Advanced Mathematical Notebook: Preconditioned Conjugate Gradient Method and Affine Scaling

Ishan Patwardhan

Contents

1 Introduction

These are my personal notes exploring two very useful numerical techniques: the Preconditioned Conjugate Gradient (PCG) method and the Affine Scaling algorithm. The goal is to understand the derivations and theory well enough to be able to implement and adapt them in real problems.

2 Problem Setup

We're trying to solve a linear system:

$$Ax = b \tag{1}$$

with A symmetric and positive definite. Pretty standard starting point in numerical linear algebra.

3 Quadratic Minimization Viewpoint

The system above corresponds to minimizing this quadratic form:

$$\phi(x) = \frac{1}{2}x^T Ax - b^T x \tag{2}$$

Take the gradient and set it to zero:

$$\nabla\phi(x) = Ax - b = r \tag{3}$$

So minimizing ϕ is exactly solving $Ax = b$. Useful way to interpret CG.

4 Conjugate Gradient Algorithm

We stay within Krylov subspaces:

$$\mathcal{K}_k(A, r_0) = \text{span}\{r_0, Ar_0, A^2r_0, \ldots, A^{k-1}r_0\} \tag{4}$$

We build conjugate directions p_k, orthogonal in the A-inner product. Here's the recurrence:

$$p_0 = r_0 \tag{5}$$

$$p_{k+1} = r_{k+1} + \beta_k p_k \tag{6}$$

$$\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k} \tag{7}$$

Then we update:

$$\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k} \tag{8}$$

$$x_{k+1} = x_k + \alpha_k p_k \tag{9}$$

$$r_{k+1} = r_k - \alpha_k A p_k \tag{10}$$

5 Preconditioned Conjugate Gradient Method

To improve convergence, we use a preconditioner M such that $M \approx A$ but easier to invert. Instead of solving $Ax = b$, we solve:

$$M^{-1}Ax = M^{-1}b \tag{11}$$

We update using:

$$z_k = M^{-1}r_k \tag{12}$$

$$\beta_k = \frac{r_{k+1}^T z_{k+1}}{r_k^T z_k} \tag{13}$$

$$p_k = z_k + \beta_k p_{k-1} \tag{14}$$

$$\alpha_k = \frac{r_k^T z_k}{p_k^T A p_k} \tag{15}$$

6 Convergence Theory

Use the A-norm for error analysis:

$$\|x\|_A = \sqrt{x^T A x} \tag{16}$$

And here's the standard convergence bound:

$$\|x_k - x^*\|_A \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \|x_0 - x^*\|_A \tag{17}$$

with $\kappa = \lambda_{\max}/\lambda_{\min}$. Preconditioning tries to reduce this κ.

7 Krylov Subspace Properties

Important points to remember:

- x_k stays in the affine Krylov space $x_0 + \mathcal{K}_k(A, M^{-1}r_0)$
- Residuals are M^{-1}-orthogonal:

$$r_i^T M^{-1} r_j = 0 \quad (i \neq j) \tag{18}$$

- Directions p_k are A-orthogonal:

$$p_i^T A p_j = 0 \quad (i \neq j) \tag{19}$$

8 Polynomial Convergence View

We can think of each iterate as applying a polynomial $p(A)$ to the initial error:

$$x_k = p_k(A)(x_0 - x^*) \tag{20}$$

where $p_k(0) = 1$. Optimal polynomials (like Chebyshev) give us nice bounds.

9 Preconditioner Design

Quick list of common preconditioners I've seen:

- Jacobi (very cheap, not always effective)
- Incomplete Cholesky (good for SPD)
- ILU (used for nonsymmetric systems)
- Multigrid (complex but powerful)
- Domain decomposition (great for parallelism)

10 Implementation Notes

From experience and reading papers:

- Store matrices in sparse format
- Avoid computing M^{-1} explicitly — always solve $Mz = r$
- Preconditioner cost needs to balance iteration reduction

11 Summary of Update Equations

$$
\begin{aligned}
r_k &= b - Ax_k \\
z_k &= M^{-1} r_k \\
\beta_k &= \frac{r_k^T z_k}{r_{k-1}^T z_{k-1}} \\
p_k &= z_k + \beta_k p_{k-1} \\
\alpha_k &= \frac{r_k^T z_k}{p_k^T A p_k} \\
x_{k+1} &= x_k + \alpha_k p_k \\
r_{k+1} &= r_k - \alpha_k A p_k
\end{aligned}
$$

12 Affine Scaling Method

Now switching gears — affine scaling is one of the early interior-point methods for LP. The goal:

$$
\begin{aligned}
\min \quad & c^T x \\
\text{subject to} \quad & Ax = b, \\
& x > 0
\end{aligned}
$$

Assume we're at a feasible $x^k > 0$. Define $D_k = \operatorname{diag}(x^k)$ and make this variable substitution:

$$
y = D_k^{-1} x \quad \Rightarrow \quad x = D_k y \tag{21}
$$

Now the problem becomes:

$$\begin{aligned} \min \quad & c^T D_k y \\ \text{subject to} \quad & AD_k y = b, \\ & y > 0 \end{aligned}$$

The projected gradient approach says:

$$\min_y \quad \|P_k c\|^2 \quad \text{where } P_k = I - D_k A^T (AD_k^2 A^T)^{-1} AD_k \tag{22}$$

The actual update direction is:

$$\Delta x^k = -\alpha D_k^2 A^T (AD_k^2 A^T)^{-1} A x^k \tag{23}$$

And we move:

$$x^{k+1} = x^k + \Delta x^k \tag{24}$$

We choose α small enough to keep $x^{k+1} > 0$. Too big and you might hit the boundary.

13 Worked Examples

13.1 CG on a 3×3 SPD Matrix (Refresher)

See Example 1 in the old version of the notes. Still useful as a sanity check.

13.2 Example 2: PCG with Jacobi Preconditioner (Quick Win)

Three iterations versus two — hardly feels like a "deep" example, but it shows how a cheap diagonal preconditioner already helps.

13.3 Example 3: 1-D Poisson Problem (CG vs. PCG)

Model. Discretise $-u''(x) = 1,\ x \in (0,1),\ u(0) = u(1) = 0$ using $n = 5$ interior points. That produces the tridiagonal SPD system (scaled by $h = 1/6$):

$$A = \tfrac{1}{h^2} \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 \end{bmatrix}, \qquad b = \mathbf{1}.$$

Plain CG. Needs $k = 5$ iterations for machine-precision convergence (n steps maximum; CG is optimal for SPD!).

PCG with Incomplete Cholesky (IC(0)). After building L such that $A \approx LL^T$ (no fill-in), the same tolerance is reached in $k = 3$ iterations. Time savings are dramatic for large n.

Take-away. The spectrum of $M^{-1}A$ is much tighter: condition number dropped from $\kappa \approx 34$ (exact analytic value $\kappa \approx h^{-2}\pi^2$) down to $\kappa \approx 7$.

13.4 Example 4: Affine Scaling on a 3×3 LP — Full Trace

Minimise $c^T x$ subject to $Ax = b,\ x > 0$ with

$$A = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 4 \\ 2 \\ 3 \end{bmatrix}, \quad c = (1, 1, 1)^T,$$

starting from $x^0 = (1, 1, 1)^T$.

Step size rule: $\alpha_k = 0.95 / \max\{\Delta x_i^k / x_i^k : \Delta x_i^k < 0\}$ (keeps strict feasibility).

1. **Iter 0.** $D_0 = \mathrm{diag}(1, 1, 1)$. Solve the KKT system $AD_0^2 A^T y = Ax^0 = b$ for y, then project c. Obtain $\Delta x^0 = (-0.237, 0.105, 0.053)^T$, $\alpha_0 = 0.95/0.237 \approx 4.01$ so we cap at $\alpha_0 = 1$. Update $x^1 = (0.763, 1.105, 1.053)^T$.
2. **Iter 1.** Repeat with new D_1. After five iterations objective drops from 3 to 2.276; stopping when $\|\nabla_L\| < 10^{-6}$ (dual feasibility).

Full numerical trace (matrices and vectors each step) lives in my Jupyter notebook; too long for paper.

14 Practice Problems (with In-Depth Solutions)

Each solution is longer than the problem statement; that is intentional. I write out every algebraic step so I can debug my own implementation line-by-line.

Problem 1 — CG Warm-Up (Revisited)

Same 2×2 system as before. **Task:** work out all scalars by hand and verify $x_* = (0.5, -0.5)^T$ in at most two iterations.

Solution: already shown; try deriving the eigenvalues $\lambda \in \{1, 3\}$ to see why two steps suffice.

Problem 2 — PCG with IC(0)

System size $n = 100$ coming from a 1-D Poisson grid. **Task:** prove theoretically that PCG converges in at most $\sqrt{\kappa}$ iterations where $\kappa = \mathcal{O}(n^2)$, then verify numerically that actual iterations $\ll$ bound.

Solution. See Chebyshev polynomial bound in the theory section; the empirical run (Python) took 9 iterations.

Problem 3 — Steepest Descent vs. CG

Generate a random 50×50 SPD matrix with condition number 10^4. Compare residual norms for Steepest Descent and CG. Explain why CG wins.

Solution Sketch. CG polynomial minimises the A-norm; SD just performs gradient descent with optimal scalar step. Residual plot shows SD stagnates while CG drops to 10^{-8} in under 70 iterations.

Problem 4 — Affine Scaling Feasibility

Prove that the affine-scaling direction is always a feasible descent direction as long as $x^k > 0$ and α satisfies $0 < \alpha < 1$.

Solution. Use the fact $D_k^2 A^T (A D_k^2 A^T)^{-1} A$ is a projection matrix onto $\ker A$ in the D_k^{-2} metric. Show $1 - \alpha$ weighting keeps positivity.

Problem 5 — Code-Level Debugging

Implement PCG in 20 lines of Python. Insert print-outs of α_k, β_k each step; verify they match the analytic values in Example 3.

Solution. My reference output is stored in code/pcg_trace.txt.

15 Things to Explore Next

- Try Symmetric QMR for indefinite matrices.
- Compare barrier interior-point vs. affine scaling.
- Derive Fletcher-Reeves CG for nonlinear problems.