PRECONDITIONING LINEAR SYSTEMS

David F. Gleich

November 25, 2019

The notes here are my own, based on Golub and van Loan, Trefethen, and Saad's textbooks, respectively.

Preconditioning is the process of taking a given linear system:

$$\boldsymbol{A}\mathbf{x} = \mathbf{b}$$

and turning it into a new linear system (with $\boldsymbol{B}$ non-singular):

$$\boldsymbol{B}\mathbf{y} = \mathbf{c}$$

such that it's "easy" to find $\mathbf{x}$ from $\mathbf{y}$ and

$$\text{an iterative method for } \boldsymbol{B}\mathbf{y} = \mathbf{c} \text{ is } \left\{ \begin{array}{c} \text{faster} \\ \text{more accurate} \\ \text{better behaved} \\ \text{convergent} \\ \text{easier, } \ldots \end{array} \right\}.$$

The standard preconditioner. The standard goal with preconditioning is to make an iterative method for $\boldsymbol{A}\mathbf{x} = \mathbf{b}$ go faster. Typically this is done by taking a non-singular matrix $\boldsymbol{M}$ and looking at the linear system: [1]

[1] So in this case $\boldsymbol{B} = \boldsymbol{MA}$, $\mathbf{x} = \mathbf{y}$ and $\mathbf{c} = \boldsymbol{M}\mathbf{b}$.

$$\boldsymbol{MA}\mathbf{x} = \boldsymbol{M}\mathbf{b}.$$

The standard idea is that $\boldsymbol{MA} \approx \boldsymbol{I}$, and we'll see how to make this idea precise shortly. *Also*, we need a *fast* way to *create* $\boldsymbol{M}$, and to *multiply* $\boldsymbol{M}$ by a vector. While this seems like an easy task, *many* preconditioners involve solving a system, hence, $\boldsymbol{M} = \boldsymbol{P}^{-1}$ for some matrix $\boldsymbol{P}$ (which could also be called a preconditioner!). Thus, just multiplying by $\boldsymbol{M}$ can be expensive itself.

Quiz Why do we need $\boldsymbol{M}$ to be non-singular?

Question 1 (The fundamental question in preconditioning) *Thus, we arise at the fundamental question. Given $\boldsymbol{A}\mathbf{x} = \mathbf{b}$, how do I pick $\boldsymbol{M}$ or $\boldsymbol{P}$ such that I actually make the iterative method faster?*

SOME THOUGHTS ON PRECONDITIONING

There is no universal preconditioner. A great open problem is to find a preconditioning strategy that works for all matrices $\boldsymbol{A}$. Recently, there has been some work on how to do this for symmetric, diagonally dominant linear systems;[2]

[2] This is the celebrated Spielman and Teng nearly-linear time solver for SDD systems. The current runtime is $O(\text{nnz}\sqrt{\log n})$ in theory, which means that it's faster to solve $\boldsymbol{A}\mathbf{x} = \mathbf{b}$ with a SDD matrix than it is to sort a vector. It's currently unknown how to extend that work to symmetric, positive definite systems, however.

Preconditioning is more art than science. As you might then expect, much of preconditioning is based on well-informed heuristic procedures. These are ideas that are theoretically grounded, but often make a *leap*. Some leaps are more effective than others!

When possible, precondition the problem, not the matrix. Suppose that our problem $\boldsymbol{A}\mathbf{x} = \mathbf{b}$ arises from a physics-based application or a complex engineered system. The problem that we want to solve gives rise to some matrix $\boldsymbol{A}$ and some right hand side $\mathbf{b}$. While we could *study* the matrix $\boldsymbol{A}$ and attempt to use a matrix-based preconditioner on $\boldsymbol{A}$, it is often a better strategy to attempt to decompose your problem as:

$\boldsymbol{A}$ = approximation with analytical solution given a right hand-side + correction.

In which case, we really have:

$$\boldsymbol{A} = \underbrace{\boldsymbol{S}}_{\text{simple}} + \underbrace{\boldsymbol{C}}_{\text{correction}}$$

and $\boldsymbol{M} = \boldsymbol{S}^{-1}$ is a good preconditioner because

$$\boldsymbol{S}^{-1}\boldsymbol{A} = \boldsymbol{I} + \boldsymbol{S}^{-1}\boldsymbol{C}.$$

1 A MORE FORMAL TREATMENT.

The following theorem justifies why S^{-1} would be a good preconditioner.

THEOREM 2 (Golub and van Loan, 3rd edition, 10.2.5) [3] *If $A = I + B$ is an n-by-n symmetric postive definite matrix and rank$(B) = r$, then Krylov methods converge in at most $r + 1$ iterations.*

[3] In the third edition, they split this into 11.3.1 and 11.3.2.

Proof This is a standard proof strategy. We show that in at most $r + 1$ iterations, the Krylov space $\mathbb{K}_{r+1}(A, \mathbf{b})$ contains the solution $\mathbf{x}$. To do so, note that:

$$\begin{aligned}\mathbb{K}_k(A, \mathbf{b}) &= \text{span}(\mathbf{b}, A\mathbf{b}, \ldots, A^{k-1}\mathbf{b}) \\ &= \text{span}(\mathbf{b}, (I + B)\mathbf{b}, (I + B)^2 \ldots, (I + B)^{k-1}\mathbf{b}) \\ &= \text{span}(\mathbf{b}, B\mathbf{b}, B^2\mathbf{b}, \ldots, B^{k-1}\mathbf{b}).\end{aligned}$$

Because B has rank r, we know that B^r has some polynomial expression in lower powers[4]; thus, the Krylov subspace terminates at this step and we know the space must contain the solution. Because of the optimality properties, any Krylov method will terminate in $r + 1$ steps in exact arithmetic. ∎

[4] This is a corrolary of the Cayley-Hamilton theorem, among other facts.

More generally speaking, we have the following theorem on the convergence of CG.

THEOREM 3 (Trefethen 38.5) *Let the CG iteration be applied to a symmetric positive definite linear system $A\mathbf{x} = \mathbf{b}$, where A has 2-norm condition number κ. Then there is a norm $\|\mathbf{z}\|_*$ where*

$$\|\mathbf{x} - \mathbf{x}_k\|_* \leq 2\|\mathbf{x} - \mathbf{x}_0\|_* \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^k.$$

This gives rise to a *linear convergence theorem* that depends on the condition number of a matrix:

$$\|\mathbf{x} - \mathbf{x}_k\|_* = O(\rho^k)$$

where ρ depends on κ.

Quiz What is $\kappa(I)$?

Suppose $\kappa(A)$ is big (like one hundred million), then what happens? We get $\rho \approx 1$ (like 0.99999999).

Suppose $\kappa(A)$ is nearly 1 (like 16), then what happens? We get $\rho \approx 0$ (like 3/5).

So given any linear system, if we take $M = A^{-1}$, we will converge in one step. But, computing $M^{-1}\mathbf{x}$ is just as expensive as our original problem. So we want something cheaper.

2 DESIGNING A PRECONDITIONER

The above theorems motivate three different types of preconditioners:

1. Find a matrix P where P^{-1} is a fast operator and $P^{-1}A \approx I$, i.e. $\kappa(MA) \ll \kappa(A)$.
2. Find a matrix P where P^{-1} is a fast operator and $P^{-1}A = I$ + low-rank.

3. Find a matrix $\boldsymbol{P}$ where $\boldsymbol{P}^{-1}$ is a fast operator and $\boldsymbol{P}^{-1}\boldsymbol{A}$ = has few eigenvalues.

In all cases we need $\boldsymbol{P}$ to be something that is easy to find as well.

Quiz Why do we get the 3rd type of preconditioner? (This is not a simple answer, but does follow from the properties of Krylov subspaces; try showing $\dim(\mathbb{K}_k(\boldsymbol{A}, \mathbf{b})) \leq 2$ when $\boldsymbol{A}$ is diagonalizable with two distinct eigenvalues.)

SOME SUBTLETIES

Suppose we want to use conjugate gradient. Then we need $\boldsymbol{A}$ to be symmetric positive and definite. Suppose we have a matrix $\boldsymbol{MA}$ where $\boldsymbol{M}$ is fast operator and easy to find. Can we always use CG? No, because

$$\boldsymbol{MA} \neq (\boldsymbol{MA})^T$$

in general.

3 TYPES OF PRECONDITIONERS

Thus, we consider four types of preconditioners:

Left	solve	$\underbrace{\boldsymbol{MA}}_{\boldsymbol{B}}\mathbf{x} = \underbrace{\boldsymbol{M}\mathbf{b}}_{\mathbf{c}}$
Right	solve	$\underbrace{\boldsymbol{AM}}_{\boldsymbol{B}}\underbrace{(\boldsymbol{M}^{-1}\mathbf{x})}_{\mathbf{y}} = \mathbf{b}$
Left &Right	solve	$\underbrace{\boldsymbol{M}_1\boldsymbol{A}\boldsymbol{M}_2}_{\boldsymbol{B}}\underbrace{(\boldsymbol{M}_2^{-1}\mathbf{x})}_{\mathbf{y}} = \underbrace{\boldsymbol{M}_1\mathbf{b}}_{\mathbf{c}}$
Symmetric	solve	$\underbrace{\boldsymbol{M}\boldsymbol{A}\boldsymbol{M}^T}_{\boldsymbol{B}}\underbrace{(\boldsymbol{M}^{-T}\mathbf{x})}_{\mathbf{y}} = \underbrace{\boldsymbol{M}\mathbf{b}}_{\mathbf{c}}$

For the CG case above, we want to use a symmetric preconditioner to preserve symmetry. Often, these are written with $\boldsymbol{C}$:

$$\underbrace{\boldsymbol{C}^{-1}\boldsymbol{A}\boldsymbol{C}^{-T}}_{\boldsymbol{B}}\mathbf{y} = \boldsymbol{C}^{-1}\mathbf{b} \qquad \mathbf{x} = \boldsymbol{C}^{-1}\mathbf{y}.$$

With the hope that $\boldsymbol{B}$ has a small condition number, or clustered eigenvalues, ...

3.1 ENSURING POSITIVE DEFINITENESS

We also need $\boldsymbol{C}^{-1}\boldsymbol{A}\boldsymbol{C}^{-T}$ to be positive definite when $\boldsymbol{A}$ is. We can insure this by taking $\boldsymbol{C}\boldsymbol{C}^T$ as the Cholesky factorization of any positive definite matrix $\boldsymbol{T}$.

3.2 OPTIMIZING CG

Once we know we are solving a preconditioned linear system, it's often advantageous to know this in the linear solver. We can rewrite CG optimally to use a preconditioner like in Golub and van Loan (4th edition) 11.5.7.

4 EXAMPLES OF PRECONDITIONERS

4.1 DIAGONALS

The simplest case of preconditioning is to use the diagonal entries. Let $\boldsymbol{A} = \boldsymbol{D} + \boldsymbol{N}$ (be a splitting into the diagonal and off-diagonal terms), then:

$$\boldsymbol{M} = \boldsymbol{D}^{-1}$$

is a preconditioner that makes

$$\boldsymbol{MA} = \boldsymbol{I} + \boldsymbol{D}^{-1}\boldsymbol{N}.$$

Quiz Is it always easy to use a diagonal precondition on a matrix?

Quiz How could you do symmetric diagonal preconditioning?

4.2 POLYNOMIALS

Recall the expansion of $\boldsymbol{A}^{-1}$ as it's Neumann series:[5]

$$(\boldsymbol{I} - \boldsymbol{A})^{-1} = \boldsymbol{I} + \boldsymbol{A} + \boldsymbol{A}^2 + \boldsymbol{A}^3 + \dots.$$

Then we can use a finite truncation as the preconditioner to $\boldsymbol{A}\mathbf{x} = \mathbf{b}$:

$$\boldsymbol{M} \approx \boldsymbol{A}^{-1} = \boldsymbol{I} + (\boldsymbol{I} - \boldsymbol{A}) + (\boldsymbol{I} - \boldsymbol{A})^2 + (\boldsymbol{I} - \boldsymbol{A})^3.$$

[5] This is a matrix based on the geometric series: $1 + t + t^2 + \dots = \frac{1}{1-t}$

4.3 INCOMPLETE FACTORIZATIONS

Incomplete Cholesky and Incomplete LU are both factorizations:

$$\boldsymbol{A} = \boldsymbol{C}\boldsymbol{C}^T - \boldsymbol{R} \qquad \boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}^T - \boldsymbol{R}$$

that are Cholesky-like and LU-like, but that have a new residual term. We call them incomplete if $\boldsymbol{R}$ has a zero-entry whenever $\boldsymbol{A}$ is non-zero. Thus, these ideas can be used for large sparse systems.

Any symmetric, positive definite matrix with a non-negative inverse (called a Stieltjes matrix) has an incomplete Cholesky factorization as worked out in Golub and van Loan.

4.4 SPARSE APPROXIMATE INVERSES

Suppose we want the best tridiagonal preconditioner for a matrix $\boldsymbol{A}$. To find this, we could consider the best approximation of the inverse:

$$\begin{aligned} &\text{minimize} && \|\boldsymbol{I} - \boldsymbol{A}\boldsymbol{M}\| \\ &\text{subject to} && \boldsymbol{M} \text{ is tridiagonal.} \end{aligned}$$

The sparsity structure should be given, so the more general problem is, given sparsity structure matrix $\boldsymbol{S}$:

$$\begin{aligned} &\text{minimize} && \|\boldsymbol{I} - \boldsymbol{A}\boldsymbol{M}\| \\ &\text{subject to} && \boldsymbol{M} \text{ has the same non-zeros as } \boldsymbol{S}. \end{aligned}$$

Consider the tridiagonal case. We can compute $\boldsymbol{M}$ a column at a time:

$$\text{Let } \boldsymbol{M}\mathbf{e}_i = \mathbf{m}_i = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \alpha \\ \beta \\ \gamma \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \text{then} \quad \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} \text{ solve } \begin{aligned} &\text{minimize} \\ & \end{aligned} \quad \left\| \mathbf{e}_i - \begin{bmatrix} \boldsymbol{A}_{i-1} & \boldsymbol{A}_i & \boldsymbol{A}_{i+1} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} \right\| .$$

4.5 MULTI-GRID

Recall how we thought about approximating the *problem* as a type of preconditioning. Suppose that $\boldsymbol{A}\mathbf{x} = \mathbf{b}$ arises from a n-by-n discretization of Poisson's equation. This gives us an $n^2 \times n^2$ linear system: $\boldsymbol{A}\mathbf{x} = \mathbf{b}$. Now, what if we had solved Poisson's equation for an $n/2$-by-$n/2$ node discretization instead? This is a continuous equation, so we might hope it's reasonable to guess that simply interpolating the solution would give us a good approximation to $\boldsymbol{A}\mathbf{x} = \mathbf{b}$? But then, we could repeat the same argument and use an $n/4$-by-$n/4$ node discretization, and so on and so forth.

This idea gives rise to a preconditioner called *multi-grid* that is incredible at solving Poisson's equations. Using a multi-grid strategy allows us to solve $\boldsymbol{A}\mathbf{x} = \mathbf{b}$ in time $O(n^2)$ where the system has size $n^2 \times n^2$. This is a linear time algorithm![6]

[6] Demmel's textbook: Applied Numerical Linear Algebra has a nice treatment of this algorithm.