Preconditionning for Conjugate gradients

① **Idea**: Solve the linear system

$$M^{-1}Ax = M^{-1}b \qquad (M \text{ non-singular})$$

instead of $Ax = b$ where

$M^{-1}A$ is much better conditionned than A. M is the preconditionner

Silly question: why use the notation $M^{-1}A$ instead of MA?

Answer: If $M = A$ then $M^{-1}A = I$
→ this is the "ultimate" preconditionner, one that gives the answer immediately.

The idea is that $M^{-1}A$ should be as close as possible to the unit matrix, and so M should be as close as possible to the matrix A.

② A "minor" problem: Nothing guarantees that $M^{-1}A$ is a positive-defin. symmetric matrix so we cannot use the CG method on $M^{-1}Ax = M^{-1}b$.

Idea: Consider $M = CC^T$ (which defines C) then

$$M^{-1} = C^{-T}C^{-1}$$

and so $M^{-1}Ax = M^{-1}b$

$$\Rightarrow \quad C^{-T}C^{-1}Ax = M^{-1}b$$

let $x = C^{-T}\tilde{x}$

$\tilde{A} = C^{-1}AC^{-T}$ then

$$C^{-T}C^{-1}AC^{-T}\tilde{x} = M^{-1}b$$

$$\Rightarrow \quad \tilde{A}\tilde{x} = C^T M^{-1} b = \tilde{b}$$
$$= C^T C^{-T} C^{-1} b = C^{-1} b$$

The claim is that

$\tilde{A}$ is symmetric
$\tilde{A}$ is positive definite

and $\tilde{A}\tilde{x} = \tilde{b}$ is the problem we solve to find $\tilde{x}$, then find x using
$x = C^{-T}\tilde{x}$.

① $\tilde{A}$ is symmetric because

$$\tilde{A}^T = (C^{-1} A C^{-T})^T = C^{-1} A^T C^{-T}$$
$$= C^{-1} A C^{-T} = \tilde{A}$$

② $\tilde{A}$ is positive definite

$$x^T \tilde{A} x = x^T C^{-1} A C^{-T} x$$
$$= (C^{-T}x)^T A (C^{-T}x) \geq 0$$

since A is pos. definite

So we could
- decide what M should be
- calculate $M = CC^T$
- evaluate $\tilde{A} = C^{-1} A C^{-T}$
- evaluate $\tilde{b} = C^{-1} b$
- Perform the CG method on $\tilde{A}\tilde{x} = \tilde{b}$
- Calculate $x = C^{-T}\tilde{x}$

But this is far too long!

There's a much quicker way which doesn't require the explicit factorization of M into CC^T

③ The preconditioned CG method

The algorithm in ~ variables looks like

$$\tilde r = \tilde b$$
$$\tilde p = \tilde r - A\tilde x_0$$
$$\tilde e = \tilde r^T \tilde r$$

do $k = 1, n$

$$\tilde y = \tilde A \tilde p$$
$$\alpha = \frac{\tilde e}{\tilde p^T \tilde y}$$
$$\tilde x = \tilde x + \tilde\alpha \tilde p$$
$$\tilde r = \tilde r - \tilde\alpha \tilde y$$
$$\tilde e' = \tilde r^T \tilde r$$
$$\tilde\beta = \frac{\tilde e'}{\tilde e}$$
$$\tilde p = \tilde r + \tilde\beta \tilde p$$
$$\tilde e = \tilde e'$$

enddo

① We want to use the fact that

$$\tilde x = C^T x$$

$\Rightarrow$ $\tilde x = \tilde x + \tilde\alpha \tilde p$

becomes

$$C^T x = C^T x + \tilde\alpha \tilde p$$

$\Rightarrow$ <u>define</u> $\tilde p = C^T p$

so $x = x + \tilde\alpha p$

② in $\tilde y = \tilde A \tilde p$ then

$$\tilde y = C^{-1} A C^{-T} C^T p$$
$$= C^{-1} A p$$

so <u>define</u> $\tilde y = C^{-1} y$ so

$$y = Ap$$

③ $\tilde p^T \tilde y = p^T C C^{-1} y = p^T y.$

④ $\tilde r = \tilde r - \tilde\alpha \tilde y \Rightarrow \tilde r = \tilde r - \tilde\alpha C^{-1} y$

so <u>define</u> $\tilde r = C^{-1} r$ so

$$r = r - \tilde\alpha y$$

⑤ $\tilde e = \tilde r^T \tilde r = r^T C^{-T} C^{-1} r$

$$= r^T (C^{-T} C^{-1})^T r$$

define $\quad z = C^{-1} C^{-T} r$

$\quad\quad = M^{-1} r$

or more precisely, define z as the solution of $Mz = r$

Then

(6) $\tilde{p} = \tilde{r} + \tilde{\beta}\tilde{p} \Rightarrow C^T p = C^{-1} r + \tilde{\beta} C^T p$

$$p = C^{-T} C^{-1} r + \tilde{\beta} p$$
$$= z + \tilde{\beta} p$$

And finally, we can construct the modified, preconditioned algorithm

$\tilde{r} = \tilde{b} - \tilde{A}\tilde{x}_0$

$C^{-1} r = C^{-1} b - C^{-1} A C^{-T} C^T x_0$

$r = b - A x_0$

$r = b - A x_0$
Solve $Mz = r$
$p = z$
$e = z^T r$
do $k = 1, n$
 $y = Ap$
 $\alpha = \dfrac{e}{p^T y}$
 $x = x + \alpha p$
 $r = r - \alpha y$
 Solve $Mz = r$
 $e' = z^T r$
 $\beta = e'/e$
 $p = z + \beta p$
 $e' = e$
enddo.

If solving $Mz = r$ is cheap, then this doesn't increase CPU time too much.

Ideas for preconditioners

① $M = \text{diag}(A)$

M = a diagonal matrix with the diagonal elements of A.

② Incomplete Cholesky factorization (for sparse A)

- If we could write $A = LL^T$ exactly then the problem would be solved. However, L is not sparse even if A is so this calculation is not feasible for v. large A.

- Idea: ① perform an "incomplete factorization" calculating the elements of L only on the positions (i,j) where $a_{ij} \neq 0$

  (if A is tridiagonal, calculate only L_{ii} and $L_{i\,i+1}$) and set all the others to 0.

  ② if it happens that one of the L_{ii} thus defined is negative, set it to an arbitrary positive # (for a complete Cholesky factorization of a real, pos. def. matrix this never happens, however, it can happen for an incomplete one).

  ③ Use the sparsity of A and L in the CG process!