16 Preconditioning

The general idea underlying any preconditioning procedure for iterative solvers is to modify the (ill-conditioned) system

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

in such a way that we obtain an equivalent system $\hat{A}\hat{\boldsymbol{x}} = \hat{\boldsymbol{b}}$ for which the iterative method converges faster.

A standard approach is to use a nonsingular matrix M, and rewrite the system as

$$M^{-1}A\boldsymbol{x} = M^{-1}\boldsymbol{b}.$$

The *preconditioner* M needs to be chosen such that the matrix $\hat{A} = M^{-1}A$ is better conditioned for the conjugate gradient method, or has better clustered eigenvalues for the GMRES method.

16.1 Preconditioned Conjugate Gradients

We mentioned earlier that the number of iterations required for the conjugate gradient algorithm to converge is proportional to $\sqrt{\kappa(A)}$. Thus, for poorly conditioned matrices, convergence will be very slow. Thus, clearly we will want to choose M such that $\kappa(\hat{A}) < \kappa(A)$. This should result in faster convergence.

How do we find $\hat{A}$, $\hat{\boldsymbol{x}}$, and $\hat{\boldsymbol{b}}$? In order to ensure symmetry and positive definiteness of $\hat{A}$ we let

$$M^{-1} = LL^T \tag{44}$$

with a nonsingular $m \times m$ matrix L. Then we can rewrite

$$\begin{aligned} A\boldsymbol{x} = \boldsymbol{b} &\iff M^{-1}A\boldsymbol{x} = M^{-1}\boldsymbol{b} \\ &\iff L^TA\boldsymbol{x} = L^T\boldsymbol{b} \\ &\iff \underbrace{L^TAL}_{=\hat{A}}\underbrace{L^{-1}\boldsymbol{x}}_{=\hat{\boldsymbol{x}}} = \underbrace{L^T\boldsymbol{b}}_{=\hat{\boldsymbol{b}}}. \end{aligned}$$

The symmetric positive definite matrix M is called *splitting matrix* or *preconditioner*, and it can easily be verified that $\hat{A}$ is symmetric positive definite, also.

One could now formally write down the standard CG algorithm with the new "hatted" quantities. However, the algorithm is more efficient if the preconditioning is incorporated directly into the iteration. To see what this means we need to examine every single line in the CG algorithm.

We start by looking at the new residual:

$$\hat{\boldsymbol{r}}_n = \hat{\boldsymbol{b}} - \hat{A}\hat{\boldsymbol{x}}_n = L^T\boldsymbol{b} - (L^TAL)(L^{-1}\boldsymbol{x}_n) = L^T\boldsymbol{b} - L^TA\boldsymbol{x}_n = L^T\boldsymbol{r}_n,$$

and also define the following abbreviations

$$\begin{aligned} \hat{\boldsymbol{p}}_n &= L^{-1}\boldsymbol{p}_n, \\ \tilde{\boldsymbol{r}}_n &= M^{-1}\boldsymbol{r}_n. \end{aligned}$$

Now we can consider how this transforms the CG algorithm (for the hatted quantities). The initialization becomes $\hat{\boldsymbol{x}}_0 = L^{-1}\boldsymbol{x}_0 = \mathbf{0}$ and

$$\hat{\boldsymbol{r}}_0 = \hat{\boldsymbol{b}} \quad \Longleftrightarrow \quad L^T\boldsymbol{r}_0 = L^T\boldsymbol{b} \quad \Longleftrightarrow \quad \boldsymbol{r}_0 = \boldsymbol{b}.$$

The initial search direction turns out to be

$$\begin{aligned} \hat{\boldsymbol{p}}_0 = \hat{\boldsymbol{r}}_0 &\iff L^{-1}\boldsymbol{p}_0 = L^T\boldsymbol{r}_0 \\ &\iff \boldsymbol{p}_0 = M^{-1}\boldsymbol{r}_0 = \tilde{\boldsymbol{r}}_0, \end{aligned}$$

where we have used the definition of the preconditioner M. The step length $\hat{\alpha}$ transforms as follows:

$$\begin{aligned} \hat{\alpha}_n &= \left(\hat{\boldsymbol{r}}_{n-1}^T\hat{\boldsymbol{r}}_{n-1}\right) / \left(\hat{\boldsymbol{p}}_{n-1}^T\hat{A}\hat{\boldsymbol{p}}_{n-1}\right) \\ &= \left(\left(L^T\boldsymbol{r}_{n-1}\right)^T L^T\boldsymbol{r}_{n-1}\right) / \left(\left(L^{-1}\boldsymbol{p}_{n-1}\right)^T \left(L^TAL\right)\left(L^{-1}\boldsymbol{p}_{n-1}\right)\right) \\ &= \left(\boldsymbol{r}_{n-1}^T LL^T\boldsymbol{r}_{n-1}\right) / \left(\boldsymbol{p}_{n-1}^T L^{-T}L^TALL^{-1}\boldsymbol{p}_{n-1}\right) \\ &= \left(\boldsymbol{r}_{n-1}^T \underbrace{LL^T}_{=M^{-1}}\boldsymbol{r}_{n-1}\right) / \left(\boldsymbol{p}_{n-1}^T A\boldsymbol{p}_{n-1}\right) \\ &= \left(\boldsymbol{r}_{n-1}^T\tilde{\boldsymbol{r}}_{n-1}\right) / \left(\boldsymbol{p}_{n-1}^T A\boldsymbol{p}_{n-1}\right). \end{aligned}$$

The approximate solution is updated according to

$$\begin{aligned} \hat{\boldsymbol{x}}_n = \hat{\boldsymbol{x}}_{n-1} + \hat{\alpha}_n\hat{\boldsymbol{p}}_{n-1} &\iff L^{-1}\boldsymbol{x}_n = L^{-1}\boldsymbol{x}_{n-1} + \hat{\alpha}_n L^{-1}\boldsymbol{p}_{n-1} \\ &\iff \boldsymbol{x}_n = \boldsymbol{x}_{n-1} + \hat{\alpha}_n\boldsymbol{p}_{n-1}. \end{aligned}$$

The residuals are updated as

$$\begin{aligned} \hat{\boldsymbol{r}}_n = \hat{\boldsymbol{r}}_{n-1} - \hat{\alpha}_n\hat{A}\hat{\boldsymbol{p}}_{n-1} &\iff L^T\boldsymbol{r}_n = L^T\boldsymbol{r}_{n-1} - \hat{\alpha}_n(L^TAL)L^{-1}\boldsymbol{p}_{n-1} \\ &\iff \boldsymbol{r}_n = \boldsymbol{r}_{n-1} - \hat{\alpha}_n A\boldsymbol{p}_{n-1}. \end{aligned}$$

The gradient correction factor β transforms as follows:

$$\begin{aligned} \hat{\beta}_n &= \left(\hat{\boldsymbol{r}}_n^T\hat{\boldsymbol{r}}_n\right) / \left(\hat{\boldsymbol{r}}_{n-1}^T\hat{\boldsymbol{r}}_{n-1}\right) \\ &= \left(\left(L^T\boldsymbol{r}_n\right)^T\left(L^T\boldsymbol{r}_n\right)\right) / \left(\left(L^T\boldsymbol{r}_{n-1}\right)^T\left(L^T\boldsymbol{r}_{n-1}\right)\right) \\ &= \left(\boldsymbol{r}_n^T \underbrace{LL^T}_{=M^{-1}}\boldsymbol{r}_n\right) / \left(\boldsymbol{r}_{n-1}^T \underbrace{LL^T}_{=M^{-1}}\boldsymbol{r}_{n-1}\right) \\ &= \left(\boldsymbol{r}_n^T\tilde{\boldsymbol{r}}_n\right) / \left(\boldsymbol{r}_{n-1}^T\tilde{\boldsymbol{r}}_{n-1}\right). \end{aligned}$$

Finally, for the new search direction we have

$$\begin{aligned} \hat{\boldsymbol{p}}_n = \hat{\boldsymbol{r}}_n + \hat{\beta}_n\hat{\boldsymbol{p}}_{n-1} &\iff L^{-1}\boldsymbol{p}_n = L^T\boldsymbol{r}_n + \hat{\beta}_n L^{-1}\boldsymbol{p}_{n-1} \\ &\iff \boldsymbol{p}_n = M^{-1}\boldsymbol{r}_n + \hat{\beta}_n\boldsymbol{p}_{n-1} \\ &\iff \boldsymbol{p}_n = \tilde{\boldsymbol{r}}_n + \hat{\beta}_n\boldsymbol{p}_{n-1}, \end{aligned}$$

where we have multiplied by L and used the definition of M in the penultimate step.

The resulting algorithm is given by

Algorithm (Preconditioned Conjugate Gradient)

Take $\boldsymbol{x}_0 = \boldsymbol{0}$, $\boldsymbol{r}_0 = \boldsymbol{b}$

Solve $M\tilde{\boldsymbol{r}}_0 = \boldsymbol{r}_0$ for $\tilde{\boldsymbol{r}}_0$

Let $\boldsymbol{p}_0 = \tilde{\boldsymbol{r}}_0$

for $n = 1, 2, 3, \ldots$

Compute a step length

$$\alpha_n = \left(\boldsymbol{r}_{n-1}^T \tilde{\boldsymbol{r}}_{n-1}\right) / \left(\boldsymbol{p}_{n-1}^T A \boldsymbol{p}_{n-1}\right)$$

(note that $\tilde{\boldsymbol{r}}_{n-1} = M^{-1}\boldsymbol{r}_{n-1}$)

Update the approximate solution

$$\boldsymbol{x}_n = \boldsymbol{x}_{n-1} + \alpha_n \boldsymbol{p}_{n-1}$$

Update the residual

$$\boldsymbol{r}_n = \boldsymbol{r}_{n-1} - \alpha_n A \boldsymbol{p}_{n-1}$$

Solve $M\tilde{\boldsymbol{r}}_n = \boldsymbol{r}_n$ for $\tilde{\boldsymbol{r}}_n$

Compute a gradient correction factor

$$\beta_n = \left(\boldsymbol{r}_n^T \tilde{\boldsymbol{r}}_n\right) / \left(\boldsymbol{r}_{n-1}^T \tilde{\boldsymbol{r}}_{n-1}\right)$$

(note that $\tilde{\boldsymbol{r}}_{n-1} = M^{-1}\boldsymbol{r}_{n-1}$ and $\tilde{\boldsymbol{r}}_n = M^{-1}\boldsymbol{r}_n$)

Set the new search direction

$$\boldsymbol{p}_n = \tilde{\boldsymbol{r}}_n + \beta_n \boldsymbol{p}_{n-1}$$

(where $\tilde{\boldsymbol{r}}_n = M^{-1}\boldsymbol{r}_n$)

end

Remark This algorithm requires the additional work that is needed to solve the linear system $M\tilde{\boldsymbol{r}}_n = \boldsymbol{r}_n$ once per iteration. Therefore we will want to choose M so that this can be done easily and efficiently.

The two extreme cases $M = I$ and $M = A$ are of no interest. $M = I$ gives us the ordinary CG algorithm, whereas $M = A$ (with $L = A^{-1/2}$) leads to the trivial preconditioned system $\hat{A}\hat{\boldsymbol{x}} = \hat{\boldsymbol{b}} \Longleftrightarrow \hat{\boldsymbol{x}} = \hat{\boldsymbol{b}}$ since (using the symmetry of A)

$$\hat{A} = L^T A L = A^{-T/2} A^{T/2} A^{1/2} A^{-1/2} = I.$$

This may seem useful at first, but to get the solution $\boldsymbol{x}$ we need

$$\begin{aligned} \boldsymbol{x} &= L\hat{\boldsymbol{x}} = A^{-1/2}\hat{\boldsymbol{x}} = A^{-1/2}\hat{\boldsymbol{b}} \\ &= A^{-1/2} L^T \boldsymbol{b} = A^{-1/2} A^{-T/2} \boldsymbol{b} = A^{-1}\boldsymbol{b}, \end{aligned}$$

which is just as complicated as the original problem.

Therefore, M should be chosen somewhere "in between". Moreover, we want

1. M should be symmetric and positive definite.

2. M should be such that $M\tilde{\boldsymbol{r}}_n = \boldsymbol{r}_n$ can be solved efficiently.

3. M should approximate A^{-1} in the sense that $\|I - M^{-1}A\| < 1$.

If we use the decomposition $A = L + D + L^T$ of the symmetric positive definite matrix A then some possible choices for M are given by

$M = D$**:** Jacobi preconditioning,

$M = L + D$**:** Gauss-Seidel preconditioning,

$M = \frac{1}{\omega}(D + \omega L)$**:** SOR preconditioning.

Another popular preconditioner is $M = HH^T$, where H is "close" to L. This method is referred to as *incomplete Cholesky factorization* (see the book by Golub and van Loan for more details).

Remark The Matlab script `PCGDemo.m` illustrates the convergence behavior of the preconditioned conjugate gradient algorithm. The matrix A here is a 1000×1000 symmetric positive definite matrix with all zeros except $a_{ii} = 0.5 + \sqrt{i}$ on the diagonal, $a_{ij} = 1$ on the sub- and superdiagonal, and $a_{ij} = 1$ on the 100th sub- and superdiagonals, i.e., for $|i - j| = 100$. The right-hand side vector is $b = [1, \ldots, 1]^T$. We observe that the basic CG algorithm converges very slowly, whereas the Jacobi-preconditioned method converges much faster.