02.03 systems of equations: symmetric postive-definite

why stop at one? part three.

if a matrix is symmetric (and positive-definite), some wise guys decided to use half the memory.

1 symmetric positive-definite matrices

definition 12. $n \times n$ matrix A is **symmetric** if $A^T = A$. matrix A is **positive-definite** if $x^T Ax > 0$ for all vectors $x \neq 0$.

example 26

show matrix $A = \begin{bmatrix} 2 & 2 \\ 2 & 5 \end{bmatrix}$ is symmetric positive-definite.

clearly A is symmetric. to show positive-definite,

$$x^T Ax = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 2 & 2 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\begin{aligned} &= 2x_1^2 + 4x_1x_2 + 5x_2^2 \\ &= 2(x_1 + x_2)^2 + 3x_2^2. \end{aligned}$$

this expression is always non-negative and cannot be zero unless both $x_2 = 0, x_1 + x_2 = 0$, which implies $x = 0$.

example 27

show matrix $A = \begin{bmatrix} 2 & 4 \\ 4 & 5 \end{bmatrix}$ is not symmetric positive-definite.

compute $x^T Ax$,

$$x^T Ax = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 2 & 4 \\ 4 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\begin{aligned} &= 2x_1^2 + 8x_1x_2 + 5x_2^2 \\ &= 2(x_1 + 4x_1x_2)^2 + 5x_2^2 \\ &= 2(x_1 + 2x_2)^2 - 8x_2^2 + 5x_2^2 \\ &= 2(x_1 + 2x_2)^2 - 3x_2^2 \end{aligned}$$

if $x_1 = -2, x_2 = 1$, then the first term is zero and both terms together are less than zero.

property 01

if $n \times n$ matrix A is symmetric, then A is positive-definite iif all its eigenvalues are positive.

proof

magic supporting theorem: assume that A is a symmetric $m \times m$ matrix with real entries. then the eigenvalues are real numbers, and the set of unit eigenvectors of A is an orthonormal set $\{w_1, \ldots, w_m\}$ that forms a basis of $\mathcal{R}^m$.

the set of unit eigenvectors is orthonormal and spans $\mathcal{R}^n$. if A is postive-definite and $Av = \lambda v$ for nonzero vector v, then $0 < v^T Av = v^T(\lambda v) = \lambda ||v||_2^2$, so $\lambda > 0$. on the other hand, if all eigenvalues of A are positive, then write any nonzero $x = c_1v_1 + \cdots + c_nv_n$ where the v_i are orthonormal unit vectors and not all c_i are zero. then $x^T Ax = (c_1v_1 + \cdots + c_nv_n)^T(\lambda_1c_1v_1 + \cdots + \lambda_nc_nv_n) = \lambda_1c_1^2 + \cdots + \lambda_nc_n^2 > 0$, so A is positive-definite. ■

property 02

if A is $n \times n$ symmetric positive-definite and X is $n \times m$ matrix of full rank with $n \geq m$, then $X^T AX$ is $m \times m$ symmetric positive-definite.

proof

The matrix is symmetric since $(X^TAX)^T = X^TAX$. To prove positive-definite, consider a nonzero m-vector v. Note that $v^T(X^TAX)v = (Xv)^TA(Xv) \geq 0$, with equality only if $Xv = 0$, due to the positive-definiteness of A. Since X has full rank, its columns are linearly independent, so that $Xv = 0$ implies $v = 0$.

definition 13

principal submatrix of square matrix A is a square submatrix whose diagonal entries are diagonal entries of A.

property 03

any principal submatrix of a symmetric positive-definite matrix is symmetric positive-definite.

2 cholesky factorization

consider tiny symmetric positive-definite matrix

$$\begin{bmatrix} a & b \\ b & c \end{bmatrix}.$$

by property 03, $a > 0$. determinant $ac - b^2$ is positive bc its product of eigenvalues, all positive by property 01. $A = R^TR$ implies

$$\underbrace{\begin{bmatrix} a & b \\ b & c \end{bmatrix}}_{\text{compare this}} = \begin{bmatrix} \sqrt{a} & 0 \\ u & v \end{bmatrix}\begin{bmatrix} \sqrt{a} & u \\ 0 & v \end{bmatrix} = \underbrace{\begin{bmatrix} a & u\sqrt{a} \\ u\sqrt{a} & u^2+v^2 \end{bmatrix}}_{\text{to this}}$$

$$\begin{aligned} \Rightarrow u &= \frac{b}{\sqrt{a}} \\ v^2 &= c - u^2 \\ &= c - \left(\frac{b}{\sqrt{a}}\right)^2 = c - \frac{b^2}{a} > 0. \end{aligned}$$

ie, v can be defined as real number and cholesky factorization exists

$$A = \begin{bmatrix} a & b \\ b & c \end{bmatrix} = \begin{bmatrix} \sqrt{a} & 0 \\ \frac{b}{\sqrt{a}} & \sqrt{c - \frac{b^2}{a}} \end{bmatrix}\begin{bmatrix} \sqrt{a} & \frac{b}{\sqrt{a}} \\ 0 & \sqrt{c - \frac{b^2}{a}} \end{bmatrix} = R^TR.$$

theorem 14 cholesky factorization

if A is symmetric positive-definite $n \times n$ matrix, then there exists an upper triangular $n \times m$ matrix R such that $A = R^TR$.

proof

construct R by induction on size n. (case $n = 2$ previously.) consider A partitioned as

$$R = \begin{bmatrix} a & | & & b^T & \\ -- & - & -- & -- & -- \\ & | & & & \\ b & | & & C & \\ & | & & & \end{bmatrix}$$

where b is $(n-1)$-vector and C is $(n-1)\times(n-1)$ submatrix. use block multiplication to simplify. set $u = \frac{b}{\sqrt{a}}$ as in 2×2 case. set $A_1 = C - uu^T$ and defining invertible matrix

$$S = \left[\begin{array}{c|c} \sqrt{a} & u^T \\ \hline 0 & \\ \vdots & I \\ 0 & \end{array}\right]$$

yields

$$S^T \left[\begin{array}{c|ccc} 1 & 0 & \dots & 0 \\ \hline 0 & & & \\ \vdots & & A_1 & \\ 0 & & & \end{array}\right] S = \left[\begin{array}{c|ccc} \sqrt{a} & 0 & \dots & 0 \\ \hline & & & \\ u & & I & \\ & & & \end{array}\right] \left[\begin{array}{c|ccc} 1 & 0 & \dots & 0 \\ \hline 0 & & & \\ \vdots & & A_1 & \\ 0 & & & \end{array}\right] \left[\begin{array}{c|ccc} \sqrt{a} & & u^T & \\ \hline 0 & & & \\ \vdots & & I & \\ 0 & & & \end{array}\right]$$

$$= \left[\begin{array}{c|ccc} a & & b^T & \\ \hline & & & \\ b & uu^T & + & A \\ & & & \end{array}\right] = A.$$

notice that A_1 is symmetric positive-definite. bc

$$\left[\begin{array}{c|ccc} 1 & 0 & \vdots & 0 \\ \hline 0 & & & \\ \vdots & & A_1 & \\ 0 & & & \end{array}\right] = (S^T)^{-1} A S^{-1}$$

is symmetric positive-definite by property 02. therefore so is $(n-1) \times (n-1)$ principal submatrix A_1 by property 03. by induction, $A_1 = V^T V$ where V is upper triangular. then define upper triangular matrix

$$R = \left[\begin{array}{c|ccc} \sqrt{a} & & u^T & \\ \hline 0 & & & \\ \vdots & & V & \\ 0 & & & \end{array}\right]$$

and verify that

$$R^T R = \left[\begin{array}{c|ccc} \sqrt{a} & 0 & \dots & 0 \\ \hline & & & \\ u & & V^T & \\ & & & \end{array}\right] \left[\begin{array}{c|ccc} \sqrt{a} & & u^T & \\ \hline 0 & & & \\ \vdots & & V & \\ 0 & & & \end{array}\right] = \left[\begin{array}{c|ccc} a & & b^T & \\ \hline & & & \\ b & uu^T & + & V^T V \\ & & & \end{array}\right]$$
$$= A. \checkmark$$

the construction of this proof is the algorithm.

algorithm

cholesky factorization

```
for k = 1:n
  if A[k,k] < 0, stop, end
  R[k,k] = √A[k,k]
  T(u) = A[k,k+1:n]\R[k,k]
  R[k,k+1:n] = T(u)
  A[k+1:n,k+1:n] = A[k+1:n,k+1:n] - uT(u)
end
```

where resulting R satisfies $A = R^T R$.

solving $Ax = b$ for symmetric positive-definite is like LU factorization, $A = R^T R \Rightarrow R^T c = b \Rightarrow Rx = c$.

also,

- cholesky without block multiplication [@mathsresource](https://).

example 28

find cholesky factorization of

$$\begin{bmatrix} 4 & -2 & 2 \\ -2 & 2 & -4 \\ 2 & -4 & 11 \end{bmatrix}.$$

the top row of R is $R_{11} = \sqrt{a_{11}} = 2, R_{1,2:3} = [-2 \quad 2]/R_{11} = [-1 \quad 1]$:

$$R = \left[\begin{array}{c|cc} 2 & -1 & -1 \\ \hline & & \\ & & \end{array}\right].$$

subtract outer product $uu^T = \begin{bmatrix} -1 \\ 1 \end{bmatrix} [-1 \quad 1]$ from lower principal 2×2 submatrix $A_{2:3,2:3}$ of A leaves

$$R = \left[\begin{array}{c|cc} & & \\ \hline & 2 & -4 \\ & -4 & 11 \end{array}\right] - \left[\begin{array}{c|cc} & & \\ \hline & 1 & -1 \\ & -1 & 1 \end{array}\right] = \left[\begin{array}{c|cc} & & \\ \hline & 1 & -3 \\ & -3 & 10 \end{array}\right].$$

repeat the same steps on 2×2 submatrix to find $R_{22} = 1, R_{23} = \frac{-3}{1} = -3$

$$R = \left[\begin{array}{c|c|c} 2 & -1 & -1 \\ \hline & 1 & -3 \\ \hline & & \end{array}\right].$$

the lower 1×1 principal submatrix of A is $10 - (-3)(-3) = 1$, so $R_{33} = \sqrt{1}$. the cholesky factor of A is

$$R = \begin{bmatrix} 2 & -1 & -1 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}.$$

3 conjugate gradient method

the [conjugate gradient method](https://) ([hestens](https://) and [steifel](https://), 1952) (and preconditioners) enhanced solving sparse matrix problems.

this method tracks down the solution of a positive-definite $n \times n$ linear system by successively locating and eliminating the n orthogonal components of error, one by one. the complexity of the algorithm is minimized by using the directions established by pairwise orthogonal residual vectors.

got that? great, the test is tomorrow.

the conjugate gradient method is based on inner product. ie, euclidean inner product is

- conjugate symmetric, $\langle v, w \rangle = \overline{\langle w, v \rangle}$;
- linear, $\langle \alpha v + \beta w, u \rangle = \alpha \langle v, u \rangle + \beta \langle w, u \rangle$ for scalar α, β;
- positive-definite, $\langle v, v \rangle > 0$ if $v \neq 0$.

definition 15

let A be symmetric positive-definite $n \times n$ matrix. for two n-vectors v, w, define the A-**inner product**

$$\langle v, w \rangle_A = v^T A w.$$

vectors v, w are A-**conjugate** if $\langle v, w \rangle_A = 0$.

note that the new inner product inherits the properties of symmetry, linearity, and positive-definiteness from the matrix A. Because A is symmetric, so is the A-inner product:

$$\langle v, w \rangle_A = v^T A w = \langle v^T A w \rangle^T = w^T A v = \langle w, v \rangle_A.$$

the A-inner product is also linear, and positive-definiteness follows from the fact that if A is positive-definite, then

$$\langle v, v \rangle_A = v^T A v > 0, \quad v \neq 0.$$

conjugate gradient is a direct method and arrives at solution x of the symmetric positive-definite system $Ax = b$ with a finite loop.

algorithm

conjugate gradient method

```
x0 = initial guess
d0 = r0 = b - Ax0
for k = 0 : n-1
  if r[k] = 0, stop, end

  alpha = (T(r[k])·r[k])/(T(d[k])·A·d[k])
  x[k+1] = x[k] + alpha*d[k]
  r[k+1] = r[k] - alpha*A·d[k]
  beta = (T(r[k+1])·r[k+1])/(T(r[k])·r[k])
  d[k+1] = r[k+1] + beta*d[k]
end
```

discussion of algorithm

conjugate gradient updates three vectors: x_k, approximate solution at step k; vector r_k, residual of x_k where

$$\begin{aligned} Ax_{k+1} + r_{k+1} &= A(x_k + \alpha_k d_k) + r_k - \alpha_k A d_k \\ &= Ax_k + r_k \\ &\Downarrow \\ r_k &= b - Ax_k \end{aligned}$$

for all k; and d_k, new search direction of x_k. α **is the step-size and** β **is the correction for the next direction.**

this method succeeds bc each residual is orthogonal to previous residuals. in at most n steps, the method will exhaust orthogonal directions, reaching zero residual and correct solution.

to accomplish orthogonality among residuals relies on choosing pairwise conjguate of search direction d_k.

wrt α_k, β_k, directions d_k is chosen from vector space span of the previous residuals, as seen inductively from the last line of the pseudocode. to ensure that the next residual is orthogonal to all past residuals, α_k is chosen so that the new residual r_{k+1} is orthogonal to the direction d_k:

$$x_{k+1} = x_k + \alpha_k d_k$$

$$b - Ax_{k+1} = b - Ax_k - \alpha_k Ad_k$$

$$r_{k+1} = r_k - \alpha_k Ad_k$$

$$0 = d_k^T r_{k+1} = d_k^T r_k - \alpha_k d_k^T Ad_k$$

$$\alpha_k = \frac{d_k^T r_k}{d_k^T Ad_k}.$$

d_{k-1} is orthogonal to r_k, so

$$d_k - r_k = \beta_{k-1} d_{k-1}$$

$$r_k^T d_k - r_k^T r_k = 0$$

which justifies rewriting $r_k^T d_k = r_k^T r_k$ for α_k of algorithm. then coefficient β_k is chosen for pairwise A-conjugacy of the d_k:

$$d_{k+1} = r_{k+1} + \beta_k d_k$$

$$0 = d_k^T Ad_{k+1} = d_k^T Ar_{k+1} + \beta_k d_k^T Ad_k$$

$$\beta_k = -\frac{d_k^T Ar_{k+1}}{d_k^T Ad_k}.$$

the expression for β_k can be rewritten in the simpler form as in the algorithm.

theorem 16 below verifies that all r_k produced by the conjugate gradient iteration are orthogonal to one another. bc they are n-dimensional vectors, at most n of the r_k are pairwise orthogonal, so either r_n or previous r_k must be zero, solving $Ax = b$. therefore, after at most n steps, conjugate gradient arrives at a solution. in theory, the method is a direct, not an iterative, method.

example 29

solve system using conjugate gradiant,

$$\begin{bmatrix} 2 & 2 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 6 \\ 3 \end{bmatrix}.$$

$$x_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, r_0 = d_0 = \begin{bmatrix} 6 \\ 3 \end{bmatrix}$$

$$\alpha = \frac{\begin{bmatrix} 6 \\ 3 \end{bmatrix}^T \begin{bmatrix} 6 \\ 3 \end{bmatrix}}{\begin{bmatrix} 6 \\ 3 \end{bmatrix}^T \begin{bmatrix} 2 & 2 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} 6 \\ 3 \end{bmatrix}} = \frac{45}{6 \cdot 18 + 3 \cdot 27} = \frac{5}{21}$$

$$x_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \frac{5}{21} \begin{bmatrix} 6 \\ 3 \end{bmatrix} = \begin{bmatrix} \frac{10}{7} \\ \frac{5}{7} \end{bmatrix}$$

$$r_1 = \begin{bmatrix} 6 \\ 3 \end{bmatrix} - \frac{5}{21} \begin{bmatrix} 18 \\ 27 \end{bmatrix} = 12 \begin{bmatrix} \frac{1}{7} \\ -\frac{2}{7} \end{bmatrix}$$

$$\beta_0 = \frac{r_1^T r_1}{r_0^T r_0} = \frac{144 \cdot \frac{5}{49}}{36 + 9} = \frac{16}{49}$$

$$d_1 = 12 \begin{bmatrix} \frac{1}{7} \\ -\frac{2}{7} \end{bmatrix} + \frac{16}{49} \begin{bmatrix} 6 \\ 3 \end{bmatrix} = \begin{bmatrix} \frac{180}{49} \\ -\frac{120}{49} \end{bmatrix}$$

$$\alpha_1 = \frac{\begin{bmatrix} \frac{12}{7} \\ -\frac{24}{7} \end{bmatrix}^T \begin{bmatrix} \frac{12}{7} \\ -\frac{24}{7} \end{bmatrix}}{\begin{bmatrix} \frac{180}{49} \\ -\frac{120}{49} \end{bmatrix}^T \begin{bmatrix} 2 & 2 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} \frac{180}{49} \\ -\frac{120}{49} \end{bmatrix}} = \frac{7}{10}$$

$$x_2 = \begin{bmatrix} \frac{10}{7} \\ \frac{5}{7} \end{bmatrix} + \frac{7}{10} \begin{bmatrix} \frac{180}{49} \\ -\frac{120}{49} \end{bmatrix} = \begin{bmatrix} 4 \\ -1 \end{bmatrix}$$

$$r_2 = 12 \begin{bmatrix} \frac{1}{7} \\ -\frac{2}{7} \end{bmatrix} - \frac{7}{10} \begin{bmatrix} 2 & 2 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} \frac{180}{49} \\ -\frac{120}{49} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\Downarrow$$

$$r_2 = b - Ax_2 = 0 \Rightarrow x_2 = \begin{bmatrix} 4 & -2 \end{bmatrix}.$$

theorem 16

let A be symmetric positive-definite $n \times n$ matrix and let $b \neq 0$ be vector. in conjugate gradient method, assume $r_k \neq 0$ for $k < n$ (if $r_k = 0 \sim$ equation solved). then for each $1 \leq k \leq n$,

a) the following subspaces of $\mathbb{R}^n$ are equal:

$$\langle x_1, \ldots, x_k \rangle = \langle r_0, \ldots, r_{k-1} \rangle = \langle d_0, \ldots, d_{k-1} \rangle;$$

b) residuals r_k are pairwise orthogonal: $r_k^T r_j = 0, j < k$;
c) directions d_k are pairwise A-conjugate: $d_k^T A d_j = 0, j < k$.

proof

(a) for $k = 1$, note $\langle x_1 \rangle = \langle d_0 \rangle = \langle r_0 \rangle$ bc $x_0 = 0$. by def $x_k = x_{k-1} + \alpha_{k-1} d_{k-1}$. this implies $\langle x_1, \ldots, x_k \rangle = \langle d_0, \ldots, d_{k-1} \rangle$. similarly, $d_k = r_k + \beta_{k-1} d_{k-1} \Rightarrow \langle r_0, \ldots, r_{k-1} \rangle = \langle d_0, \ldots, d_{k-1} \rangle$

when $k = 0$, self-evident. assume (b),(c) hold for k and prove (b),(c) for $k+1$. multiply def of r_{k+1} by r_j^T on left:

$$\underbrace{r_j^T}_{} r_{k+1} = \underbrace{r_j^T}_{} r_k - \frac{r_k^T r_k}{d_k^T A d_k} \cdot \underbrace{r_j^T}_{} A d_k.$$

if $j \leq k-1$, then $r_j^T r_k = 0$ by induction (b). bc r_j can be expressed as combination of $d_0, \ldots, d_j$, term $r_j^T A d_k = 0$ from induction (c) and (b) holds. if $j = k$, then $r_k^T r_{k+1} = 0$ from previous bc $d_k^T A d_k = r_k^T A d_k + \beta_{k-1} d_{k-1}^T A d_k = r_k^T A d_k$ using induction (c) proves (b).

with $r_j^T r_k = 0$ and previous $r_j^T r_{k+1}$ with $j = k+1$,

$$\frac{r_{k+1}^T r_{k+1}}{r_k^T r_k} = -\frac{r_{k+1}^T A d_k}{d_k^T A d_k}.$$

multiply def of d_{k+1} from left by $d_j^T A$,

$$\underbrace{d_j^T \cdot A}_{} \cdot d_{k+1} = \underbrace{d_j^T \cdot A}_{} \cdot r_{k+1} - \frac{r_{k+1}^T \cdot A \cdot d_k}{d_k^T \cdot A \cdot d_k} \cdot \underbrace{d_j^T \cdot A}_{} \cdot d_k.$$

if $j = k$, then $d_k^T A d_{k+1} = 0$ using the symmetry of A. If $j \leq k-1$, then $Ad_j = \frac{r_j - r_j + 1}{\alpha_j}$ (from def of r_{k+1}) is orthogonal to r_{k+1}, first term LHS is zero and second term is zero by induction, which proves (c). ■ i guess.

usw

while that was gnarly, conjugate gradient has advantages: no row operations, no triple loops. however, conjugate gradient has many more operations in general ~ n^3 for its n steps vs $\frac{n^3}{3}$ for gaussian elimination. however, if A is sparse and large enough that $\frac{n^3}{3}$ is not feasible, conjugate gradient run as iterative may provide a sufficient approximation, hooray.

but it didnt get play then bc of round-off error for ill-conditioned A. **preconditioning** (which improves the system) compensates for that now.

4 preconditioning

convergence of iterative methods can be accelerated by use of preconditioning. the idea is to reduce the effective condition number of the problem.

the preconditioned form of the $n \times n$ linear system $Ax = b$ is

$$M^{-1}Ax = M^{-1}b,$$

where M is an invertible $n \times n$ matrix called the **preconditioner**. an effective preconditioner reduces the condition number of the problem by attempting to invert A. conceptually: (1) M should be close to A and (2) simple to invert. (those are not compatible goals, btw.)

using $M = A$ brings the condition number to one but A is likely difficult to invert if options were sought. $M = I$ does not improve the condition number. so a middle choice like the **jacobi preconditioner** $M = D = diag(A)$ is fair game.

let $z_k = M^{-1}b - M^{-1}Ax_k = M^{-1}r_k$ be preconditioned residual. then

$$\alpha_k = \frac{(z_k, z_k)M}{(d_k, M^{-1}Ad_k)M}$$

$$x_{k+1} = x_k + \alpha d_k$$

$$z_{k+1} = z_k - \alpha M^{-1} A d_k$$

$$\beta_k = \frac{(z_{k+1}, z_{k+1})M}{(z_k, z_k)M}$$

$$d_{k+1} = z_{k+1} + \beta_k d_k,$$

$$(z_k, z_k)M = z_k^T M z_k = z_k^T r_k$$

$$(d_k, M^{-1}Ad_k)M = d_k^T A d_k$$

$$(z_{k+1}, z_{k+1})M = z_{k+1}^T M z_{k+1} = z_{k+1}^T r_{k+1}.$$

algorithm

preconditioned conjugate gradient method

```
x0 = initial guess
r0 = b - Ax0
d0 = z0 = inv(M)r0
for k = 0:n-1
  if rk = 0, stop, end
```

```
  alpha = (T(r[k])·z[k])/(T(d[k])·A·d[k])
  x[k+1] = x[k] + alpha·d[k]
  r[k+1] = r[k] - alpha·A·d[k]
  z[k+1] = inv(M)·r[k+1]
  beta = (T(r[k+1])·z[k+1])/(T(r[k])·z[k])
  d[k+1] = z[k+1] + beta·d[k]
end
```

to save operations, use back substitution with M^{-1} and not matrix multiplication.

other preconditioners

symmetric successive over-relaxation (SSOR)

$$M = (D - \omega L)D^{-1}(D + \omega U) = (I + \omega L D^{-1})(D + \omega U).$$

if $\omega = 1$, previous is **gauss-seidel preconditioner**.

if SSOR used, $z = M^{-1}v$ can be solved with two back substitutions

$$\begin{aligned}(I + \omega L D^{-1})c &= v \\ (D + \omega U)z &= c.\end{aligned}$$