Preconditioning is the process of taking a given linear system:

\[ Ax = b \]

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

\[ By = c \]

such that it's "easy" to find \( x \) from \( y \) and

an iterative method for \( By = c \) is faster, more accurate, better behaved, convergent, easier, ... \( 1 \).

The standard preconditioner. The standard goal with preconditioning is to make an iterative method for \( Ax = b \) go faster. Typically this is done by taking a non-singular matrix \( M \) and looking at the linear system:

\[ MAx = Mb. \]

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

Quiz Why do we need \( M \) to be non-singular?

Question 1 (The fundamental question in preconditioning) Thus, we arise at the fundamental question. Given \( Ax = b \), how do I pick \( M \) or \( P \) 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 \( A \). Recently, there has been some work on how to do this for symmetric, diagonally dominant linear systems;\(^2\)

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 \( Ax = b \) arises from a physics-based application or a complex engineered system. The problem that we want to solve gives rise to some matrix \( A \) and some right hand side \( b \). While we could study the matrix \( A \) and attempt to use a matrix-based preconditioner on \( A \), it is often a better strategy to attempt to decompose your problem as:

\[ A = \text{approximation with analytical solution given a right hand-side} + \text{correction.} \]

In which case, we really have:

\[ A = S + C \]

and \( M = S^{-1} \) is a good preconditioner because

\[ S^{-1}A = I + S^{-1}C. \]

\(^1\) So in this case \( B = MA, x = y \) and \( c = Mb. \)

\(^2\) 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 \( Ax = b \) 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.
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 positive definite matrix and $\text{rank}(B) = r$, then Krylov methods converge in at most $r + 1$ iterations.

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

$$
\mathbb{K}_k(A, b) = \text{span}(b, Ab, \ldots, A^{k-1}b)
= \text{span}(b, (I + B)b, (I + B)^2b, \ldots, (I + B)^{k-1}b)
= \text{span}(b, Bb, B^2b, \ldots, B^{k-1}b).
$$

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.

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 $Ax = b$, where $A$ has $2$-norm condition number $\kappa$. Then there is a norm $\|z\|_*$ where

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

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

$$
\|x - x_k\|_* = O(\rho^k)
$$

where $\rho$ depends on $\kappa$.

**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 $M^{-1}x$ 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 $P^{-1}A = I + \text{low-rank}$.
3. Find a matrix $P$ where $P^{-1}$ is a fast operator and $P^{-1}A$ has few eigenvalues.

In all cases we need $P$ 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 $\dim(\mathcal{K}_k(A, b)) \leq 2$ when $A$ is diagonalizable with two distinct eigenvalues.)

**SOME SUBLTIES**

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

$$MA \neq (MA)^T$$

in general.

### 3 TYPES OF PRECONDITIONERS

Thus, we consider four types of preconditioners:

- **Left** solve $MAx = Mb$
- **Right** solve $AMA^{-1}x = b$
- **Left & Right** solve $M_1AMA_2(M_2^{-1}x) = M_1b$
- **Symmetric** solve $MAM^T(M_2^{-1}x) = Mb$

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

$$C^{-1}AC^{-T}y = C^{-1}b \quad x = C^{-1}y.$$

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

### 3.1 ENSURING POSITIVE DEFINITENESS

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

### 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 $A = D + N$ (be a splitting into the diagonal and off-diagonal terms), then:

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

is a preconditioner that makes

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

**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 $A^{-1}$ as it's Neumann series:\textsuperscript{5}

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

Then we can use a finite truncation as the preconditioner to $Ax = b$:

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

4.3 INCOMPLETE FACTORIZATIONS

Incomplete Cholesky and Incomplete LU are both factorizations:

$$A = CC^T - R \quad A = LU^T - R$$

that are Cholesky-like and LU-like, but that have a new residual term. We call them incomplete if $R$ has a zero-entry whenever $A$ 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 $A$. To find this, we could consider the best approximation of the inverse:

$$\text{minimize } \|I - AM\|$$

subject to $M$ is tridiagonal.

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

$$\text{minimize } \|I - AM\|$$

subject to $M$ has the same non-zeros as $S$.

Consider the tridiagonal case. We can compute $M$ a column at a time:

$$\begin{bmatrix} 0 \\ \vdots \\ 0 \alpha \\ \beta \\ \gamma \\ 0 \vdots \\ 0 \end{bmatrix}$$

Let $Me_i = m_i = \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix}$ then solve $\text{minimize } \|e_i - \begin{bmatrix} A_{i-1} & A_i & A_{i+1} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix}\|$.

4.5 MULTI-GRID

Recall how we thought about approximating the problem as a type of preconditioning. Suppose that $Ax = b$ arises from a $n$-by-$n$ discretization of Poisson’s equation. This gives us an $n^2 \times n^2$ linear system: $Ax = b$. 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 $Ax = b$? 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 $Ax = b$ in time $O(n^2)$ where the system has size $n^2 \times n^2$. This is a linear time algorithm!\textsuperscript{6}

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

\textsuperscript{6} Demmel’s textbook: Applied Numerical Linear Algebra has a nice treatment of this algorithm.