PART I. OPTIMIZATION: CLASSICAL APPROACHES
(LECTURE 4)

Shpilev Petr Valerievich
Faculty of Mathematics and Mechanics, SPbU

September, 2025

Санкт-Петербургский государственный университет

Comments

In this lecture, we will consider the so-called hard case arising from the application of the More-Sorensen's Theorem. The rest of the lecture is devoted to the consideration of conjugate gradient method and its practical implementations.

The Hard Case in Trust-Region Subproblem

Assumption Relaxed: If $q_1^T g = 0$ or eigenvalue λ_1 is multiple with $Q_1^T g = 0$ (where Q_1 is the matrix whose columns span the subspace corresponding to the eigenvalue λ_1), the root-finding fails: there may not be a value $\lambda \in (-\lambda_1, \infty)$ s.t. $\|p(\lambda)\| = \Delta$. This is called the hard case.

Solution: Take $\lambda = -\lambda_1$, as guaranteed by Theorem 12. Use:

$$p = \sum_{\lambda_j \neq \lambda_1} \frac{q_j^T g}{\lambda_j + \lambda} q_j + \tau z$$

Where: $(B - \lambda_1 I)z = 0$, $\|z\| = 1$, z orthogonal to all q_j with $\lambda_j \neq \lambda_1$.

Then:

$$\|p\|^2 = \sum_{\lambda_j \neq \lambda_1} \frac{(q_j^T g)^2}{(\lambda_j + \lambda)^2} + \tau^2$$

Choose τ such that $\|p\| = \Delta$. Then p and $\lambda = -\lambda_1$ satisfy the optimality conditions (4a)-(4c) of Theorem 12.

Comments

Up to now we assumed that the condition $q_1^T g \neq 0$, which ensures that the root-finding equation has a solution inside the interval from $-\lambda_1$ to infinity. However, what happens if this assumption fails? Suppose that $q_1^T g = 0$, or more generally, that λ_1 is a multiple eigenvalue and the entire eigenspace associated with λ_1 is orthogonal to g. In this situation, the root-finding approach no longer works, because there may be no value of λ in the open interval from $-\lambda_1$ to infinity that makes the norm of $p(\lambda)$ equal to Δ.

This pathological scenario is called the hard case. At first, it might seem that no solution exists, but Moré-Sorensen's theorem guarantees that a correct solution still lies within the closed interval from $-\lambda_1$ to infinity. In fact, there is only one possibility: λ must be equal to $-\lambda_1$.

Once λ is fixed at $-\lambda_1$, we must carefully construct the step p. We cannot simply omit the components corresponding to eigenvalue λ_1. Instead, we recognize that the matrix $B - \lambda_1 I$ is singular. Therefore, there exists a vector z, of unit norm, such that $(B - \lambda_1 I)z = 0$. This vector z is precisely an eigenvector corresponding to the eigenvalue λ_1. Moreover, it is orthogonal to all eigenvectors associated with other eigenvalues.

Using this property, we can express p as the sum of two terms. The first term is the usual series over all eigenvalues not equal to λ_1: for each such index j, we take $q_j^T g/(\lambda_j + \lambda)$ multiplied by q_j. The second term is τz, where τ is a scalar parameter. The squared norm of p is then the sum of the squared contributions of the first part plus τ^2. By choosing τ appropriately, we can always ensure that the norm of p equals Δ. This construction gives us both p and $\lambda = -\lambda_1$ that satisfy the optimality conditions of the trust-region subproblem.

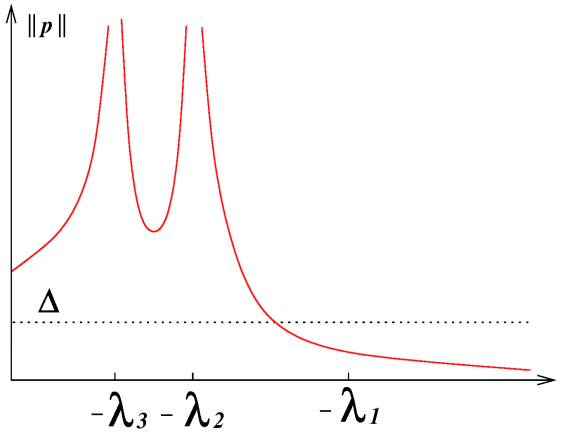

Figure: The hard case: $\|p(\lambda)\| < \Delta$ for all $\lambda \in (-\lambda_1, \infty)$.

- $\|p(\lambda)\| < \Delta$ for all $\lambda \in (-\lambda_1, \infty)$.
- No λ solves $\|p(\lambda)\| = \Delta$ in this interval.
- Occurs when $Q_1^T g = 0$ for eigenspace Q_1 of λ_1.

Comments

The hard case can also be visualized graphically. Recall that in the standard situation, as λ increases from values close to $-\lambda_1$, the norm of $p(\lambda)$ eventually decreases and crosses the trust-region radius Δ. That crossing point defines λ^*. But in the hard case, this crossing never happens.

Specifically, for every λ strictly greater than $-\lambda_1$, the computed step norm remains strictly less than Δ. In other words, the curve representing the norm of $p(\lambda)$ lies entirely below the horizontal line at height Δ, throughout the entire interval from $-\lambda_1$ to infinity. As a consequence, there is no root of the equation $\|p(\lambda)\| = \Delta$ within this range.

This behavior occurs exactly when the projection of g onto the eigenspace corresponding to the smallest eigenvalue λ_1 vanishes. Equivalently, the subspace spanned by the eigenvectors for λ_1 is orthogonal to g. In this case, the search direction never accumulates enough length to reach the boundary of the trust region, no matter how large λ becomes.

Nevertheless, the theory assures us that the trust-region problem still has a valid solution. The solution is obtained by setting $\lambda = -\lambda_1$ and then augmenting the step with a suitable component along the eigenvector z of that eigenvalue. By adjusting the scalar τ, as discussed earlier, we can inflate the norm of p to exactly Δ.

This picture underscores the importance of theoretical guarantees in numerical optimization. Even when the intuitive root-finding approach fails completely, the structure of the problem ensures that a solution always exists. The so-called hard case is therefore not a breakdown of the trust-region method, but rather a reminder that careful handling of eigenvalue structure is essential in practical algorithms.

We derive the conjugate gradient method and discuss its convergence. For simplicity, we drop the qualifier "linear" throughout.

The conjugate gradient method is an iterative method for solving the linear system:

$$Ax = b,$$

where A is a $n \times n$ symmetric positive definite.

The system is equivalent to the minimization problem

$$\min \phi(x) \stackrel{\text{def}}{=} \frac{1}{2} x^T A x - b^T x,$$

which has the same unique solution.

The gradient of ϕ equals the residual of the linear system:

$$\nabla \phi(x) = Ax - b \stackrel{\text{def}}{=} r(x),$$

so at $x = x_k$ we have:

$$r_k = Ax_k - b.$$

Comments

The conjugate gradient method is one of the most important iterative algorithms for solving systems of linear equations of the form $Ax = b$, where A is a symmetric positive definite matrix of size $n \times n$. Such matrices arise frequently in optimization, differential equations, and engineering problems. Instead of attempting to solve this system directly by factorization, the conjugate gradient method reformulates the task as a minimization problem. Specifically, we define a quadratic function $\phi(x) = \frac{1}{2} x^T A x - b^T x$. This quadratic function is strictly convex because the matrix A is positive definite, and therefore it possesses a unique minimizer. The remarkable fact is that the minimizer of ϕ coincides exactly with the solution of the linear system $Ax = b$.

To advance further, we note the connection between the gradient of ϕ and the residual of the linear system. Computing the derivative gives $\nabla \phi(x) = Ax - b$, which we denote as $r(x)$. Thus, the residual vector r indicates how far our current iterate x is from solving the system. In the iterative process, at each step k, the residual is defined as $r_k = Ax_k - b$. This close relationship between optimization and linear algebra is the foundation of the conjugate gradient method. The algorithm will use properties of residuals and special search directions to reach the exact solution in at most n steps.

Definition

A set of nonzero vectors $\{p_0, p_1, \ldots, p_l\}$ is conjugate with respect to a $n \times n$ symmetric positive definite matrix A if

$$p_i^T A p_j = 0 \quad \text{for all } i \neq j.$$

Any such set is linearly independent.

Key Property: A quadratic function $\phi(x)$ can be minimized in at most n steps along conjugate directions.

Given an initial point $x_0 \in \mathbb{R}^n$ and a set of conjugate directions $\{p_0, \ldots, p_{n-1}\}$, define the iterates

$$x_{k+1} = x_k + \alpha_k p_k,$$

where the step length α_k is the one-dimensional minimizer of the quadratic function $\phi(\cdot)$ along $x_k + \alpha p_k$, given by

$$\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}, \quad r_k = Ax_k - b.$$

Comments

To design the method, we must introduce the concept of conjugate directions. A set of nonzero vectors $\{p_0, p_1, \ldots, p_l\}$ is said to be conjugate with respect to the matrix A if $p_i^T A p_j = 0$ whenever $i \neq j$. This condition generalizes the notion of orthogonality. While ordinary orthogonality uses the Euclidean inner product, conjugacy uses the inner product defined by the positive definite matrix A. Importantly, any conjugate set of vectors is automatically linearly independent, and therefore it can span the entire space when its size reaches n.

The key property is striking: if we minimize a quadratic function ϕ along conjugate directions, we obtain the exact solution in at most n steps. The procedure begins from an initial guess $x_0 \in \mathbb{R}^n$, together with a chosen sequence of conjugate directions $\{p_0, \ldots, p_{n-1}\}$. At each iteration, we update the iterate according to $x_{k+1} = x_k + \alpha_k p_k$. The step length α_k is determined by exact minimization of the quadratic function along this one-dimensional line. A straightforward calculation shows that $\alpha_k = -(r_k^T p_k)/(p_k^T A p_k)$, where $r_k = Ax_k - b$. This formula guarantees that each step makes the residual orthogonal to the chosen direction, ensuring progress toward the minimizer.

Theorem 16

For any $x_0 \in \mathbb{R}^n$, the sequence $\{x_k\}$ generated by the conjugate direction algorithm $x_{k+1} = x_k + \alpha_k p_k$, with $\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}$, $r_k = Ax_k - b$, converges to the solution x^* of the linear system $Ax = b$ in at most n steps.

Proof: Since the directions $\{p_i\}$ are linearly independent, they span $\mathbb{R}^n$. Therefore,

$$x^* - x_0 = \sigma_0 p_0 + \sigma_1 p_1 + \cdots + \sigma_{n-1} p_{n-1}$$

for some scalars σ_k. Premultiplying by $p_k^T A$ and using conjugacy yields

$$\sigma_k = \frac{p_k^T A(x^* - x_0)}{p_k^T A p_k}.$$

We now show that these coefficients σ_k equal the step sizes α_k from the algorithm.

Comments

The convergence of the conjugate direction method can now be established formally. The following theorem holds. Under our assumptions, for any initial approximation, the conjugate directions method converges to the solution in no more than n steps. To be precise, suppose we begin from any initial point x_0. Because the set of directions $\{p_0, \ldots, p_{n-1}\}$ is linearly independent, it spans the entire space $\mathbb{R}^n$. Consequently, the difference between the exact solution x^* and the initial guess x_0 can be expressed as a linear combination: $x^* - x_0 = \sigma_0 p_0 + \sigma_1 p_1 + \cdots + \sigma_{n-1} p_{n-1}$. Here, σ_k are certain scalar coefficients.

To determine these coefficients, we premultiply both sides of the equation by $p_k^T A$. By conjugacy, all terms vanish except the k-th, which yields $\sigma_k = \frac{p_k^T A(x^* - x_0)}{p_k^T A p_k}$. This expression shows that the exact displacement from the initial point toward the solution is uniquely determined by the geometry of the conjugate directions. The next step is to prove that these coefficients σ_k are precisely the step lengths α_k chosen by the algorithm. If this identity holds, then the sequence of iterates constructed by the algorithm coincides with the linear decomposition of the exact solution, and hence the method must terminate in at most n steps.

Since the algorithm generates

$$x_k = x_0 + \alpha_0 p_0 + \alpha_1 p_1 + \cdots + \alpha_{k-1} p_{k-1},$$

premultiplying by $p_k^T A$ and using conjugacy yields

$$p_k^T A(x_k - x_0) = 0.$$

Thus,

$$p_k^T A(x^* - x_0) = p_k^T A(x^* - x_k) = p_k^T (b - Ax_k) = -p_k^T r_k.$$

Comparing this with $\sigma_k = \dfrac{p_k^T A(x^* - x_0)}{p_k^T A p_k}$ and $\alpha_k = -\dfrac{r_k^T p_k}{p_k^T A p_k}$, we obtain $\sigma_k = \alpha_k$, completing the proof.

□

Comments

After k iterations, the current point has the form $x_k = x_0 + \alpha_0 p_0 + \alpha_1 p_1 + \cdots + \alpha_{k-1} p_{k-1}$. Premultiplying by $p_k^T A$ and using conjugacy shows that $p_k^T A(x_k - x_0) = 0$. This implies $p_k^T A(x^* - x_0) = p_k^T A(x^* - x_k)$. Rewriting the last term gives $p_k^T (b - Ax_k) = -p_k^T r_k$.

Comparing this identity with the earlier expression $\sigma_k = \frac{p_k^T A(x^* - x_0)}{p_k^T A p_k}$, and with the definition $\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}$, we see that $\sigma_k = \alpha_k$. Therefore, the coefficients in the linear representation of the exact solution coincide with the actual step lengths computed by the method. This equality completes the proof of convergence: the conjugate direction method always produces the exact solution in at most n iterations. This result is remarkable because it provides both an algorithm and a theoretical guarantee of finite termination.

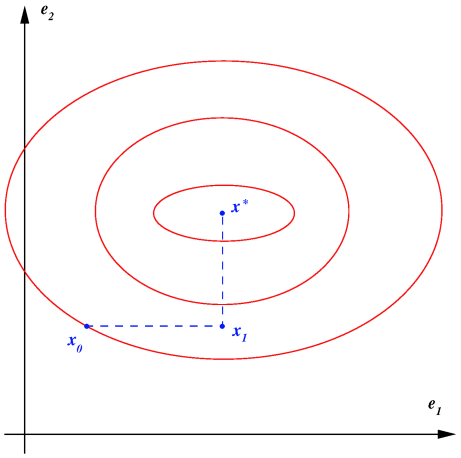

Figure: Successive minimizations along coordinate directions solve the problem in n steps.

If A is diagonal, the level sets of $\phi(\cdot)$ are axis-aligned ellipses. Minimization along coordinate directions yields the solution in n iterations.

Comments

There is a simple interpretation of the properties of conjugate directions when the matrix A is diagonal. In this situation, the quadratic function $\phi(\mathrm{x}) = \frac{1}{2}\mathrm{x}^{\mathrm{T}}\mathrm{Ax} - \mathrm{b}^{\mathrm{T}}\mathrm{x}$ has level sets that are ellipses aligned with the coordinate axes. This alignment means that if we minimize the function one coordinate at a time, we move directly toward the exact minimizer. More precisely, in each step, we perform a line minimization along a coordinate direction, such as $\mathrm{e}_1, \mathrm{e}_2, \ldots, \mathrm{e}_{\mathrm{n}}$. Each such step recovers the exact value of the corresponding component of the solution vector x^*. After n steps, the full minimizer is obtained. This property reflects the separability of the problem: since A is diagonal, there are no cross terms linking different variables, and each coordinate can be optimized independently. Thus, in the diagonal case, a naive coordinate descent method solves the problem in exactly n iterations. While this may appear trivial, it provides the motivation for constructing methods that preserve this efficiency even when A is not diagonal.

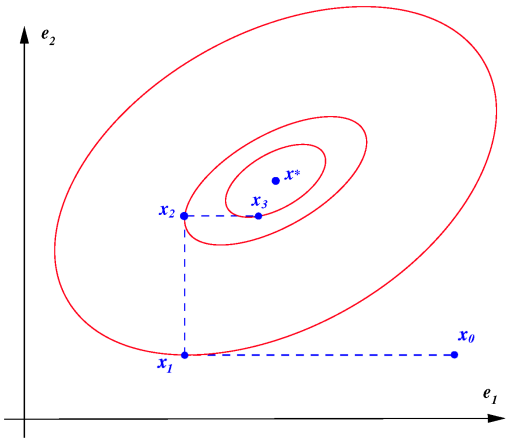

Figure: Successive minimizations along coordinate directions fail for nondiagonal Hessians.

When A is not diagonal, the level sets of $\phi(\cdot)$ are not aligned with coordinate directions. Coordinate-wise minimization does not reach the solution in n steps.

Comments

When the matrix A is not diagonal, the picture changes drastically. The quadratic function ϕ still has elliptical contours, but these ellipses are now tilted with respect to the coordinate axes. The axes of the ellipses are determined by the eigenvectors of A, not by the coordinate directions. If we insist on minimizing along coordinate directions one after another, the procedure no longer converges in n steps. In fact, the method may zigzag endlessly, never hitting the solution exactly. This happens because the coordinates are no longer independent; cross terms in the quadratic couple the variables together. As a result, optimizing one coordinate while ignoring the others disturbs the progress we have already made. Therefore, the naive coordinate descent method loses its remarkable efficiency as soon as A contains off-diagonal elements. This observation motivates us to look for new directions, ones that are not necessarily aligned with the coordinate axes but still allow us to systematically eliminate the error component by component. This is the starting point for the theory of conjugate directions.

We can diagonalize A by transforming variables using the conjugate directions:

$$\hat{x} = S^{-1}x, \quad S = [p_0\, p_1\, \dots\, p_{n-1}]$$

Then $\hat{\phi}(\hat{x}) = \phi(S\hat{x}) = \frac{1}{2}\hat{x}^T(S^TAS)\hat{x} - (S^Tb)^T\hat{x}$

- S^TAS is diagonal by A-conjugacy $\Rightarrow$ we minimize $\hat{\phi}$ via n univariate steps.
- The ith direction in $\hat{x}$-space corresponds to p_i in x-space.
- Hence, this coordinate search is equivalent to the conjugate direction algorithm.

Conclusion: Conjugate direction method solves the problem in at most n steps.

Comments

The key idea to overcome this difficulty is to construct a new set of directions, called conjugate directions, with respect to the matrix A. Suppose we gather such directions $p_0, p_1, \dots, p_{n-1}$, and arrange them as the columns of a matrix S. Then we define a change of variables by setting $\hat{x} = S^{-1}x$. In this new variable space, the quadratic function $\phi(S\hat{x})$ becomes $\hat{\phi}(\hat{x}) = \frac{1}{2}\hat{x}^T(S^TAS)\hat{x} - (S^Tb)^T\hat{x}$. By construction, the matrix S^TAS is diagonal, because the directions p_i are A-conjugate. Therefore, in the transformed space, the minimization problem reduces exactly to the simple diagonal case discussed earlier: we need only n one-dimensional minimizations along the coordinate directions of $\hat{x}$. Each coordinate direction in the transformed space corresponds to a conjugate direction in the original space. Thus, the procedure of coordinate minimization in the transformed coordinates is equivalent to moving along conjugate directions in the original coordinates. Once again, we conclude that the conjugate direction algorithm finds the exact minimizer in no more than n steps.

Note: If A is diagonal, each coordinate step recovers one component of the minimizer. After k steps, the quadratic is minimized over $\text{span}\{e_1, \ldots, e_k\}$.

We will use that the residuals satisfy the recurrence:

$$r_{k+1} = b - Ax_{k+1} = \underbrace{b - A(x_k}_{r_k} + \alpha_k p_k) = r_k + \alpha_k A p_k.$$

Theorem 17(Expanding Subspace Minimization)

Let $x_0 \in \mathbb{R}^n$ be any starting point, and suppose that the sequence $\{x_k\}$ is generated by the algorithm $x_{k+1} = x_k + \alpha_k p_k$, where $\alpha_k = -\dfrac{r_k^T p_k}{p_k^T A p_k}$.
Then the residual $r_k = b - Ax_k$ satisfies $r_k^T p_i = 0$, for $i = 0, 1, \ldots, k-1$, and x_k is the minimizer of the quadratic function

$$\phi(x) = \frac{1}{2} x^T A x - b^T x \quad \text{over the affine subspace}$$

$$\{x \mid x = x_0 + \text{span}\{p_0, p_1, \ldots, p_{k-1}\}\}.$$

Comments

There is another perspective that further clarifies the efficiency of conjugate direction methods. In the diagonal case, each coordinate step correctly recovers one component of the solution. After k steps, the quadratic has been minimized over the span of the first k coordinate directions. A similar property holds in the general case. Suppose we start from an arbitrary point x_0 and generate iterates by the rule $x_{k+1} = x_k + \alpha_k p_k$, where α_k is given by $-\frac{r_k^T p_k}{p_k^T A p_k}$. Here r_k denotes the residual, defined as $b - Ax_k$. Then we can verify a recurrence for the residuals: $r_{k+1} = r_k + \alpha_k A p_k$. Theorem 17 states that the residual r_k is orthogonal to all previous search directions, that is, $r_k^T p_i = 0$ for all $i = 0, 1, \ldots, k-1$. Moreover, the current iterate x_k is the minimizer of ϕ over the affine subspace consisting of $x_0 + \text{span}\{p_0, \ldots, p_{k-1}\}$. In other words, after k steps, we have minimized ϕ over a k-dimensional subspace generated by the search directions. Each new step expands this subspace, and after at most n steps, the entire space is covered and the exact minimizer is obtained. This expanding subspace property provides the geometric intuition behind the finite termination of conjugate direction methods.

Proof: We begin by showing that a point $\tilde{x}$ minimizes ϕ over the set

$$\{x \mid x = x_0 + \operatorname{span}\{p_0, p_1, \ldots, p_{k-1}\}\}$$

if and only if $r(\tilde{x})^T p_i = 0$ for all $i = 0, 1, \ldots, k-1$.
Let

$$h(\sigma) = \phi\left(x_0 + \sum_{i=0}^{k-1} \sigma_i p_i\right), \quad \sigma \in \mathbb{R}^k.$$

This function is a strictly convex quadratic and hence has a unique minimizer σ^* such that

$$\frac{\partial h(\sigma^*)}{\partial \sigma_i} = 0, \quad i = 0, 1, \ldots, k-1.$$

Using the chain rule, this yields

$$\nabla\phi(x_0 + \sum_{i=0}^{k-1} \sigma_i^* p_i)^T p_i = 0, \quad i = 0, 1, \ldots, k-1,$$

and hence

$$r(\tilde{x})^T p_i = 0, \quad \text{where } \tilde{x} = x_0 + \sum_{i=0}^{k-1} \sigma_i^* p_i.$$

Comments

Let us carefully examine the reasoning behind the proof. The idea is that if we want to minimize our quadratic function, denoted as ϕ, over a subspace spanned by certain search directions, the necessary and sufficient condition for optimality is that the residual at the minimizer must be orthogonal to all those search directions. In more detail, consider any point that can be expressed as the initial vector x_0 plus a linear combination of p_0 through p_{k-1}. We introduce a new function, called $h(\sigma)$, which takes the coefficients σ and evaluates ϕ at this linear combination. Since ϕ is a strictly convex quadratic, the function h is also strictly convex, which ensures the existence of a unique minimizer. The unique minimizer σ^* satisfies the condition that the partial derivative of h with respect to each σ_i equals zero. Applying the chain rule, this requirement translates into the gradient of ϕ at the minimizing point being orthogonal to every search direction p_i. By the definition of the residual, which is $\nabla\phi(x)$, this condition is exactly that $r(\tilde{x})^T p_i = 0$ for all indices i between 0 and $k-1$. This is a very important property: the residual is not arbitrary, but constrained to be orthogonal to the entire span of previous directions. Thus, the geometry of the problem links minimization along subspaces with orthogonality relations between residuals and search directions. This reasoning forms the foundation of the conjugate gradient method, because it ensures that by constructing the sequence of points in this way, we always maintain orthogonality, which is essential for convergence.

Proof of Theorem 17 (continued)

We now use induction to show that x_k satisfies

$$r_k^T p_i = 0, \quad \text{for all } i = 0, 1, \ldots, k-1.$$

For the case $k = 1$, since $x_1 = x_0 + \alpha_0 p_0$ minimizes ϕ along p_0, we have

$$r_1^T p_0 = 0.$$

Assume inductively that

$$r_{k-1}^T p_i = 0 \quad \text{for } i = 0, 1, \ldots, k-2.$$

Using $r_k = r_{k-1} + \alpha_{k-1} A p_{k-1}$,

we obtain $p_{k-1}^T r_k = p_{k-1}^T r_{k-1} + \alpha_{k-1} p_{k-1}^T A p_{k-1} = 0,$

by the definition of α_{k-1}.

For $i = 0, 1, \ldots, k-2$, we have

$$p_i^T r_k = p_i^T r_{k-1} + \alpha_{k-1} p_i^T A p_{k-1} = 0,$$

since the first term vanishes by the induction hypothesis and the second by conjugacy of $\{p_i\}$. We have shown that $r_k^T p_i = 0$, for $i = 0, 1, \ldots, k-1$, thus the theorem is proved. □

Comments

Now we extend the previous result by proving that this orthogonality is preserved at every iteration. We use mathematical induction. For the base case, when $k = 1$, the update $x_1 = x_0 + \alpha_0 p_0$ minimizes ϕ along direction p_0. This immediately implies that the residual r_1 is orthogonal to p_0. Next, assume that for some step $k - 1$, the residual r_{k-1} is orthogonal to all earlier directions p_0 through p_{k-2}. This is our induction hypothesis. Then, using the recurrence relation for residuals, namely $r_k = r_{k-1} + \alpha_{k-1} A p_{k-1}$, we analyze the orthogonality conditions. First, when we take the dot product of r_k with p_{k-1}, the formula simplifies to zero because of the definition of the step length α_{k-1}. Second, when we consider p_i with index less than $k - 1$, the inner product $p_i^T r_k$ splits into two parts: $p_i^T r_{k-1}$, which vanishes by the induction assumption, and $\alpha_{k-1} p_i^T A p_{k-1}$, which vanishes because the directions are conjugate. Therefore, r_k is orthogonal to all p_i for $i = 0, 1, \ldots, k-1$. This completes the induction and confirms that every new residual maintains orthogonality with all previous search directions. This recursive orthogonality is not just a neat property—it is the backbone of why the method works. It ensures that each step eliminates error components in new independent directions, so after a finite number of steps, the method can reach the exact solution in exact arithmetic.

- Each new direction p_k is built from only p_{k-1} and r_k.
- Conjugacy to all previous directions is automatic.
- Requires low memory and computation.

Direction update:

$$p_k = -r_k + \beta_k p_{k-1},$$

where the scalar β_k is determined by requiring $p_{k-1}^T A p_k = 0$. Premultiplying the update by $p_{k-1}^T A$ yields:

$$\beta_k = \frac{r_k^T A p_{k-1}}{p_{k-1}^T A p_{k-1}}.$$

Initial direction:

$$p_0 = -r_0.$$

Comments

We now shift from general properties to a very practical insight of the conjugate gradient method: it does not need to store or recompute all the previous directions. Instead, each new direction p_k is built from only two ingredients: the current residual r_k and the immediately preceding direction p_{k-1}. This is known as a two-term recurrence. The advantage is striking: the new direction automatically remains conjugate to all earlier directions, even though we never explicitly refer to them. This fact keeps memory requirements extremely low and computational effort minimal. To be specific, the update formula is $p_k = -r_k + \beta_k p_{k-1}$. Here, β_k is a scalar that ensures conjugacy with the previous direction. We find β_k by requiring that the inner product of $p_{k-1}^T A p_k$ equals zero. Solving this condition gives $\beta_k = \frac{r_k^T A p_{k-1}}{p_{k-1}^T A p_{k-1}}$. Notice the simplicity: all the information comes from the current residual and the last search direction. The very first direction is chosen as $-r_0$, which is the steepest descent direction at the starting point. This two-term recurrence is the key that makes the conjugate gradient method both elegant and efficient. Without it, we would have to store a growing set of vectors and perform costly orthogonalizations, but with it the method requires only minimal updates and yet guarantees the full conjugacy property.

Conjugate Gradient: Preliminary Version

Goal: Perform 1D minimizations along A-conjugate directions defined via recurrence using residuals.

Algorithm 8 (CG — Preliminary Version):

given x_0
$r_0 \leftarrow Ax_0 - b, \quad p_0 \leftarrow -r_0$
$k \leftarrow 0$
while $r_k \neq 0$ do
  $\alpha_k \leftarrow -(r_k^T p_k)/(p_k^T A p_k)$
  $x_{k+1} \leftarrow x_k + \alpha_k p_k$
  $r_{k+1} \leftarrow Ax_{k+1} - b$
  $\beta_{k+1} \leftarrow (r_{k+1}^T A p_k)/(p_k^T A p_k)$
  $p_{k+1} \leftarrow -r_{k+1} + \beta_{k+1} p_k$
  $k \leftarrow k + 1$
end while

Note: This version is useful for studying the essential properties of the conjugate gradientmethod; efficiency improvements follow in next version.

Comments

We now have all the ingredients to describe the conjugate gradient method as an explicit algorithm. The procedure starts from an initial guess x_0. We compute the initial residual $r_0 = Ax_0 - b$, and set the first search direction $p_0 = -r_0$. Then, iteration begins. At each step, we compute a step length $\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}$. This formula guarantees minimization of ϕ along the direction p_k. Next, we update the solution: $x_{k+1} = x_k + \alpha_k p_k$. We also update the residual: $r_{k+1} = Ax_{k+1} - b$. Then, to build the new search direction, we compute $\beta_{k+1} = \frac{r_{k+1}^T A p_k}{p_k^T A p_k}$, and define $p_{k+1} = -r_{k+1} + \beta_{k+1} p_k$. Finally, we increment k and repeat the process while the residual is nonzero. This structure embodies everything we have established: orthogonality of residuals, conjugacy of directions, and the use of the two-term recurrence. It is worth emphasizing that this preliminary version of the algorithm is primarily pedagogical: it reveals the essential mechanics of conjugate gradients. Later refinements make the method more efficient and numerically stable. But even at this stage, the method is already extremely powerful, since in exact arithmetic it can find the exact solution in at most n steps for an n-dimensional system.

Note:

- The directions $p_0, p_1, \ldots, p_{n-1}$ are conjugate.
- The residuals r_i are mutually orthogonal.
- Each residual r_k and direction p_k lies in the Krylov subspace of degree k for r_0, defined as. $\mathcal{K}(r_0; k) \stackrel{\text{def}}{=} \text{span}\{r_0, Ar_0, \ldots, A^k r_0\}$.

Theorem 18

Suppose that the k-th iterate generated by the conjugate gradient method is not the solution point x^*. The following four properties hold:

$$r_k^T r_i = 0, \quad \text{for } i = 0, 1, \ldots, k-1;$$
$$\text{span}\{r_0, r_1, \ldots, r_k\} = \text{span}\{r_0, Ar_0, \ldots, A^k r_0\};$$
$$\text{span}\{p_0, p_1, \ldots, p_k\} = \text{span}\{r_0, Ar_0, \ldots, A^k r_0\};$$
$$p_k^T A p_i = 0, \text{ for } i = 0, 1, \ldots, k-1.$$

Therefore, the sequence $\{x_k\}$ converges to x^* in at most n steps.

Comments

At this stage, we can summarize the central structural properties of the conjugate gradient method. The method simultaneously produces two sequences of vectors: the residuals and the search directions. The residuals, denoted by r_k, represent the current gradient information, and a remarkable fact is that these residuals are mutually orthogonal. This means that each residual points in a direction completely independent of all the previous ones. In parallel, the search directions, denoted by p_k, are conjugate with respect to the matrix A, meaning that the inner product of $p_i^T A p_j$ equals zero whenever $i \neq j$. These properties guarantee that the algorithm never revisits the same error component twice. Another fundamental idea is that both the residuals and the search directions lie in Krylov subspaces. More precisely, each search direction p_k and residual r_k belong to the Krylov subspace generated by the initial residual r_0, which is the span of $r_0, Ar_0, A^2 r_0, \ldots, A^k r_0$. This nested sequence of subspaces provides the geometric framework for the method. Because the dimension of the Krylov subspaces can increase at most by one at each iteration, and because the search directions are linearly independent, the method must terminate in at most n steps for an n-dimensional system. The orthogonality of residuals, the conjugacy of directions, and the structure of Krylov subspaces together explain the efficiency of the conjugate gradient method: it is both memory-efficient and theoretically optimal in exact arithmetic.

Proof: The proof proceeds by induction.

For $k = 0$, the relations

$$\text{span}\{r_0\} = \text{span}\{r_0\}, \quad \text{span}\{p_0\} = \text{span}\{r_0\}$$

hold trivially.

The conjugacy condition (for $k = 1$)

$$p_1^T A p_0 = -r_1^T A p_0 + \beta_1 p_0^T A p_0 = 0$$

holds by construction of $\beta_1 = \frac{r_1^T A p_0}{p_0^T A p_0}$.

Assume the following for some k:

$$r_k \in \text{span}\{r_0, Ar_0, \ldots, A^k r_0\} \text{ and } p_k \in \text{span}\{r_0, Ar_0, \ldots, A^k r_0\}$$

Then (by multiplying the second of these expressions by A) we obtain:

$$Ap_k \in \text{span}\{Ar_0, \ldots, A^{k+1} r_0\}, \quad r_{k+1} = r_k + \alpha_k A p_k \in \text{span}\{r_0, Ar_0, \ldots, A^{k+1} r_0\}$$

Hence,

$$\text{span}\{r_0, \ldots, r_{k+1}\} \subset \text{span}\{r_0, Ar_0, \ldots, A^{k+1} r_0\}$$

Comments

The proof of these properties is naturally based on induction. We begin with the base case. For $k = 0$, the span of the residual set is simply the span of r_0, and similarly, the span of the initial search direction is also the span of r_0. These relations are trivial but essential as a starting point. For $k = 1$, the conjugacy condition between p_0 and p_1 follows directly from the construction of β_1. Recall that β_1 is defined as the ratio of $r_1^T A p_0$ divided by $p_0^T A p_0$. Substituting this into the recurrence relation for the search direction p_1, we find that the inner product of $p_1^T A p_0$ equals zero, which establishes conjugacy at the first step. With this foundation, the induction hypothesis assumes that for some index k, both r_k and p_k belong to the Krylov subspace of degree k. That is, they can be written as linear combinations of $r_0, Ar_0, \ldots, A^k r_0$. Multiplying the second inclusion by the matrix A, we see that Ap_k lies in the Krylov subspace of degree $k + 1$. Using the recurrence relation for residuals, namely $r_{k+1} = r_k + \alpha_k A p_k$, we then conclude that r_{k+1} also belongs to the Krylov subspace of degree $k + 1$. This establishes the forward inclusion and prepares the ground for the reverse inclusion that completes the induction argument.

Proof of Theorem 18 (continued)

To prove the reverse inclusion

$$\text{span}\{r_0, Ar_0, \ldots, A^{k+1}r_0\} \subset \text{span}\{r_0, r_1, \ldots, r_{k+1}\}$$

we use the induction assumption to deduce that:

$$A^{k+1}r_0 = A(A^k r_0) \in \text{span}\{Ap_0, Ap_1, \ldots, Ap_k\}$$

From the update rule ($r_{k+1} = r_k + \alpha_k Ap_k$) we have

$$Ap_i = \frac{r_{i+1} - r_i}{\alpha_i}, \quad i = 0, 1, \ldots, k$$

and we conclude:

$$A^{k+1}r_0 \in \text{span}\{r_0, r_1, \ldots, r_{k+1}\}$$

By combining this expression with the induction hypothesis for second conditions we obtain

$$\text{span}\{r_0, Ar_0, \ldots, A^{k+1}r_0\} \subset \text{span}\{r_0, r_1, \ldots, r_{k+1}\}$$

and equality holds (when k is replaced by $k+1$).

Comments

The next step in the induction is to prove the reverse inclusion: that the Krylov subspace of degree $k+1$ is contained in the span of the first $k+1$ residuals. This is the crucial part because it establishes equality between the two spans. To achieve this, we use the induction assumption that $A^k r_0$ belongs to the span of $Ap_0, Ap_1, \ldots, Ap_k$. Multiplying by A, we obtain that $A^{k+1}r_0$ lies in the span of $Ap_0, Ap_1, \ldots, Ap_k$. The update formula for residuals, which is $r_{i+1} = r_i + \alpha_i Ap_i$, can be rearranged to give $Ap_i = \frac{r_{i+1} - r_i}{\alpha_i}$. This shows that each Ap_i lies in the span of residuals. Therefore, $A^{k+1}r_0$ is also a linear combination of residuals $r_0, r_1, \ldots, r_{k+1}$. By combining this result with the earlier inclusion, we establish that the span of the first $k+1$ residuals is equal to the Krylov subspace of degree $k+1$. This completes the induction step for the residuals.

Proof of Theorem 18 (continued)

To show the relation

$$\mathrm{span}\{p_0, \ldots, p_{k+1}\} = \mathrm{span}\{r_0, Ar_0, \ldots, A^{k+1}r_0\}$$

we use:

$$\begin{aligned}
&\mathrm{span}\{p_0, \ldots, p_k, p_{k+1}\} \\
&= \mathrm{span}\{p_0, \ldots, p_k, r_{k+1}\} \quad (\text{since } p_{k+1}^{\mathrm{T}} = -r_{k+1}^{\mathrm{T}} + \beta_{k+1} p_k^{\mathrm{T}}) \\
&= \mathrm{span}\{r_0, Ar_0, \ldots, A^k r_0, r_{k+1}\} \quad (\text{induction}) \\
&= \mathrm{span}\{r_0, \ldots, r_{k+1}\} \\
&= \mathrm{span}\{r_0, Ar_0, \ldots, A^{k+1} r_0\} \quad (\text{from previous step})
\end{aligned}$$

To prove conjugacy for p_{k+1} we multiply $p_{k+1}^{\mathrm{T}} = -r_{k+1}^{\mathrm{T}} + \beta_{k+1} p_k^{\mathrm{T}}$ by Ap_i:

$$p_{k+1}^{\mathrm{T}} A p_i = -r_{k+1}^{\mathrm{T}} A p_i + \beta_{k+1} p_k^{\mathrm{T}} A p_i, \quad i = 0, \ldots, k$$

When $i = k$, the right-hand side is zero by the definition of β_{k+1}.

For $i < k$, by induction hypothesis ($Ap_i = \dfrac{r_{i+1} - r_i}{\alpha_i}$) and by Theorem 17 we get:

$$r_{k+1}^{\mathrm{T}} p_i = 0, \quad i = 0, \ldots, k.$$

Comments

Once the relationship between residuals and Krylov subspaces is established, we can turn to the search directions. The goal is to show that the span of the first $k+1$ search directions equals the Krylov subspace of degree $k+1$. This follows by analyzing the recurrence relation for directions. Recall that each p_{k+1} is defined as $-r_{k+1} + \beta_{k+1} p_k$. Hence, the span of $p_0, \ldots, p_{k+1}$ is equivalent to the span of $p_0, \ldots, p_k$ together with r_{k+1}. Using the induction assumption, we know that the span of $p_0, \ldots, p_k$ equals the span of $r_0, \ldots, r_k$, which itself equals the Krylov subspace of degree k. Adding r_{k+1} extends this to the Krylov subspace of degree $k+1$. Therefore, the equality of spans holds also for the directions. The second property to establish is conjugacy. To verify that p_{k+1} is conjugate to all previous directions, we compute the inner product of $p_{k+1}^{\mathrm{T}} A p_i$ for $i \leq k$. Expanding this using the recurrence relation, we obtain two terms: $-r_{k+1}^{\mathrm{T}} A p_i + \beta_{k+1} p_k^{\mathrm{T}} A p_i$. For $i = k$, this vanishes by the definition of β_{k+1}. For $i < k$, both terms vanish by the induction hypothesis, since the residuals are orthogonal and the previous directions are already conjugate.

Proof of Theorem 18 (continued)

For $i = 0, \ldots, k-1$, the induction gives:

$$\begin{aligned} Ap_i &\in A\,\mathrm{span}\{r_0, Ar_0, \ldots, A^i r_0\} \\ &= \mathrm{span}\{Ar_0, \ldots, A^{i+1} r_0\} \subset \mathrm{span}\{r_0, Ar_0, \ldots, A^{i+1} r_0\} \\ &= \mathrm{span}\{p_0, \ldots, p_{i+1}\} \end{aligned}$$

Thus,

$$r_{k+1}^{\mathrm{T}} A p_i = 0, \quad i = 0, \ldots, k-1$$

Together with

$$p_k^{\mathrm{T}} A p_i = 0 \text{ for } i < k \text{ (because of the induction hypothesis)}$$

this implies that

$$p_{k+1}^{\mathrm{T}} A p_i = -r_{k+1}^{\mathrm{T}} A p_i + \beta_{k+1} p_k^{\mathrm{T}} A p_i = 0 \text{ for } i = 0, \ldots, k-1.$$

Hence,

$$p_{k+1}^{\mathrm{T}} A p_i = 0, \quad i = 0, \ldots, k.$$

Comments

The final step in the induction concerns verifying the remaining conjugacy conditions. For indices i strictly less than k, we must show that the inner product of $r_{k+1}^{\mathrm{T}} A p_i$ vanishes. From the induction hypothesis, we know that Ap_i belongs to the span of search directions, specifically the span of $p_0, \ldots, p_{i+1}$. Since r_{k+1} is orthogonal to all these earlier search directions, it follows that $r_{k+1}^{\mathrm{T}} A p_i = 0$. Noting that by virtue of the induction assumption $p_k^{\mathrm{T}} A p_i = 0$ for $i < k$, and combing this with the last conclusions we obtain that the new direction p_{k+1} is conjugate to every previous direction (let me remind you that for $i = k$, this vanishes by the definition of β_{k+1}). This means that the whole sequence of search directions retains pairwise A-conjugacy as the algorithm proceeds. Importantly, this guarantees that each step eliminates the error component along a new independent direction, making the method fundamentally different from simple gradient descent. While steepest descent tends to zigzag and revisit directions, the conjugate gradient method systematically constructs independent search directions, ensuring finite termination in exact arithmetic.

Proof of Theorem 18 (final)

It follows that the directions generated by the conjugate gradient method is indeed a conjugate direction. Therefore the algorithm terminates in at most n steps.
To prove the orthogonality of the gradients, observe that

$$r_k^T p_i = 0, \quad i = 0, \ldots, k-1 \text{ and any } k = 1, 2, \ldots, n\text{-}1$$

and from

$$p_i = -r_i + \beta_i p_{i-1}$$

we conclude:

$$r_i \in \text{span}\{p_i, p_{i-1}\}, \quad i = 1, \ldots, k-1$$

so

$$r_k^T r_i = 0, \quad i = 1, \ldots, k-1$$

Finally, $r_k^T r_0 = -r_k^T p_0 = 0$. □

Note: This result relies on the choice $p_0 = -r_0$; in fact, the result does not hold for other choices of p_0. Since the gradients r_k are mutually orthogonal, the term "conjugate gradient method" is somewhat misleading. It is the search directions, not the gradients, that are conjugate with respect to A.

Comments

We can now summarize the complete result. The induction proves that every new residual belongs to the growing Krylov subspaces, that every new search direction is also within these subspaces, and that the directions are pairwise conjugate with respect to the matrix A. Therefore, the conjugate gradient method generates a conjugate sequence of directions. Since the Krylov subspaces expand by at most one dimension per iteration, and since the directions remain linearly independent, the method must terminate in at most n steps in an n-dimensional space. An additional property is that the residuals themselves are mutually orthogonal. This follows because each residual lies in the span of two successive directions, specifically p_i and p_{i-1}, and because conjugacy implies orthogonality of residuals. Thus, the residuals form an orthogonal sequence while the search directions form a conjugate sequence. It is important to notice a nuance here: the orthogonality of residuals relies crucially on the initial choice $p_0 = -r_0$. If we had chosen a different initial direction, the entire orthogonality result might not hold. This explains why the name "conjugate gradient method" can be slightly misleading. The gradients themselves are not conjugate; rather, they are orthogonal. It is the search directions that are conjugate with respect to the matrix A.

Key Idea: Using Theorems 17 and 18 we can eliminate matrix-vector products in the step size formulas using orthogonality and recurrence properties.

Orthogonality Condition:

$$r_k^T p_i = 0, \quad \text{for } i = 0, 1, \ldots, k-1$$

Conjugate Direction Recurrence:

$$p_{k+1} = -r_{k+1} + \beta_{k+1} p_k$$

Updated Step Length:

$$\text{we can replace } \alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k} \text{ by } \alpha_k = \frac{r_k^T r_k}{p_k^T A p_k}$$

From $\alpha_k A p_k = r_{k+1} - r_k$ and $\beta_{k+1} = \dfrac{r_{k+1}^T A p_k}{p_k^T A p_k}$ we obtain:

$$\beta_{k+1} = \frac{r_{k+1}^T \alpha_k A p_k}{\alpha_k p_k^T A p_k} = \frac{r_{k+1}^T (r_{k+1} - r_k)}{\alpha_k p_k^T A p_k} = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$$

Comments

In practice, the conjugate gradient method can be expressed in a more economical form, which reduces computational effort while preserving its theoretical properties. The key idea is that many of the quantities in the original algorithm can be simplified by exploiting orthogonality of residuals and recurrence relations of search directions. For example, instead of writing the step length α_k as $-\frac{r_k^T p_k}{p_k^T A p_k}$, we can show that it equals $\frac{r_k^T r_k}{p_k^T A p_k}$. This avoids extra inner products involving residuals and directions. Similarly, the recurrence for β_{k+1} can be simplified. Starting from the relation that $\alpha_k A p_k = r_{k+1} - r_k$, one can eliminate the matrix-vector product and show that $\beta_{k+1} = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$. Both simplifications make the algorithm more efficient, as they require only inner products and vector updates, which are cheap compared to matrix-vector multiplications. At the same time, the recurrence for the search directions remains unchanged: each new direction p_{k+1} is equal to $-r_{k+1} + \beta_{k+1} p_k$. Altogether, these formulas provide a practical and streamlined version of conjugate gradients that avoids redundant computations.

Economical Version of the CG Method (II)

Algorithm 9 (Conjugate Gradient Method):

given x_0
$r_0 \leftarrow Ax_0 - b, \quad p_0 \leftarrow -r_0, \quad k \leftarrow 0$
while $r_k \neq 0$ do
  $\alpha_k \leftarrow \frac{r_k^T r_k}{p_k^T A p_k}$
  $x_{k+1} \leftarrow x_k + \alpha_k p_k$
  $r_{k+1} \leftarrow r_k + \alpha_k A p_k$
  $\beta_{k+1} \leftarrow \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$
  $p_{k+1} \leftarrow -r_{k+1} + \beta_{k+1} p_k$
  $k \leftarrow k + 1$
end while

Note: At each iteration, only the last two values of the vectors x_k, r_k, and p_k are needed; older values can be overwritten to save memory. The main cost is the matrix-vector product Ap_k; inner products and vector updates require only a small multiple of n operations. CG is suitable for large problems, as it does not modify the matrix and avoids fill-in. For small problems, direct methods are preferable due to better numerical stability.

Comments

This practical form of the algorithm can be summarized in a compact iterative scheme. We begin with an initial guess x_0, compute the initial residual $r_0 = Ax_0 - b$, and set the first search direction $p_0 = -r_0$. Then, at each iteration, the algorithm updates four quantities. First, the step length α_k is computed as $\frac{r_k^T r_k}{p_k^T A p_k}$. Second, the new iterate x_{k+1} is obtained as $x_k + \alpha_k p_k$. Third, the residual is updated as $r_{k+1} = r_k + \alpha_k A p_k$. Fourth, the coefficient β_{k+1} is determined as $\frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$. Finally, the new search direction is defined as $-r_{k+1} + \beta_{k+1} p_k$. The loop continues until the residual vanishes or falls below a prescribed tolerance. An important practical remark is that at any point, only the most recent values of x, r, and p need to be stored; earlier ones can be overwritten, which greatly reduces memory requirements. The dominant cost per iteration is the matrix-vector product of A and p_k, while inner products and vector operations are relatively inexpensive. This makes conjugate gradients particularly attractive for large-scale problems, where factorization methods would require excessive storage and introduce fill-in. For small systems, however, direct solvers may remain preferable, because they are numerically more robust.

Key Observation: While CG terminates in at most n iterations in exact arithmetic, it often finds the solution in many fewer iterations when A's eigenvalues have favorable distributions.

Polynomial Interpretation:

From the update $x_{k+1} = x_k + \alpha_k p_k$ and Krylov subspace property:

$$x_{k+1} = x_0 + \sum_{i=0}^{k} \alpha_i p_i = x_0 + \sum_{j=0}^{k} \gamma_j A^j r_0$$

Let's define a polynomial $P_k^*(A) = \sum_{j=0}^{k} \gamma_j A^j$ such that:

$$x_{k+1} = x_0 + P_k^*(A) r_0$$

Comments

A remarkable property of the conjugate gradient method is that, although in exact arithmetic it must converge in at most n steps for an n-dimensional system, in practice it often achieves a very accurate solution much earlier. The reason lies in the distribution of eigenvalues of the system matrix A. When the eigenvalues are clustered or have certain favorable patterns, the convergence can be dramatically faster. This phenomenon can be explained through a polynomial interpretation. Each iterate can be expressed as the initial guess x_0 plus a linear combination of powers of A applied to the initial residual. In other words, after k steps, the solution can be written as $x_0 + P_k^*(A) r_0$, where P_k^* is a polynomial of degree k in A. This viewpoint reveals that the conjugate gradient method constructs, at each iteration, the best possible polynomial transformation of the residual within the Krylov subspace. Thus, instead of simply marching forward step by step, the algorithm is implicitly filtering the spectrum of A through these polynomials, which explains why the method adapts to the eigenvalue distribution and often converges much faster than the theoretical bound suggests.

Optimality of the Conjugate Gradient Method

Key Idea
Among all methods restricted to the Krylov subspace $\mathcal{K}(r_0; k)$, Algorithm 9 minimizes the A-norm distance to the solution x^* after k steps.

A-norm: $\|z\|_A^2 = z^T A z$
Using this norm and the definition of ϕ, and the fact that x^* minimizes ϕ it is easy to show that:

$$\frac{1}{2}\|x - x^*\|_A^2 = \frac{1}{2}(x - x^*)^T A(x - x^*) = \phi(x) - \phi(x^*)$$

CG Optimality:
Algorithm 9 produces x_{k+1} minimizes $\phi(x)$ and hence $\|x - x^{*2}_A\|$ over

$$x_0 + \text{span}\{p_0, p_1, \ldots, p_k\} = x_0 + \text{span}\{r_0, Ar_0, \ldots, A^k r_0\}$$

Polynomial Characterization: P_k^* solves the following problem:

$$\min_{P_k} \|x_0 + P_k(A)r_0 - x^*\|_A$$

where P_k ranges over all polynomials of degree k.

Comments

The deeper sense of optimality in conjugate gradients is revealed when we measure error in the so-called A-norm. For a vector z, this norm is defined as $\sqrt{z^T A z}$. It arises naturally because the method minimizes the quadratic functional $\phi(x) = \frac{1}{2}x^T Ax - b^T x$. The exact solution x^* minimizes ϕ, and the difference $\phi(x) - \phi(x^*)$ equals $\frac{1}{2}\|x - x^*\|_A^2$. Therefore, minimizing ϕ is equivalent to minimizing the A-norm of the error. The key result states that after k steps, the iterate x_{k+1} is the unique vector within the affine space $x_0 + \text{span}\{r_0, Ar_0, \ldots, A^k r_0\}$ that minimizes this A-norm error. Equivalently, there exists a polynomial P_k^* of degree k such that $x_{k+1} = x_0 + P_k^*(A)r_0$, and this polynomial is chosen to minimize the A-norm distance to x^* among all degree k polynomials. This explains why conjugate gradients are not just efficient but also optimal within the Krylov subspace framework: each step delivers the best possible approximation according to the geometry induced by the matrix A.

Since

$$r_0 = Ax_0 - b = Ax_0 - Ax^* = A(x_0 - x^*)$$

we have

$$x_{k+1} - x^* = x_0 + P_k^*(A)r_0 - x^* = [I + P_k^*(A)A](x_0 - x^*) \qquad (*)$$

Let $0 < \lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$ be the eigenvalues of A, and let $v_1, v_2, \ldots, v_n$ be the corresponding orthonormal eigenvectors, so that

$$A = \sum_{i=1}^{n} \lambda_i v_i v_i^T$$

Since the eigenvectors span $\mathbb{R}^n$, we can write

$$x_0 - x^* = \sum_{i=1}^{n} \xi_i v_i \qquad (**)$$

for some coefficients ξ_i.

Comments

A central idea in the analysis of iterative methods such as Conjugate Gradient is to express the error in terms of the eigenstructure of the matrix. The error after k iterations, relative to the true solution, can be written as a transformed version of the initial error. This transformation involves a polynomial of the matrix A. The beauty of spectral decomposition is that it allows us to diagonalize the problem conceptually: instead of thinking about the full vector space at once, we decompose the initial error into contributions along the eigenvectors of A. Each eigencomponent evolves independently, scaled by the eigenvalues and the chosen polynomial. Thus, the problem of understanding error propagation reduces to studying scalar effects of polynomials on eigenvalues. If we express the initial error as a linear combination of eigenvectors, then the coefficients in this expansion describe how strongly each eigenmode is present. The subsequent iterations dampen these modes differently depending on the spectrum of A. This perspective highlights why convergence depends not only on the condition number but also on how the initial error aligns with the eigenstructure. Some components may be eliminated faster, while others persist longer, depending on the polynomial chosen by the method. This structural insight provides the foundation for explaining the optimality of the Conjugate Gradient method.

It is easy to show that any eigenvector of A is also an eigenvector of $P_k(A)$ for any polynomial P_k. For our particular matrix A and its eigenvalues λ_i and eigenvectors v_i, we have

$$P_k(A)v_i = P_k(\lambda_i)v_i, \quad i = 1, 2, \ldots, n.$$

By substituting $(**)$ into $(*)$ we have

$$x_{k+1} - x^* = \sum_{i=1}^{n} [1 + \lambda_i P_k^*(\lambda_i)]\xi_i v_i.$$

By using the fact that $\|z\|_A^2 = z^T A z = \sum_{i=1}^{n} \lambda_i (v_i^T z)^2$, we have

$$\|x_{k+1} - x^*\|_A^2 = \sum_{i=1}^{n} \lambda_i [1 + \lambda_i P_k^*(\lambda_i)]^2 \xi_i^2.$$

Since the polynomial P_k^* generated by the CG method is optimal with respect to this norm, we have

$$\|x_{k+1} - x^*\|_A^2 = \min_{P_k} \sum_{i=1}^{n} \lambda_i [1 + \lambda_i P_k(\lambda_i)]^2 \xi_i^2.$$

Comments

Once the error has been decomposed along eigenvectors, an important property emerges: polynomials of the matrix act diagonally on this decomposition. That is, applying $P_k(A)$ to an eigenvector simply scales it by $P_k(\lambda_i)$. This simplification means that the evolution of each error component depends only on the corresponding eigenvalue. In the context of the Conjugate Gradient method, this leads to an elegant interpretation: at each iteration, the algorithm chooses a polynomial that minimizes the A-norm of the error. The minimization is performed across all possible polynomials of a given degree, which makes the CG polynomial P_k^* optimal. The resulting expression shows that the error at iteration k+1 is a weighted sum of scaled eigencomponents, where the weights depend on both the initial error and the polynomial evaluated at the eigenvalues. This framework highlights that CG does not act blindly—it systematically constructs polynomials that suppress certain eigenvectors as effectively as possible. The implication is profound: convergence is not random but guided by an optimal balance over the spectrum of the matrix. This explains why CG often converges much faster than simple bounds suggest, particularly when eigenvalues are clustered.

Bounding the Error: CG Convergence Estimate

By extracting the largest of the terms $[1+\lambda_i P_k(\lambda_i)]^2$, we obtain

$$\|x_{k+1}-x^*\|_A^2 \le \min_{P_k}\max_{1\le i\le n}[1+\lambda_i P_k(\lambda_i)]^2\left(\sum_{j=1}^{n}\lambda_j\xi_j^2\right)$$

$$=\min_{P_k}\max_{1\le i\le n}[1+\lambda_i P_k(\lambda_i)]^2\,\|x_0-x^*\|_A^2,$$

where we used the fact that

$$\|x_0-x^*\|_A^2=\sum_{j=1}^{n}\lambda_j\xi_j^2.$$

The expression above allows us to quantify the convergence rate of the CG method by estimating the nonnegative scalar

$$\min_{P_k}\max_{1\le i\le n}[1+\lambda_i P_k(\lambda_i)]^2. \qquad (***)$$

In other words, we search for a polynomial P_k that minimizes this quantity. In some practical cases, we can find this polynomial explicitly and draw interesting conclusions about CG behavior.

Comments

The next step is to move from exact expressions to bounds on the error. By extracting the worst-case factor from the sum, we obtain an inequality that ties the error norm to a minimization problem over polynomials. Specifically, the convergence rate depends on minimizing the maximum value of $[1+\lambda_i P_k(\lambda_i)]^2$ across all eigenvalues. This formulation isolates a single scalar quantity that encapsulates the challenge of convergence. The question is no longer how each individual eigencomponent behaves, but rather what is the best polynomial that simultaneously controls all components. The beauty of this approach is that it connects numerical linear algebra with approximation theory: the problem of finding such polynomials is directly related to Chebyshev polynomials, which are known for minimizing the maximum deviation on an interval. This perspective allows us to derive sharp convergence bounds that depend on the spectral condition number. More importantly, it shows that convergence is governed not by the dimension of the system but by how well polynomials can approximate the inverse function $1/\lambda$ over the spectrum. This explains why CG is often so efficient even for very large systems, provided the eigenvalues are not too widely spread.

Theorem 19

If A has only r distinct eigenvalues, then the Conjugate Gradient (CG) iteration will terminate at the solution in at most r iterations.

Proof: Suppose that the eigenvalues $\lambda_1, \ldots, \lambda_n$ take on the r distinct values $\tau_1 < \tau_2 < \cdots < \tau_r$.
We define a polynomial $Q_r(\lambda)$ by

$$Q_r(\lambda) = \frac{(-1)^r}{\tau_1 \cdots \tau_r}(\lambda - \tau_1) \cdots (\lambda - \tau_r)$$

so that $Q_r(\lambda_i) = 0$ for all $i = 1, \ldots, n$, and $Q_r(0) = 1$.

It follows that $Q_r(\lambda) - 1$ has a root at $\lambda = 0$, so we define a function $\tilde{P}_{r-1}$ by

$$\tilde{P}_{r-1}(\lambda) = \frac{Q_r(\lambda) - 1}{\lambda}$$

which is a polynomial of degree $r - 1$.

Comments

One of the most striking theoretical results about the Conjugate Gradient method is its finite-termination property. If the matrix has only r distinct eigenvalues, the algorithm is guaranteed to find the exact solution in at most r steps. This is remarkable because it links algebraic structure directly to computational behavior. The proof relies on constructing a special polynomial that vanishes at all the eigenvalues of the matrix. Since the CG error after k steps involves such polynomials applied to the spectrum, having a polynomial that annihilates all eigencomponents means that the error becomes zero. The explicit construction shows that if we multiply linear factors corresponding to each distinct eigenvalue, we obtain a polynomial of degree r. Adjusting it properly, one can derive a degree-$(r - 1)$ polynomial that fits into the CG framework. This observation demonstrates that the method is not only asymptotically convergent but, in principle, exact after finitely many steps. In practice, floating-point errors prevent perfect termination, yet the result still has deep implications: it explains why clustered eigenvalues lead to rapid convergence, and why preconditioning strategies aim to reduce the number of distinct effective eigenvalues. Thus, the finite-termination theorem serves both as a theoretical cornerstone and as guidance for practical algorithm design.

From $(***)$, by setting $k = r - 1$, we obtain

$$0 \le \min_{P_{r-1}} \max_{1\le i\le n} [1 + \lambda_i P_{r-1}(\lambda_i)]^2 \le \max_{1\le i\le n} [1 + \lambda_i \tilde{P}_{r-1}(\lambda_i)]^2 = \max_{1\le i\le n} Q_r^2(\lambda_i) = 0.$$

Hence, the minimal value is zero, and substituting into the estimation for $\|x_r - x^*\|_A^2$, we find:

$$\|x_r - x^*\|_A^2 = 0 \quad \Rightarrow \quad x_r = x^*. \qquad \square$$

Building on similar reasoning, Luenberger derives the following estimate, which provides a useful characterization of the CG method's behavior.

Theorem 20 (Luenberger)

If A has eigenvalues $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$, then the CG method satisfies the estimate:

$$\|x_{k+1} - x^*\|_A^2 \le \left(\frac{\lambda_{n-k} - \lambda_1}{\lambda_{n-k} + \lambda_1}\right)^2 \|x_0 - x^*\|_A^2.$$

Proof in: David G. Luenberger, Yinyu Ye *Linear and Nonlinear Programming*, 2016 (see Theorem (Partial Conjugate Gradient Method), p. 275).

Comments

At this point, let us carefully examine how the finite-termination result is established. The essential construction is to define a polynomial that vanishes at all distinct eigenvalues of the matrix. Suppose the eigenvalues take the values $\tau_1, \tau_2, \ldots, \tau_r$. We build a polynomial $Q_r(\lambda)$ as the product of terms, each of the form $(\lambda - \tau_j)$, divided by the product of τ_j, multiplied by the factor $(-1)^r$. By design, this polynomial evaluates to zero at every eigenvalue and equals one at $\lambda = 0$. Now, subtracting one gives us a polynomial with a root at zero. Dividing by λ then produces a polynomial of degree $r - 1$, which we call $\tilde{P}_{r-1}$. This polynomial lies in the admissible set considered in the convergence analysis. The importance of this step is that it provides an explicit candidate showing that the minimization problem over all such polynomials has value zero. Therefore, when we substitute into the bound for the A norm of the error, the result vanishes. In other words, the error vector after r steps becomes exactly zero, which implies that the method produces the exact solution. Thus, the termination property follows directly from polynomial approximation arguments.

The polynomial viewpoint offers a powerful way of interpreting conjugate gradient convergence. Each iteration of the method can be seen as applying a polynomial transformation to the eigenvalues of the matrix. The residual error after k steps is essentially the initial error multiplied by a polynomial of degree k that vanishes at selected eigenvalues. This is why the annihilating polynomial plays such an important role: if such a polynomial can eliminate all eigenvalues simultaneously, then the error must vanish. From this perspective, one can also derive precise estimates of the convergence rate, even when the number of distinct eigenvalues is large. Luenberger's estimate provides a useful bound in terms of the extreme eigenvalues. The bound shows that the error after $k + 1$ steps can be reduced relative to the initial error by a factor that depends on the ratio of the largest and smallest eigenvalues among those still influencing the residual. This estimate is not only a theoretical curiosity but also a practical tool, because it demonstrates that eigenvalue clustering strongly improves performance. When most eigenvalues are located in a narrow interval, the polynomial can approximate zero much more effectively on that interval, leading to rapid error reduction. On the other hand, when eigenvalues are widely spread, the polynomial must stretch across a larger range, which weakens its effectiveness and slows convergence. Thus, the polynomial interpretation provides both a proof of finite termination and an explanation for varying rates of practical convergence.

Suppose the eigenvalues of A consist of m large values and the remaining $n-m$ eigenvalues clustered near 1. Let

$$\epsilon = \lambda_{n-m} - \lambda_1$$

Then by Theorem 20, after $m+1$ steps of CG we have:

$$\|x_{m+1} - x^*\|_A \approx \epsilon \|x_0 - x^*\|_A$$

For small ϵ, this implies that CG gives a good approximation in just $m+1$ steps.

The next figure compares two scenarios:

(i) One with 5 large eigenvalues, and the rest clustered in $[0.95, 1.05]$

(ii) One with randomly distributed eigenvalues

In both cases, we plot $\log(\|x_{m+1} - x^*\|_A^2)$ versus iteration number to visualize convergence.

Comments

Let us illustrate how the convergence estimate becomes especially insightful in cases where eigenvalues have particular structure. Suppose the spectrum of the matrix consists of a few large eigenvalues, say m of them, while the remaining $n-m$ eigenvalues are tightly clustered around one. Define $\epsilon = \lambda_{n-m} - \lambda_1$. According to the inequality of Luenberger, after $m+1$ iterations the error in the A norm is approximately ϵ times the initial error. This means that once the large eigenvalues have been handled, the method effectively suppresses the remaining clustered part extremely quickly. In practical terms, when eigenvalues form tight clusters, the conjugate gradient method can achieve a highly accurate solution in a surprisingly small number of iterations. On the contrary, when eigenvalues are widely spread and lack clustering, convergence slows down significantly. This observation explains the different shapes of convergence curves that we often see in numerical experiments. It also shows why preconditioning methods, which aim to cluster eigenvalues, can be so powerful. The structure of the spectrum thus directly dictates the practical efficiency of the method, reinforcing the deep connection between algebraic properties and computational performance.

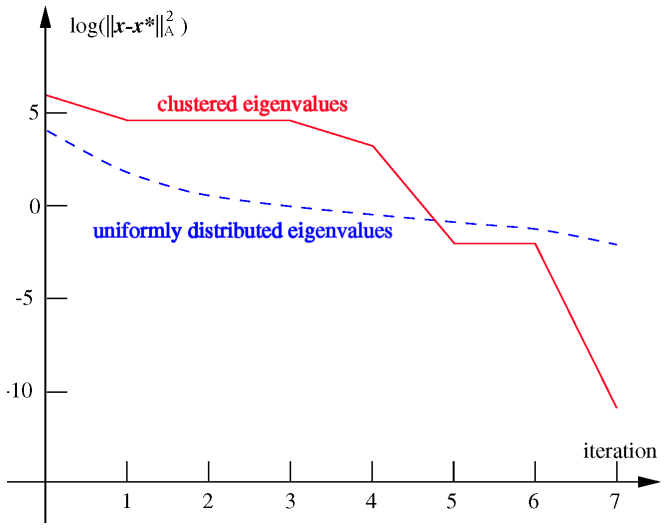

Figure: Performance of the conjugate gradient method: (i) Five large eigenvalues, remaining clustered near 1; (ii) Uniformly distributed eigenvalues.

Comments

The performance plot should be read through the lens of spectral geometry and polynomial damping. The vertical axis is "logarithm of the A-norm squared of the error", and the horizontal axis is the iteration count. A straight line with a steep negative slope corresponds to geometric decay per iteration. In the clustered case, with five outliers and the rest in a tight interval around one, the curve falls very rapidly over the first few steps: the CG polynomials quickly place near-zeros at or near the outlying eigenvalues, neutralizing those components. After this early phase, the remaining spectrum is short, and near-Chebyshev polynomials achieve uniformly small values on that short interval, so the slope remains favorable. By contrast, when eigenvalues are spread more uniformly across a wide interval, no low-degree polynomial can be simultaneously small everywhere; the best minimax factor stays comparatively large, and the slope is much gentler. You may also notice a "knee": once the dominant outliers are handled, the rate transitions to that controlled by the residual cluster width, consistent with the "epsilon" discussion. Thus, the figure is not just empirical; it visualizes exactly how polynomial annihilation and uniform approximation govern convergence. The key takeaway is that the number of iterations tracks spectral features—outliers and cluster width—rather than the dimension of the system. This is why deflation and preconditioning, which mimic the spectral clustering effect, so dramatically improve practical performance.

To improve the convergence rate of the CG method, we apply a change of variables:

$$\hat{x} = Cx,$$

where C is a nonsingular matrix. Such procedure is known as *preconditioning*. This transforms the original quadratic:

$$\hat{\phi}(\hat{x}) = \frac{1}{2}\hat{x}^T(C^{-T}AC^{-1})\hat{x} - (C^{-T}b)^T\hat{x}.$$

- Solve the equivalent system:

$$(C^{-T}AC^{-1})\hat{x} = C^{-T}b$$

  using CG on the transformed system.
- Convergence depends on the eigenvalues of $C^{-T}AC^{-1}$.
- **Goal:** Choose C such that
  - $\kappa(C^{-T}AC^{-1}) \ll \kappa(A)$, or
  - the eigenvalues of $C^{-T}AC^{-1}$ are clustered.

Comments

The discussion naturally brings us to the idea of preconditioning. We have seen that the efficiency of the conjugate gradient method depends crucially on the distribution of eigenvalues. If they are tightly clustered, convergence is rapid, whereas widely spread eigenvalues slow the method down. The purpose of preconditioning is to artificially transform the system into one with a more favorable spectrum. Concretely, we apply a change of variables by setting $\hat{x} = Cx$, where C is a nonsingular matrix. Substituting into the quadratic form shows that the new system matrix becomes $C^{-T}AC^{-1}$. The right-hand side becomes $C^{-T}b$. Solving this equivalent system with conjugate gradients is mathematically identical to solving the original one, but the convergence behavior is governed by the eigenvalues of the transformed matrix. The goal is therefore to choose the preconditioner C such that the condition number of $C^{-T}AC^{-1}$ is much smaller than that of A, or equivalently, such that its eigenvalues are well clustered. Preconditioning is thus not an optional enhancement but rather a central technique in modern iterative linear algebra. It is the main tool to bridge the gap between theoretical properties and practical efficiency when solving very large linear systems.

Preconditioned Conjugate Gradient Algorithm

Algorithm 10 (Preconditioned CG):

Given x_0, preconditioner M.
$r_0 \leftarrow Ax_0 - b$
Solve $My_0 = r_0$
$p_0 \leftarrow -y_0$, $k \leftarrow 0$
while $r_k \neq 0$ do
  $\alpha_k \leftarrow \frac{r_k^T y_k}{p_k^T A p_k}$
  $x_{k+1} \leftarrow x_k + \alpha_k p_k$
  $r_{k+1} \leftarrow r_k + \alpha_k A p_k$
  Solve $My_{k+1} = r_{k+1}$
  $\beta_{k+1} \leftarrow \frac{r_{k+1}^T y_{k+1}}{r_k^T y_k}$
  $p_{k+1} \leftarrow -y_{k+1} + \beta_{k+1} p_k$
  $k \leftarrow k + 1$
end while

Note: If $M = I$, we recover the standard CG method. Algorithm works via $M = C^T C$ (no explicit use of C). In terms of computational effort, the main difference between the preconditioned and unpreconditioned CG methods is the need to solve systems of the form $My = r$.

Comments

The preconditioned conjugate gradient method extends the classical conjugate gradient idea by inserting a preconditioner matrix that modifies the geometry of the problem. We begin with an initial guess, denoted as x_0, and define the residual $r_0 = Ax_0 - b$. Instead of working with this residual directly, we solve the auxiliary system $My_0 = r_0$, where M is the preconditioner. This produces a vector y_0 that acts as a transformed residual. The initial search direction is then chosen as $-y_0$.

At each iteration, the algorithm computes a step length $\alpha_k = \frac{r_k^T y_k}{p_k^T A p_k}$. The new approximation is updated as $x_{k+1} = x_k + \alpha_k p_k$. The residual is also updated, and we again solve a preconditioning system to obtain the new vector y_{k+1}. A correction factor β_{k+1} is formed as $\frac{r_{k+1}^T y_{k+1}}{r_k^T y_k}$. Finally, the new search direction is built as $-y_{k+1} + \beta_{k+1} p_k$.

This modification preserves the essential structure of conjugate gradients while accelerating convergence. If we set $M = I$, the procedure reduces exactly to the classical method. The only real extra cost is solving the small system involving M at each step. Thus, the efficiency of the whole method hinges on choosing M wisely: it must be easy to invert but powerful enough to cluster eigenvalues effectively.

Preconditioning Strategies

There is no universally best preconditioner: trade-offs depend on the problem.
Key considerations:

- Effectiveness of M,
- Cost of computing/storing M,
- Cost of solving $My = r$.

For structured problems (e.g., PDE discretizations), $My = r$ may correspond to a simplified or coarser version of $Ax = b$.
General-purpose preconditioners:

- SSOR, banded, incomplete Cholesky.
- Incomplete Cholesky: compute sparse $\tilde{L} \approx L$ so that

$$A \approx \tilde{L}\tilde{L}^T, \quad M = \tilde{L}\tilde{L}^T, \quad C = \tilde{L}^T$$
$$C^{-T}AC^{-1} = \tilde{L}^{-1}A\tilde{L}^{-T} \approx I$$

- Solve $My = r$ via two triangular solves with $\tilde{L}$; cost is comparable to matrix-vector product Ap.

Challenges:

- $\tilde{L}$ may not exist or be unstable $\Rightarrow$ modify diagonals or allow more fill-in (more costly).

Comments

The effectiveness of preconditioning rests on the choice of the matrix M. There is no universal best option; instead, the trade-offs depend on the problem structure. The preconditioner must satisfy three competing demands: it should substantially improve convergence, it should not be too expensive to compute or store, and solving the system $My = r$ should itself be inexpensive. In practice, one often exploits knowledge about the underlying application. For instance, in partial differential equation discretizations, the preconditioner is frequently built to mimic a coarser or simplified version of the original operator. Solving $My = r$ is then akin to solving a cheaper approximation of the full problem.

Several general-purpose strategies have also been proposed. Symmetric successive over-relaxation, banded preconditioners, and especially incomplete Cholesky factorizations are widely used. The incomplete Cholesky method works by approximating the Cholesky factorization of $A = LL^T$. Instead of computing the full factor L, we compute a sparse matrix $\tilde{L}$, chosen to remain close to L but much cheaper to store. The preconditioner is then $M = \tilde{L}\tilde{L}^T$, and one can view the transformed system as $\tilde{L}^{-1}A\tilde{L}^{-T}$, which ideally resembles the identity matrix. This means the eigenvalues of the preconditioned system are well clustered, ensuring rapid convergence.

In terms of cost, solving $My = r$ amounts to two triangular substitutions with $\tilde{L}$, which is comparable to performing one matrix-vector multiplication with A. Challenges remain, however: $\tilde{L}$ may not exist, or numerical instability may occur under strict sparsity constraints. Remedies include adjusting diagonal elements or allowing more fill-in, but these increase computational burden. Thus, the art of preconditioning is in balancing accuracy and efficiency, tailoring M to the specific problem.