Lecture 03.09, Iterative methods for solving $Ax = b$

So far in this class we have talked about the two main problems in linear algebra: solving $Ax = b$ and solving $Ax = \lambda x$. We introduced a number of direct methods for $Ax = b$ (LU, Cholesky, banded) and we talked about direct methods for the least square problems, for the case when a solution doesn't actually exist and all we can do is find the closest approximation to it. We talked about the SVD and eigenvalue problems, and how one cannot hope to solve those directly for $n \geq 5$, emphasizing now the need for iterative (rather than direct) methods: power method and QR iteration, reduction to Hessenberg or bidiagonal/tridiagonal form.

In the last week of the quarter, we will come back to solving $Ax = b$, but this time we will proposed iterative methods for it. Why might we need such methods?

Recall that, in general, the operation count for a direct solving method is $O(n^3)$. This limits the size of the system you may work on; some systems are just too large. In fact, in industrial settings, some systems are just too large to fit in the fast memory of your computer! And by “too large” we mean that you may have hundreds of thousands of equations, with hundreds of thousands of variables.

Recall how, when setting up a linear system to numerically solve an ordinary differential equation, decreasing the stepsize $h$ is needed in order to increase accuracy (and to not get garbage). And that’s just when we attempt to solve on $[0, 1]$. Now consider trying to plot the trajectory of a shuttle flying to the Moon, or even clearer, a rocketship to Mars. Think how small (relative to the size of the problem) must the stepsize be in order for the rocketship to actually get there and not be lost forever in the vacuum of space, because of accumulated errors! The size of such systems would go well into millions of variables.

But even a hundred thousand is $10^6$, and if you cube that, you get $10^{18}$. Even if computations are done at the speed of $3\text{GHz}$, or $30 \times 10^9$ operations per second, and even if memory were not an issue, one would need $3.33 \times 10^8$ seconds to solve such a system. And if you realize that there are 3600 seconds in an hour, that means roughly $10^5$ hours, or 416 days to solve such a system!

Which is why linear algebraists are always on the lookout for faster methods, and faster in this case almost always means “iterative”. Direct methods, that is, methods which will work and yield perfect answers in exact arithmetic, cannot so far take us there fast enough.

We will learn this week two “classical” iterative methods for $Ax = b$, the Jacobi method and the Gauss-Seidel method, which are similar but not the same.

We will start with the Jacobi method. This method is based on a simple observation, which is that if we solve a $n \times n$ system $Ax = b$, we will necessarily have, for each $1 \leq i \leq n$,

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \quad \text{or, rewritten,} \quad x_i = \frac{1}{a_{ii}} \left( b_i - \sum_{j \neq i} a_{ij} x_j \right).$$

Of course, this assumes $a_{ii} \neq 0$, for all $i$. The true solution means that each component satisfies an “averaging out” equation as the one above.

This will be the basis for the following recurrence: start at some “guess” $x_0$, and then perform

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left( b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right).$$
Intuitively, this will allow us to approach the solution (although we will see that it will happen only under certain conditions).

We can iterate this recurrence until, for example, we see that \( \text{norm}(x^{(k+1)} - x^{(k)}) < \epsilon \), where \( \epsilon \) is some tolerance we set; when that happens, we can declare convergence.

**Example.** Let \( A = \begin{bmatrix} 4 & 1 & 3 \\ 1 & 5 & 1 \\ 2 & -1 & 8 \end{bmatrix} \) and \( b = \begin{bmatrix} 17 \\ 14 \\ 12 \end{bmatrix} \).

Start with \( x^{(0)} = 0 \); we obtain

\[
\begin{align*}
  x^{(1)} &= \begin{bmatrix} 17/4, 14/5, 3/2 \end{bmatrix}^T = \begin{bmatrix} 4.25, 2.8, 1.5 \end{bmatrix}^T \\
  x^{(2)} &= \begin{bmatrix} 97/40, 33/20, 63/80 \end{bmatrix}^T = \begin{bmatrix} 2.425, 1.65, 0.7875 \end{bmatrix}^T \\
  x^{(3)} &= \begin{bmatrix} 3.246875, 2.1575, 1.1 \end{bmatrix}^T \\
  x^{(4)} &\approx \begin{bmatrix} 2.885, 1.93, 0.96 \end{bmatrix}^T \\
  x^{(5)} &\approx \begin{bmatrix} 3.05, 2.03, 1.02 \end{bmatrix} \\
  x^{(6)} &\approx \begin{bmatrix} 2.98, 1.99, 0.99 \end{bmatrix}^T, \text{ etc.}
\end{align*}
\]

True solution \( [3, 2, 1]^T \).

What is going on? Denote by \( D \) the diagonal part of \( A \).

Note that \( x^{(k+1)} = D^{-1}(b - (A - D)x^{(k)}) \). Rewrite \( r^{(k)} = b - Ax^{(k)} \), the \( k \)th residual. Then

\[
x^{(k+1)} = D^{-1}(b - (A - D)x^{(k)}) = D^{-1}(b - Ax^{(k)} + Dx^{(k)}) = D^{-1}r^{(k)} + x^{(k)} ,
\]

or

\[
x^{(k+1)} = x^{(k)} + D^{-1}r^{(k)} .
\]

We will get even more out of this recurrence, next time.