# Day 12: Iterative Methods for Linear Systems

Scaling AI to Millions of Dimensions Without Direct Inversion

## 1. The Limitation of Direct Methods

Direct methods like Gaussian Elimination or LU decomposition find an exact solution in a finite number of steps. However, their computational cost is $O(n^3)$ for a dense $n \times n$ matrix. This becomes prohibitive for the massive systems ($n > 10^6$) common in AI (e.g., PDEs, graph networks, large optimization problems).

**Iterative methods** approximate the solution $\mathbf{x}$ to $A\mathbf{x} = \mathbf{b}$ by starting with an initial guess $\mathbf{x}^{(0)}$ and producing a sequence of improving estimates $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots$ until convergence. Their key advantage is that they only require **matrix-vector products** $A\mathbf{v}$, which are $O(n^2)$ and can exploit sparsity to become $O(n)$.

## 2. Stationary Methods: Jacobi and Gauss-Seidel

These are among the simplest iterative methods. They decompose $A$ into its diagonal ($D$), lower ($L$), and upper ($U$) triangular parts: $A = D + L + U$.

**Iterative Update Rules**

- **Jacobi Method:** Updates all variables simultaneously using the previous iteration.
$$\mathbf{x}^{(k+1)} = D^{-1}(\mathbf{b} - (L+U)\mathbf{x}^{(k)})$$
- **Gauss-Seidel Method:** Updates variables sequentially, using the newest values immediately.
$$\mathbf{x}^{(k+1)} = (D+L)^{-1}(\mathbf{b} - U\mathbf{x}^{(k)})$$

**Convergence:** Guaranteed if $A$ is strictly diagonally dominant or symmetric positive definite (SPD). Gauss-Seidel is typically faster than Jacobi.

## 3. The Conjugate Gradient (CG) Method

The CG method is the **cornerstone iterative method** for large-scale SPD systems. It's not just an iterative solver; it's an optimization algorithm that minimizes the quadratic form $\frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{b}^T\mathbf{x}$.

**The CG Algorithm (Conceptual)**

For SPD matrix $A$, initial guess $\mathbf{x}_0$:

1. Initialize: residual $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$, search direction $\mathbf{p}_0 = \mathbf{r}_0$.
2. For $k = 0, 1, \ldots$ until convergence:

$$\begin{aligned}
\alpha_k &= \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{p}_k^T A \mathbf{p}_k} \quad \text{(step size)} \\
\mathbf{x}_{k+1} &= \mathbf{x}_k + \alpha_k \mathbf{p}_k \\
\mathbf{r}_{k+1} &= \mathbf{r}_k - \alpha_k A \mathbf{p}_k \\
\beta_k &= \frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k} \\
\mathbf{p}_{k+1} &= \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k \quad \text{(new conjugate direction)}
\end{aligned}$$

**Why it's powerful:** In exact arithmetic, CG converges in at most $n$ steps. In practice, with a good preconditioner, it often converges in far fewer steps, making it suitable for very large $n$.

## 4. Krylov Subspace Methods

CG is a member of the broader family of **Krylov subspace methods**. These methods search for the solution $\mathbf{x}_k$ in the Krylov subspace:

$$\mathcal{K}_k = \text{span}\{\mathbf{r}_0, A\mathbf{r}_0, A^2\mathbf{r}_0, \ldots, A^{k-1}\mathbf{r}_0\}$$

Other famous members include:

- **GMRES:** For non-symmetric matrices. Minimizes the residual norm $\|\mathbf{b} - A\mathbf{x}\|_2$ over the Krylov subspace.
- **BiCGSTAB:** A variant for non-symmetric systems that avoids the long recurrences of GMRES.

## 5. The Ultimate Iterative Method in AI: Stochastic Gradient Descent (SGD)

While CG solves linear systems, AI's most critical problem is non-linear optimization. **SGD and its variants (Adam, etc.) are iterative methods for this purpose.**

The connection is profound: SGD can be seen as an iterative method for solving the *normal equations* $X^T X\mathbf{w} = X^T\mathbf{y}$ of linear regression, but where each update uses a noisy, stochastic approximation of the true gradient (a single data point or mini-batch). This makes it scalable to datasets with millions of samples ($n$) and features ($d$).

## 6. Applications in AI/ML

**Where Iterative Methods Power AI**

- **Deep Learning:** Training via SGD, Adam, etc. is iterative optimization. Second-order methods (like K-FAC) use CG to approximate the inverse Hessian.
- **Large-Scale Linear Models:** Solving ridge regression $(X^TX + \lambda I)\mathbf{w} = X^T\mathbf{y}$ for large $d$ using CG.
- **Graph Neural Networks:** Propagating information on large graphs often involves solving systems related to the graph Laplacian, done iteratively.
- **Gaussian Processes:** Solving $(K + \sigma_n^2 I)^{-1}\mathbf{y}$ for inference is done with CG (as $K$ is dense and $n$ is large).
- **Reinforcement Learning:** Policy evaluation in Dynamic Programming methods like Value Iteration is an iterative algorithm.

## Key Takeaway

Iterative methods are the workhorses of large-scale numerical computing and AI. They trade the exactness of direct methods for the scalability and efficiency of approximation. From solving massive linear systems with Conjugate Gradient to training deep networks with SGD, they provide the fundamental algorithms that make modern AI computationally feasible.