Iterative Methods for $A\mathbf{x} = \mathbf{b}$

Question 1. We have a direct method in the $PA = LU$ factorization. Why would we want an iterative method?

Mainly its speed. If an iterative method can give a good approximation faster than the $PA = LU$ factorization (recall the $\frac{2n^3}{3}$ operation count), then it might be worth the tradeoff in accuracy.

We'll be looking at three methods (which are all related):

1. Jacobi Method
2. Gauss-Seidel Method
3. SOR Method (Successive Over-Relaxation)

While that sounds like a lot, all three are variations on a theme.

Jacobi Method

The Jacobi Method is a spin on Fixed-Point Iteration. We generate a sequence of vectors $\mathbf{x}_i$, $i = 0, 1, 2, \ldots$ where $\mathbf{x}_0$ is the initial guess for the solution. To describe the process, we need notation for the elements of the vector $\mathbf{x}_i$, which we will denote as $\mathbf{x}_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{in} \end{bmatrix}$. For instance, the second guess of the sequence when solving a 3×3 system is:

$$\begin{bmatrix} x_{21} \\ x_{22} \\ x_{23} \end{bmatrix}.$$

The idea is to solve for each element in terms of the others. So solve for x_{i1} in the first row. Solve for x_{i2} in the second row. And so on - you can find a formula for all of them as long as the diagonal elements aren't zero; if they are, try rearranging the rows before applying the process. We use these generated formulas to get new values for each from the current guess.

Example 1. Apply three steps of the Jacobi method to estimate the solution to $\begin{bmatrix} 3 & 1 \\ 1 & -2 \end{bmatrix}\mathbf{x} = \begin{bmatrix} 10 \\ 1 \end{bmatrix}$ with an initial guess of $\mathbf{x}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

Let's take a moment to find the exact answer using Gaussian Elimination. $\begin{bmatrix} 3 & 1 & 10 \\ 1 & -2 & 1 \end{bmatrix} \sim \begin{bmatrix} 1 & -2 & 1 \\ 3 & 1 & 10 \end{bmatrix} \sim \begin{bmatrix} 1 & -2 & 1 \\ 0 & 7 & 7 \end{bmatrix} \sim \begin{bmatrix} 1 & -2 & 1 \\ 0 & 1 & 1 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \end{bmatrix}$. So we hope the iteration goes towards $\begin{bmatrix} 3 \\ 1 \end{bmatrix}$.

The original system

$$\begin{aligned}3x_1 + x_2 &= 10\\ x_1 - 2x_2 &= 1\end{aligned}$$

becomes

$$\begin{aligned}x_1 &= \frac{10 - x_2}{3}\\ x_2 &= \frac{x_1 - 1}{2}.\end{aligned}$$

So $\mathbf{x}_1 = \begin{bmatrix}\frac{10-1}{3}\\ \frac{1-1}{2}\end{bmatrix} = \begin{bmatrix}3\\0\end{bmatrix}$
and $\mathbf{x}_2 = \begin{bmatrix}\frac{10-0}{3}\\ \frac{3-1}{2}\end{bmatrix} = \begin{bmatrix}\frac{10}{3}\\1\end{bmatrix}$.
Finally, $\mathbf{x}_3 = \begin{bmatrix}\frac{10-1}{3}\\ \frac{\frac{10}{3}-1}{2}\end{bmatrix} = \begin{bmatrix}3\\ \frac{7}{6}\end{bmatrix}$. Note that while the individual elements might be less correct in a step, the overall error as measured by one of our norms is decreasing. The Jacobi Method in this example is therefore converging.

Example 2. Apply three steps of the Jacobi method to estimate the solution to $\begin{bmatrix}1 & -2\\3 & 1\end{bmatrix}\mathbf{x} = \begin{bmatrix}1\\10\end{bmatrix}$ with an initial guess of $\mathbf{x}_0 = \begin{bmatrix}1\\1\end{bmatrix}$.

The original system

$$\begin{aligned}x_1 - 2x_2 &= 1\\ 3x_1 + x_2 &= 10\end{aligned}$$

becomes

$$\begin{aligned}x_1 &= 1 + 2x_2\\ x_2 &= 10 - 3x_1.\end{aligned}$$

So $\mathbf{x}_1 = \begin{bmatrix}1+2\\10-3\end{bmatrix} = \begin{bmatrix}3\\7\end{bmatrix}$
and $\mathbf{x}_2 = \begin{bmatrix}1+2(7)\\10-3(3)\end{bmatrix} = \begin{bmatrix}15\\1\end{bmatrix}$.
Finally, $\mathbf{x}_3 = \begin{bmatrix}1+2(1)\\10-3(15)\end{bmatrix} = \begin{bmatrix}3\\-35\end{bmatrix}$.
Note that this system is the same as the first, with the rows swapped in order. Yet this is not nearly as close to the real answer of $\begin{bmatrix}1\\3\end{bmatrix}$. The Jacobi Method in this example is diverging. More to come on convergence...

The Jacobi Method is a rewriting of the original system of equations, and is still a system of equations. So, we describe the method in matrix form for convenient programming. Start with the matrix A and break it into three parts: L, for the sub-diagonal elements, D for the diagonal elements, and u for the super-diagonal elements so that $A = L + D + U$. In particular,

$$A = \begin{bmatrix} 3 & 1 \\ 1 & -2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} + \begin{bmatrix} 3 & 0 \\ 0 & -2 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}.$$

Then because we are solving for the diagonal elements, we are writing the system as:

$$D\mathbf{x}_{i+1} = \mathbf{b} - U\mathbf{x}_i - L\mathbf{x}_i.$$

As long as D has no zeros on the diagonal (which would be a real problem for the method, as we couldn't solve for that element of $\mathbf{x}$ in that row), we have that D^{-1} is also a diagonal matrix, and its diagonal elements are the reciprocals of the diagonal elements of D:

$$\mathbf{x}_{i+1} = D^{-1}(\mathbf{b} - (L + U)\mathbf{x}_i).$$

We can now see how the method is related to fixed point iteration.

Example 3. Use the matrix form of the Jacobi method to find the first two estimates with initial guess of the zero vector for the system given by $A = \begin{bmatrix} 4 & -1 & 1 \\ -1 & 8 & 4 \\ 2 & 1 & 16 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} 16 \\ 16 \\ 16 \end{bmatrix}$.

Since $A = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ 2 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 4 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 16 \end{bmatrix} + \begin{bmatrix} 0 & -1 & 1 \\ 0 & 0 & 4 \\ 0 & 0 & 0 \end{bmatrix}$, we have

$$\mathbf{x}_1 = \begin{bmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{8} & 0 \\ 0 & 0 & \frac{1}{16} \end{bmatrix} \cdot \left(\begin{bmatrix} 16 \\ 16 \\ 16 \end{bmatrix} - \begin{bmatrix} 0 & -1 & 1 \\ -1 & 0 & 4 \\ 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{8} & 0 \\ 0 & 0 & \frac{1}{16} \end{bmatrix} \cdot \left(\begin{bmatrix} 16 \\ 16 \\ 16 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 4 \\ 2 \\ 1 \end{bmatrix}.$$

This is followed by $\mathbf{x}_2 =$

$$\begin{bmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{8} & 0 \\ 0 & 0 & \frac{1}{16} \end{bmatrix} \cdot \left(\begin{bmatrix} 16 \\ 16 \\ 16 \end{bmatrix} - \begin{bmatrix} 0 & -1 & 1 \\ -1 & 0 & 4 \\ 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \\ 1 \end{bmatrix} \right) = \begin{bmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{8} & 0 \\ 0 & 0 & \frac{1}{16} \end{bmatrix} \cdot \left(\begin{bmatrix} 16 \\ 16 \\ 16 \end{bmatrix} - \begin{bmatrix} -1 \\ 0 \\ 10 \end{bmatrix} \right) = \begin{bmatrix} \frac{17}{4} \\ 0 \\ \frac{3}{8} \end{bmatrix}.$$

Gauss-Seidel

Essentially, the Gauss-Seidel Method is the Jacobi Method with the following realization. Wait, I've already computed better values for some of these! The hope is that this will lead to better accuracy sooner (faster convergence).

Example 4. Using the same matrices from Example 3, the Jacobi method was solving:

$$\begin{aligned} x_1 &= \frac{16 + x_2 - x_3}{4} \\ x_2 &= \frac{16 + x_1 - 4x_3}{8} \\ x_3 &= \frac{16 - 2x_1 - x_2}{16} \end{aligned}$$

The Gauss-Seidel method says: when solving for x_2, use the newer value for x_1. When solving for x_3, use the new values for both x_1 and x_2.

So with the initial guess of zeros, we have

$$\begin{aligned} x_{11} &= \frac{16 + 0 - 0}{4} = 4 \\ x_{12} &= \frac{16 + 4 - 4(0)}{8} = \frac{5}{2} \\ x_{13} &= \frac{16 - 2(4) - \frac{5}{2}}{16} = \frac{11}{32} \end{aligned}$$

so $\mathbf{x}_1 = \begin{bmatrix} 4 \\ \frac{5}{2} \\ \frac{11}{32} \end{bmatrix}$.

The second step is:

$$\begin{aligned} x_{21} &= \frac{16 + \frac{5}{2} - \frac{11}{32}}{4} = \frac{581}{128} \\ x_{22} &= \frac{16 + \frac{581}{128} - 4(\frac{11}{32})}{8} = \frac{1387}{579} \\ x_{23} &= \frac{16 - 2\frac{581}{128} - \frac{1387}{579}}{16} = \frac{574}{2029} \end{aligned}$$

so $\mathbf{x}_2 \approx \begin{bmatrix} 4.5391 \\ 2.3955 \\ 0.2829 \end{bmatrix}$. For reference, the true answer is about $\begin{bmatrix} 4.5363 \\ 2.4264 \\ 0.2813 \end{bmatrix}$ so that's quite good for two iterations.

Example 5. For the system given by $A = \begin{bmatrix} 4 & 0 & 4 \\ 8 & 8 & 0 \\ 0 & 1 & 8 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} 16 \\ 16 \\ 16 \end{bmatrix}$, apply two steps of Gauss-Seidel with an initial guess of zeros.

$$\begin{aligned} x_1 &= \frac{16 - 4x_3}{4} = 4 - x_3 \\ x_2 &= \frac{16 - 8x_1}{8} = 2 - x_1 \\ x_3 &= \frac{16 - x_2}{8} \end{aligned}$$

We have:

$$\begin{aligned} x_{11} &= 4 - 0 = 4 \\ x_{12} &= 2 - 4 = -2 \\ x_{13} &= \frac{16 - (-2)}{8} = \frac{9}{4} \end{aligned}$$

so $\mathbf{x}_1 = \begin{bmatrix} 4 \\ -2 \\ \frac{9}{8} \end{bmatrix}$. Then,

$$\begin{aligned} x_{21} &= 4 - \frac{9}{4} = \frac{7}{4} \\ x_{22} &= 2 - \frac{7}{4} = \frac{1}{4} \\ x_{23} &= \frac{16 - \frac{1}{4}}{8} = \frac{63}{32} \end{aligned}$$

so $\mathbf{x}_2 = \begin{bmatrix} \frac{7}{4} \\ \frac{1}{4} \\ \frac{63}{32} \end{bmatrix}$.

Gauss-Seidel also has a matrix form. Expressed in the same way as Jacobi, it is:

$$\mathbf{x}_{i+1} = D^{-1}(\mathbf{b} - U\mathbf{x}_i - L\mathbf{x}_{i+1}).$$

This is not so great for thinking of it as fixed point iteration, though, because the vector we're solving for is on both sides of the equation. Isolating $\mathbf{x}_{i+1}$ gives instead

$$\mathbf{x}_{i+1} = (L + D)^{-1}(\mathbf{b} - U\mathbf{x}_i)$$

which now has the disadvantage that $L + D$ isn't nearly as nice to invert as just D was. However, in practice, we don't fully invert them anyway - we use back substitution, and $L + D$ IS still triangular.

Successive Over-Relaxation (SOR)

The final iterative method starts with Gauss-Seidel and adds the idea that the new computation is a good direction to move to get closer to the real solution - so let's move further that way. The hope is that this will converge faster still.

We introduce a weight ω (omega) to balance the current solution with the next one. If $\omega = \frac{1}{2}$, SOR averages together the current solution and the next one.

Example 6. Write the SOR equations for the system of Example 3.

$$
\begin{aligned}
x_{i+1,1} &= (1-\omega)x_{i1} + \omega\left(\frac{16 + x_{i2} - x_{i3}}{4}\right)\\
x_{i+1,2} &= (1-\omega)x_{i2} + \omega\left(\frac{16 + x_{i+1,1} - 4x_{i3}}{8}\right)\\
x_{i+1,3} &= (1-\omega)x_{i3} + \omega\left(\frac{16 - 2x_{i+1,1} - x_{i+1,2}}{16}\right)
\end{aligned}
$$

We are interested in what happens for $\omega > 1$. The parameter ω is called the *relaxation parameter* and when it is more than 1, we then have *over-relaxation.*

Computing the first step for $\omega = 1.5$ with the initial guess of zeros:

$$
\begin{aligned}
x_{i+1,1} &= (1-1.5)(0) + 1.5\left(\frac{16}{4}\right) = 6\\
x_{i+1,2} &= (1-1.5)(0) + 1.5\left(\frac{16 + 6 - 4(0)}{8}\right) = \frac{33}{8}\\
x_{i+1,3} &= (1-1.5)(0) + 1.5\left(\frac{16 - 2(6) - \frac{33}{8}}{16}\right) = -\frac{3}{256}
\end{aligned}
$$

In matrix form, SOR is:

$$
\mathbf{x}_{i+1} = (\omega L + D)^{-1}[(1-\omega)D\mathbf{x}_i - \omega U\mathbf{x}_i] + \omega(\omega L + D)^{-1}\mathbf{b}
$$

Convergence

None of the iterative methods presented are guaranteed to converge for all matrices A and all initial guesses. However, there is a property of the matrix A that will guarantee convergence for all initial guesses.

Definition 2. A $n \times n$ matrix A is *strictly diagonally dominant* if, for each row, $1 \le i \le n$, the absolute value of the diagonal element, $|a_{ii}|$, is larger than the total of the absolute values of all other elements in the row, $\sum_{j\neq i}|a_{ij}|$.

Example 7. Determine if each of the matrices in the previous examples are strictly diagonally dominant.

1 - yes, 2 - no, 3 and 4 - yes, 5 - no. While Gauss-Seidel worked for the initial guess there, it was not guaranteed to do so, and in fact, SOR with that initial guess and over-relaxed values of the parameter seem to diverge.

Theorem 3. *If the $n \times n$ matrix A is strictly diagonally dominant, then A is invertible and for every vector $\boldsymbol{b}$ and every starting guess, the Jacobi Method applied to $A\boldsymbol{x} = \boldsymbol{b}$ converges to the unique solution. Also, the Gauss-Seidel Method applied to $A\boldsymbol{x} = \boldsymbol{b}$ converges to the unique solution.*

Question 4. Are these iterative methods faster?

It depends on how many iterations, but typically yes for large matrices. Looking at the operation count for Jacobi, $L+U$ doesn't actually require any work - just reference the right parts of A. Then multiplication by $\mathbf{x}$ is $2n^2$ for a matrix-vector product. Then subtraction from $\mathbf{b}$ is n operations and because D is diagonal, that's n divisions for a total of $2n^2+2n \approx 2n^2$ per iteration. As long as the iteration number p is lower than n, $2n^2 \cdot (p) << \frac{2n^3}{3}$.

Similar analysis based on the matrix versions of Gauss-Seidel and SOR give that they are roughly $3n^2$ per iteration.