Chapter 8

Overview and Preconditioned Algorithms

All of the iterative methods discussed in Part I of this book converge very rapidly if the coefficient matrix A is close to the identity. Unfortunately, in most applications, A is not close to the identity, but one might consider replacing the original linear system $Ax = b$ by the modified system

$$M^{-1}Ax = M^{-1}b \quad \text{or} \quad AM^{-1}\hat{x} = b, \quad x = M^{-1}\hat{x}. \tag{8.1}$$

These are referred to as *left* and *right* preconditioning, respectively. If M is Hermitian and positive definite, then one can precondition symmetrically and solve the modified linear system

$$L^{-1}AL^{-H}y = L^{-1}b, \quad x = L^{-H}y, \tag{8.2}$$

where $M = LL^H$. The matrix L could be the Hermitian square root of M or the lower triangular Cholesky factor of M or any other matrix satisfying $M = LL^H$. In either case, it is necessary only to be able to solve linear systems with coefficient matrix M, not to actually compute M^{-1} or L.

If the *preconditioner* M can be chosen so that

1. linear systems with coefficient matrix M are easy to solve, and
2. $M^{-1}A$ or AM^{-1} or $L^{-1}AL^{-H}$ approximates the identity,

then an efficient solution technique results from applying an iterative method to the modified linear system (8.1) or (8.2).

The exact sense in which the preconditioned matrix should approximate the identity depends on the iterative method being used. For *simple iteration*, one would like $\rho(I - M^{-1}A) << 1$ to achieve fast asymptotic convergence or $\|I - M^{-1}A\| << 1$ to achieve large error reduction at each step.

For the *CG* or *MINRES* methods for Hermitian positive definite problems, one would like the condition number of the symmetrically preconditioned matrix $L^{-1}AL^{-H}$ to be close to one, in order for the error bound based on the Chebyshev polynomial to be small. Alternatively, a preconditioned matrix with just a few large eigenvalues and the remainder tightly clustered would also be good for the CG and MINRES algorithms, as would a preconditioned matrix

with just a few distinct eigenvalues. For MINRES applied to a Hermitian indefinite linear system but with a positive definite preconditioner, it is again the eigenvalue distribution of the preconditioned matrix that is of importance. The eigenvalues should be distributed in such a way that a polynomial of moderate degree with value one at the origin can be made small at all of the eigenvalues.

For *GMRES*, a preconditioned matrix that is close to normal and whose eigenvalues are tightly clustered around some point away from the origin would be good, but other properties might also suffice to define a good preconditioner. It is less clear exactly what properties one should look for in a preconditioner for some of the other non-Hermitian matrix iterations (such as BiCG, QMR, CGS, or BiCGSTAB), but again, since each of these methods converges in one iteration if the coefficient matrix is the identity, there is the intuitive concept that the preconditioned matrix should somehow approximate the identity.

It is easy to modify the algorithms of Part I to use left preconditioning—simply replace A by $M^{-1}A$ and b by $M^{-1}b$ everywhere they appear. Right or symmetric preconditioning requires a little more thought since we want to generate approximations x_k to the solution of the original linear system, not the modified one in (8.1) or (8.2).

If the CG algorithm is applied directly to equation (8.2), then the iterates satisfy

$$\begin{aligned} y_k &= y_{k-1} + a_{k-1}\hat{p}_{k-1}, \quad a_{k-1} = \frac{\langle \hat{r}_{k-1}, \hat{r}_{k-1} \rangle}{\langle \hat{p}_{k-1}, L^{-1}AL^{-H}\hat{p}_{k-1} \rangle}, \\ \hat{r}_k &= \hat{r}_{k-1} - a_{k-1}L^{-1}AL^{-H}\hat{p}_{k-1}, \\ \hat{p}_k &= \hat{r}_k + b_{k-1}\hat{p}_{k-1}, \quad b_{k-1} = \frac{\langle \hat{r}_k, \hat{r}_k \rangle}{\langle \hat{r}_{k-1}, \hat{r}_{k-1} \rangle}. \end{aligned}$$

Defining

$$x_k \equiv L^{-H}y_k, \quad r_k \equiv L\hat{r}_k, \quad p_k \equiv L^{-H}\hat{p}_k,$$

we obtain the following preconditioned CG algorithm for $Ax = b$.

Algorithm 2P. Preconditioned Conjugate Gradient Method (PCG) (for Hermitian positive definite problems, with Hermitian positive definite preconditioners).

Given an initial guess x_0, compute $r_0 = b - Ax_0$ and solve $Mz_0 = r_0$. Set $p_0 = z_0$. For $k = 1, 2, \ldots$,

Compute Ap_{k-1}.

Set $x_k = x_{k-1} + a_{k-1}p_{k-1}$, where $a_{k-1} = \frac{\langle r_{k-1}, z_{k-1}\rangle}{\langle p_{k-1}, Ap_{k-1}\rangle}$.

Compute $r_k = r_{k-1} - a_{k-1}Ap_{k-1}$.

Solve $Mz_k = r_k$.

Set $p_k = z_k + b_{k-1}p_{k-1}$, where $b_{k-1} = \frac{\langle r_k, z_k\rangle}{\langle r_{k-1}, z_{k-1}\rangle}$.

The same modifications can be made to any of the MINRES implementations, provided that the preconditioner M is positive definite. To obtain a preconditioned version of Algorithm 4, first consider the Lanczos algorithm applied directly to the matrix $L^{-1}AL^{-H}$ with initial vector $\hat{q}_1$. Successive vectors satisfy

$$\hat{v}_j = L^{-1}AL^{-H}\hat{q}_j - \alpha_j\hat{q}_j - \beta_{j-1}\hat{q}_{j-1},$$

$$\alpha_j = \langle L^{-1}AL^{-H}\hat{q}_j, \hat{q}_j\rangle - \beta_{j-1}\langle \hat{q}_{j-1}, \hat{q}_j\rangle,$$

$$\hat{q}_{j+1} = \hat{v}_j/\beta_j, \quad \beta_j = \|\hat{v}_j\|.$$

If we define $q_j \equiv L\hat{q}_j$, $v_j \equiv L\hat{v}_j$, and $w_j \equiv M^{-1}q_j$, then the same equations can be written in terms of q_j, v_j, and w_j.

Preconditioned Lanczos Algorithm (for Hermitian matrices A, with Hermitian positive definite preconditioners M).

Given v_0, solve $M\tilde{w}_1 = v_0$, and set $\beta_0 = \langle v_0, \tilde{w}_1\rangle^{1/2}$.
Set $q_1 = v_0/\beta_0$ and $w_1 = \tilde{w}_1/\beta_0$. Define $q_0 \equiv 0$. For $j = 1, 2, \ldots$,

Set $v_j = Aw_j - \beta_{j-1}q_{j-1}$.

Compute $\alpha_j = \langle v_j, w_j\rangle$, and update $v_j \leftarrow v_j - \alpha_j q_j$.

Solve $M\tilde{w}_{j+1} = v_j$.

Set $q_{j+1} = v_j/\beta_j$ and $w_{j+1} = \tilde{w}_{j+1}/\beta_j$, where $\beta_j = \langle v_j, \tilde{w}_{j+1}\rangle^{1/2}$.

If Algorithm 4 of section 2.5 is applied directly to the preconditioned linear system (8.2) and if we let y_k and $\hat{p}_k$ denote the iterates and direction

vectors generated by that algorithm and if we then define $x_k \equiv L^{-H} y_k$ and $p_k \equiv L^{-H}\hat{p}_k$, then these vectors are generated by the following preconditioned algorithm.

Algorithm 4P. Preconditioned Minimal Residual Algorithm (PMINRES)
(for Hermitian problems, with Hermitian positive definite preconditioners).

Given x_0, compute $r_0 = b - Ax_0$ and solve $Mz_0 = r_0$.
Set $\beta = \langle r_0, z_0\rangle^{1/2}$, $q_1 = r_0/\beta$, and $w_1 = z_0/\beta$.
Initialize $\xi = (1, 0, \ldots, 0)^T$. For $k = 1, 2, \ldots$,

Compute q_{k+1}, w_{k+1}, $\alpha_k \equiv T(k,k)$, and $\beta_k \equiv T(k+1,k) \equiv T(k,k+1)$ using the preconditioned Lanczos algorithm.

Apply F_{k-2} and F_{k-1} to the last column of T; that is

$$\begin{pmatrix} T(k-2,k) \\ T(k-1,k) \end{pmatrix} \leftarrow \begin{pmatrix} c_{k-2} & s_{k-2} \\ -\bar{s}_{k-2} & c_{k-2} \end{pmatrix} \begin{pmatrix} 0 \\ T(k-1,k) \end{pmatrix}, \quad \text{if } k > 2,$$

$$\begin{pmatrix} T(k-1,k) \\ T(k,k) \end{pmatrix} \leftarrow \begin{pmatrix} c_{k-1} & s_{k-1} \\ -\bar{s}_{k-1} & c_{k-1} \end{pmatrix} \begin{pmatrix} T(k-1,k) \\ T(k,k) \end{pmatrix}, \quad \text{if } k > 1.$$

Compute the kth rotation, c_k and s_k, to annihilate the $(k+1,k)$ entry of T.[1]

Apply kth rotation to ξ and to last column of T:

$$\begin{pmatrix} \xi(k) \\ \xi(k+1) \end{pmatrix} \leftarrow \begin{pmatrix} c_k & s_k \\ -\bar{s}_k & c_k \end{pmatrix} \begin{pmatrix} \xi(k) \\ 0 \end{pmatrix}.$$

$$T(k,k) \leftarrow c_k T(k,k) + s_k T(k+1,k), \quad T(k+1,k) \leftarrow 0.$$

Compute $p_{k-1} = [w_k - T(k-1,k)p_{k-2} - T(k-2,k)p_{k-3}]/T(k,k)$, where undefined terms are zero for $k \le 2$.

Set $x_k = x_{k-1} + a_{k-1}p_{k-1}$, where $a_{k-1} = \beta\xi(k)$.

Right-preconditioned algorithms for non-Hermitian matrices are similarly derived so that the algorithms actually generate and store approximations to the solution of the original linear system.

Preconditioners can be divided roughly into three categories:

I. Preconditioners designed for general classes of matrices; e.g., matrices with nonzero diagonal entries, positive definite matrices, M-matrices. Examples of such preconditioners are the Jacobi, Gauss–Seidel, and

[1]The formula is $c_k = |T(k,k)|/\sqrt{|T(k,k)|^2 + |T(k+1,k)|^2}$, $\bar{s}_k = c_k T(k+1,k)/T(k,k)$, but a more robust implementation should be used. See, for example, BLAS routine DROTG [32].

SOR preconditioners, the incomplete Cholesky, and modified incomplete Cholesky preconditioners.

II. Preconditioners designed for broad classes of underlying problems; e.g., elliptic partial differential equations. Examples are multigrid and domain decomposition preconditioners.

III. Preconditioners designed for a specific matrix or underlying problem; e.g., the transport equation. An example is the diffusion synthetic acceleration (DSA) preconditioner, which will be mentioned but not analyzed in section 9.2.

An advantage of category I preconditioners is that they can be used in settings where the exact origin of the problem is not necessarily known—for example, in software packages for solving systems of ordinary differential equations or optimization problems. Most of the preconditioners in category I require knowledge of at least some of the entries of A. Usually, this information is readily available, but sometimes it is much easier to compute matrix–vector products, through some special formula, than it is to compute the actual entries of the matrix (in the standard basis). For example, one might use a finite difference approximation to the product of a Jacobian matrix with a given vector, without ever computing the Jacobian itself [23]. For such problems, efficient general preconditioners may be difficult to derive, and we do not address this topic here.

For practical preconditioners designed for general matrices, there are few quantitative theorems to describe just how good the preconditioner is, e.g., how much smaller the condition number of the preconditioned matrix is compared to that of the original matrix. There are, however, comparison theorems available. For large classes of problems, one may be able to prove that a preconditioner M_1 is better than another preconditioner M_2, in that, say, $\rho(I - M_1^{-1}A) < \rho(I - M_2^{-1}A)$. Such results are discussed in Chapter 10. These results may lead to theorems about the optimal preconditioner of a given form; e.g., the optimal diagonal preconditioner.

For classes of problems arising from partial differential equations, it is sometimes possible to show that a preconditioner alters the dependence of the condition number on the mesh size used in a finite difference or finite element approximation. That is, instead of considering a single matrix A and asking how much a particular preconditioner reduces the condition number of A, we consider a class of matrices A_h and preconditioners M_h parameterized by a mesh spacing h. It can sometimes be shown that while the condition number of A_h grows like $O(h^{-2})$ as $h \to 0$, the condition number of $M_h^{-1/2} A_h M_h^{-1/2}$ is only $O(h)$ or $O(1)$. This is not of much help if one's goal is to solve a specific linear system $Ax = b$, but if the goal is to solve the underlying partial differential equation, it quantifies the difficulty of solving the linear system in relation to the accuracy of the finite difference or finite element scheme. In Chapter 11, incomplete decompositions are considered, and it is shown that for

a model problem, a modified incomplete Cholesky decomposition reduces the condition number of A from $O(h^{-2})$ to $O(h^{-1})$. Multigrid methods, discussed in Chapter 12, have proved especially effective for solving problems arising from partial differential equations because they often eliminate the dependence of the condition number on h entirely.

Despite great strides in developing preconditioners for general linear systems or for broad classes of underlying problems, it is still possible in many situations to use physical intuition about a specific problem to develop a more effective preconditioner. Note that this is different from the situation with the iterative techniques themselves. Seldom (if ever) can one use physical properties of the problem being solved to devise an iteration strategy (i.e., a choice of the polynomial P_k for which $r_k = P_k(A)r_0$) that is better than, say, the CG method. For this reason, the subject of preconditioners is still a very broad one, encompassing all areas of science. No complete survey can be given. In this book, we present some known general theory about preconditioners and a few example problems to illustrate their use in practice.

Comments and Additional References.

The idea of preconditioning the CG method actually appeared in the original Hestenes and Stiefel paper [79]. It was not widely used until much later, however, after works such as [27] and [99]. See also [87], which describes some early applications. It was the development of effective preconditioning strategies that helped bring the CG algorithm into widespread use as an iterative method.