper and lower triangular parts are just to conform to a particular, common convention).

Using this, Jacobi Iteration amounts to writing the equation as $D\vec{x} = (U + L)\vec{x} + \vec{b}$, then using the inverse matrix D^{-1} to solve for $\vec{x} = D^{-1}(U + L)\vec{x} + D^{-1}\vec{b}$. From here, we again treat the $\vec{x}$ on the left hand side as in the next stage we want to compute, and on the right hand side as in the stage already computed.

Gauss-Seidel Iteration amounts to writing the equation as $(D - L)\vec{x} = U\vec{x} + \vec{b}$, then inverting $(D - L)$ to solve for $\vec{x} = (D - L)^{-1}U\vec{x} + (D - L)^{-1}\vec{b}$. Again, the left $\vec{x}$ lives in the stage we are computing, and the right $\vec{x}$ lives in the previous stage. This description of the method has the slight advantage that it is expressed only as an operation on the previously computed stage, rather than appealing to what progress has been made towards the next one; however, it is really the same as before, because the actual operation simply duplicates the change done to that variable each time it occurs.

The matrix approach has the advantage of being more compact to express, although for the same reason it can be somewhat more difficult to follow what's going on. Working out the matrices involved will reveal the same equations, though. Ultimately, what's going on is the same, and either way of phrasing the procedure will yield the same results.