# Basics of Iterative Methods for Linear Systems

Justin L. Clough

November 9, 2018

Consider solving the system $Ax = b$ with $x, b \in \mathcal{R}^n$; assume that a unique exact solution, $x^*$, exists. For any row $r$ of this system:

$$a_{r1}x_1 + a_{r2}x_2 + a_{r3}x_3 + ... + a_{rn}x_n - b_r = 0 \tag{1}$$

$$\sum_{i=1}^{n} a_{ri}x_i - b_r = 0 \tag{2}$$

If we solve for any particular $x_j$;

$$a_{rj}x_j = b_r - \sum_{\substack{i=1 \\ i \neq j}}^{n} a_{ri}x_i \tag{3}$$

$$x_j = \frac{1}{a_{rj}} \left( b_r - \sum_{\substack{i=1 \\ i \neq j}}^{n} a_{ri}x_i \right) \tag{4}$$

But if $a_{rj} = 0$ then our method blows up! Note that if we choose to always work on the diagonal (i.e., choose $r = j$), then we are okay since most physics based problems have $a_{jj} \neq 0$. We may also now decompose $A$ to a diagonal, lower, and upper matrices $D, L, U$ such that $A = D - L - U$. The method is now:

$$x_j = \frac{1}{a_{jj}} \left( b_j - \sum_{\substack{i=1 \\ i \neq j}}^{n} a_{ji}x_i \right) \tag{5}$$

$$x = D^{-1}(b + (L + U)x) \tag{6}$$

But we can only use the method to solve for element $x_j$ if we already know all other elements of $x$! Instead, use $x$ from the last (or initial guess) to update our current guess. Let the superscript represent the step number, e.g., $a^k$ is the k$^{\text{th}}$ step of estimating $a$ granted an initial guess $a^0$ has been given.

$$x_j^{k+1} = \frac{1}{a_{jj}} \left( b_j - \sum_{\substack{i=1 \\ i \neq j}}^{n} a_{ji}x_i^k \right) \tag{7}$$

$$x^{k+1} = D^{-1}(b + (L + U)x^k) \tag{8}$$

The above is defined as the *Jacobi Method* to estimate $x^{k+1}$.

Observe that when calculating $x_j^k$, we have already calculated updated values of $x_p^k$ $\forall p = 1, 2, ...j-1$. Using these values right away gives the method:

$$x_j^{k+1} = \frac{1}{a_{jj}} \left( b_j - \sum_{i=1}^{j-1} a_{ji} x_i^{k+1} - \sum_{i=j+1}^{n} a_{ji} x_i^k \right) \tag{9}$$

$$x^{k+1} = (D-L)^{-1} U x^{k+1} + (D-L)^{-1} b \tag{10}$$

Above the is the *Gauss-Seidel* method. But what if $|a_{jj}| << 1$? This will cause issues for Floating Point Evaluations (FPE). Instead "relax" the correction between iterations by $\omega \in (0, 2)$:

$$x_j^{k+1} = (1-\omega) x_j^k + \frac{\omega}{a_{jj}} \left( b_j - \sum_{i=1}^{j-1} a_{ji} x_i^{k+1} - \sum_{i=j+1}^{n} a_{ji} x_i^k \right) \tag{11}$$

$$x^{k+1} = (1-\omega) x^k + \omega \left( (D-L)^{-1} U x^{k+1} + (D-L)^{-1} b \right) \tag{12}$$

which then defines the *Successive Over Relaxation Method* (SOR). All three above methods are *Stationary Iterative Methods* as the interim estimates $x^k$ can both over or under estimate the exact, stationary, solution $x^*$.

Another group of iterative solvers are *Krylov Subspace Methods*. This group includes the *Conjugate Gradient* (CG) and *Generalized Minimum Residual* (GMRES) methods. These methods work by successively building a subspace of the solution space until the solution space (and solution) is sufficiently represented. The CG method is described below.

Want to again solve $Ax = b$ where $x, b \in \mathcal{R}^n$ and it is assumed that a unique exact solution $x^*$ exists. We also restrict $A$ to be symmetric and positive-definite. First, define the $A$ *inner product* between $u, v \in \mathcal{R}^n$ as:

$$u^T A v = (u, v)_A \tag{13}$$

and if this $A$ inner product is zero, then $u$ and $v$ are $A$-*orthogonal* with respect to each other. Let there exists a set of vectors $\mathcal{P} = \{p_i | p_i \in \mathcal{R}^n, (p_i, p_j)_A = \delta_{ij} \forall i, j = 1, 2, ..., n\}$. This means that $\{p_i\}_{i=1}^n$ form a basis of $\mathcal{R}^n$ which is where the solution lives:

$$x^* = \sum_{i=1}^{n} \alpha_i p_i \tag{14}$$

with $\alpha_i \in \mathcal{R}$. Returning to the original problem, apply the new information, and left-dotting with $p_k^T$:

$$Ax = b \tag{15}$$

$$A\sum_{i=1}^{n} \alpha_i p_i = b \tag{16}$$

$$p_k^T A\sum_{i=1}^{n} \alpha_i p_i = p_k^T b \tag{17}$$

$$\sum_{i=1}^{n} \alpha_i (p_k, p_i)_A = (p_k, b) \tag{18}$$

But recall that $(p_k, p_i) = \delta_{ik}$ and so the summation breaks down into a single term:

$$\alpha_i (p_i, p_i)_A = (p_i, b) \tag{19}$$

$$\Rightarrow \alpha_i = \frac{(p_i, b)}{(p_i, p_i)_A} \tag{20}$$

There is now a way to calculate the $\{\alpha_i\}_{i=1}^n$. But where do the $\{p_i\}_{i=n}^n$ come from? To generate $\{p_i\}_{i=n}^n$, repose the linear problem as:

$$r^k = b - Ax^k \tag{21}$$

where the previous iteration notation is used and $r^k$ is the residual from using the $\mathrm{k}^{\mathrm{th}}$ guess, $x^k$. Naively, we want to "minimize" $r$ but it is a vector; instead define:

$$f(x) = \frac{1}{2} x^T A x - x^T b \tag{22}$$

which has the following properties:

1. $f(x)$ has a minimum at $x = x^*$ (i.e., $r = 0$).
2. $f(x)$ is quadratic with respect to $x$.
3. $f(x)$ is positive so long as $A$ is Symmetric-Positive-Definite.
4. The gradient of $f(x^k) = -r^k$; i.e., $r^k$ "points downhill" in the steepest direction at the guess $k$.

Now pick any $x^0$ as an initial guess which gives a corresponding $r^0$ which is taken as the first basis. We calculate $\alpha_0$ as shown and let $x^1 = x^0 + \alpha_0 p_0$. To guarantee $(p_i, p_k) = \delta_{ik}$ we use a Gram-Schmidt based method:

$$p_k = r_k - \sum_{i=1}^{k-1} \frac{(p_i, r_k)_A}{(p_i, p_i)_A} p_i \tag{23}$$

We now have a method to calculate both $\{\alpha_i\}_{i=1}^n$ and $\{p_i\}_{i=1}^n$. This method can be used over $k$ steps until $||r^k||_{L_2}$ is below an acceptable tolerance. Note that (with exact calculations) $x^*$ is guaranteed in $n$ steps!