§6.5. Descent methods and conjugate gradients (CG)

→ Descent methods, and the improved descent method (or the CG method) is a powerful technique for solving

$$\boxed{A\underset{\sim}{u} = f},$$

when A is SPD (symmetric positive definite), or a SND (symmetric negative definite).

→ We begin considering the basic method of descent method first and extend the concept to CG.

→ The descent method is based on solving a minimization problem of the fn $\phi : \mathbb{R}^m \to \mathbb{R}$ defined by

$$\boxed{\phi(u) = \frac{1}{2}u^T A u - u^T f} \quad \cdots \quad (1)$$

Ex. $m=2$, $u = (u_1, u_2)$, $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{bmatrix}$ → symmetric

$$\phi(u) = \phi(u_1, u_2)$$

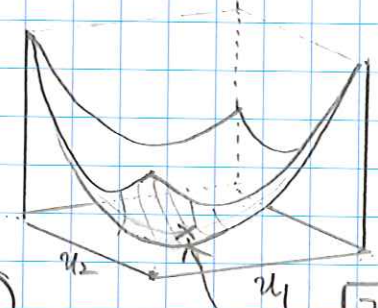
$$= \frac{1}{2}\begin{bmatrix} u_1 & u_2 \end{bmatrix}\begin{bmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} - \begin{bmatrix} u_1 & u_2 \end{bmatrix}\begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

$$= \frac{1}{2}\left(a_{11}u_1^2 + 2a_{12}u_1u_2 + a_{22}u_2^2\right) - (u_1 f_1 + u_2 f_2)$$

→ a quadratic fn of u_1 & u_2.

→ Since A is SPD;

$\exists!\ u^*$: a min. pt. that minimized ϕ.



($m=2$)

See Levegue p. 79.

$\boxed{\exists!\ u^*\ :\ \text{min. pt}}$

⇒ To find $U^* = (u_1^*, u_2^*)$, we look for

$$\begin{cases} 0 = \dfrac{\partial \phi}{\partial u_1} = a_{11} u_1^* + a_{12} u_2^* - f_1 \\ 0 = \dfrac{\partial \phi}{\partial u_2} = a_{21} u_1^* + a_{22} u_2^* - f_2 \end{cases} \quad \cdots (2)$$

⇒ (2) is equivalent to $AU^* = f$:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{bmatrix} \begin{bmatrix} u_1^* \\ u_2^* \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

⇒ Therefore, we conclude that if we find U^* for the minimization problem of (1), then we solve the linear system $AU = f$.

Rmk. In general, for any m, the above is also true & for

$$\begin{cases} U \in \mathbb{R}^m, \\ A \in \mathbb{R}^{m \times m} \text{ : SPD}, \end{cases}$$

we seek for U^* s.t

$$\boxed{0 = \nabla \phi(U^*) = AU^* - f}$$

Rmk. If A is indefinite (neither positive nor negative definite), then the eigenvalues of A are NOT all of the same sign, then $\exists\, U^*$ s.t $\nabla \phi(U^*) = 0$, but U^* is NOT a min or max pt, rather a saddle pt.

→ Any iteration method to find U^* will fail in this case.

→ Now our goal is to minimize $\phi(v)$, or find v^* s.t.

$$\boxed{\nabla\phi(v^*) = 0.} \qquad \phi(v) = \frac{1}{2}v^T A v - v^T f.$$

In order to solve $Av = f$.

→ The question on "how" to find v^* is then the next important topic.

→ Depending on the choice of a search direction from any initial pt. $v = (v_1, v_2)$ on $\phi(v)$, to $v^* = (v_1^*, v_2^*)$, one can achieve slow or fast convergence.

→ Two ways to chose the kth search direction s_k:

(i) $s_k = -\nabla\phi(v_k)$; the negative gradient

⇒ the method of steepest descent

(ii) $s_k = p_k \left(\neq -\nabla\phi(v_k)\right)$, where p_k is A-conjugate to all previous search directions, p_j, $j = 0, 1, \cdots, k-1$, i.e.,

Def. A-conjugate: $\boxed{p_k^T (A p_j) = 0}$, $j = 0, 1, \cdots, k-1$.

⇒ Conjugate gradient (CG) method.

Note. Let's denote "k" to be our subindex for the k^{th} iteration from now.

→ Once we set our search direction, s_k, then we find a new pt. v_{k+1} from v_k iteratively:

$$\boxed{v_{k+1} = v_k + \alpha_k s_k} \quad \cdots (3) \quad k = 0, 1, \cdots$$

where α_k is a scalar, satisfying:

$$\boxed{\min_{\alpha \in \mathbb{R}} \phi(v_{k+1}) = \min_{\alpha \in \mathbb{R}} \phi(v_k + \alpha_k s_k)} \quad \cdots (4)$$

→ Here, [(i) $\alpha_k > 0$ → not yet converged
(ii) $\alpha_k = 0$ → converged.

→ To solve (4), we compute the derivative of ϕ w.r.t α and set it to zero.

→ To take such derivative, let's first write ϕ as a continuous fcn of α (i.e, removing the discrete "k" for now):

$$\phi(v + \alpha s) = \frac{1}{2}(v+\alpha s)^T A (v+\alpha s) - (v+\alpha s)^T f$$
$$= \left(\frac{1}{2} v^T A v - v^T f\right) + \alpha(s^T A v - s^T f) + \frac{1}{2}\alpha^2 s^T A s$$

→ $\frac{d}{d\alpha}\phi(v+\alpha s) = s^T A v - s^T f + \alpha s^T A s = 0$

→ $$\boxed{\alpha = \frac{s^T(f - Av)}{s^T A s}} \quad \cdots (5)$$

→ Note here that, in iteration, we get

$\nabla\phi(v_k) = A v_k - f \equiv -r_k$, and we set] 9/27/16

Def $\boxed{r_k = f - A v_k}$: the residual vector ... (6)

(A) Steepest Descent

→ For the steepest descent method, we set

$$\boxed{s_k = -\nabla\phi(U_k)} \quad ⑦$$: the direction that always points in the direction of most rapid decrease of ϕ

$$= r_k$$

→ ⑤ + ⑥ + ⑦ ; $\alpha_k = \dfrac{r_k^T r_k}{r_k^T A r_k}$ --- ⑧

③ ; $U_{k+1} = U_k + \alpha_k r_k$

→ Note in ⑧, there are two matrix-vector multiplications

(i) $r_k = f - AU_k$, &

(ii) Ar_k

→ We can reduce them into one by replacing AU_k with Ar_{k-1}

$$r_k = f - AU_k$$
$$= f - A(U_{k-1} + \alpha_{k-1} r_{k-1})$$
$$= r_{k-1} - \alpha_{k-1} A r_{k-1} \quad \text{--- ⑨}$$

→ From ⑨, we removed the need to compute AU_k, but just kept Ar_k.

Algorithm

Choose a guess U_0

$r_0 = f - A U_0$ (initial residual)

for $k = 1, 2, \cdots$

$w_{k-1} = A r_{k-1}$ (the only matrix-vector multiplication)

$\alpha_{k-1} = \dfrac{r_{k-1}^T r_{k-1}}{r_{k-1}^T w_{k-1}}$ (compute search parameter)

$U_k = U_{k-1} + \alpha_{k-1} r_{k-1}$ (update soln)

$r_k = r_{k-1} - \alpha_{k-1} w_{k-1}$ (compute new residual) = new search direction

If $\|r_k\| < \varepsilon$, ε = tolerence, then

stop

else

continue

end if

endfor

Performance

(i) slow

(ii) See Lelegue (Figures 4.3, 4.4)

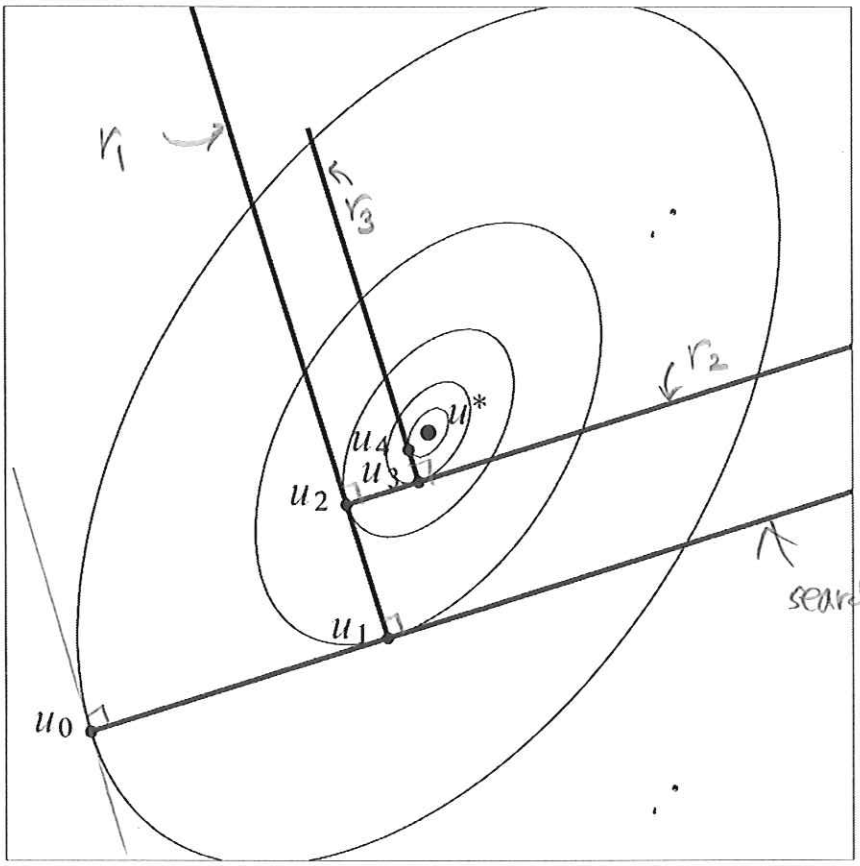


Figure 4.3. *Several iterates of the method of steepest descent in the case $m = 2$. The concentric ellipses are level sets of $\phi(u)$.*

Rmk $r_0, r_1, r_2, \ldots$ the search directions are always orthogonal to the contour lines.

→ We move along the direction of the gradient to the pt. where $\phi(v)$ is minimized along the line.

→ $r_0 \perp r_1$, $r_1 \perp r_2$, $\ldots$, $r_{k-1} \perp r_k$, etc.

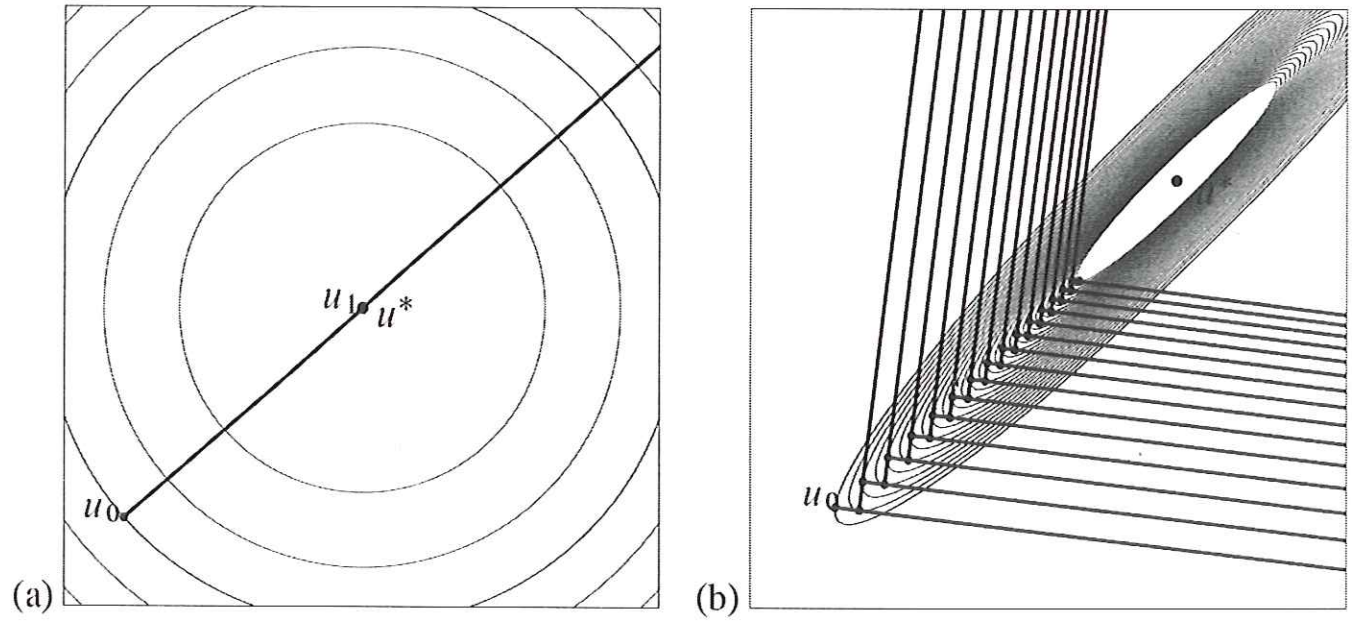


Figure 4.4. (a) *If A is a scalar multiple of the identity, then the level sets of $\phi(u)$ are circular and steepest descent converges in one iteration from any initial guess u_0.* (b) *If the level sets of $\phi(u)$ are far from circular, then steepest descent may converge slowly.*

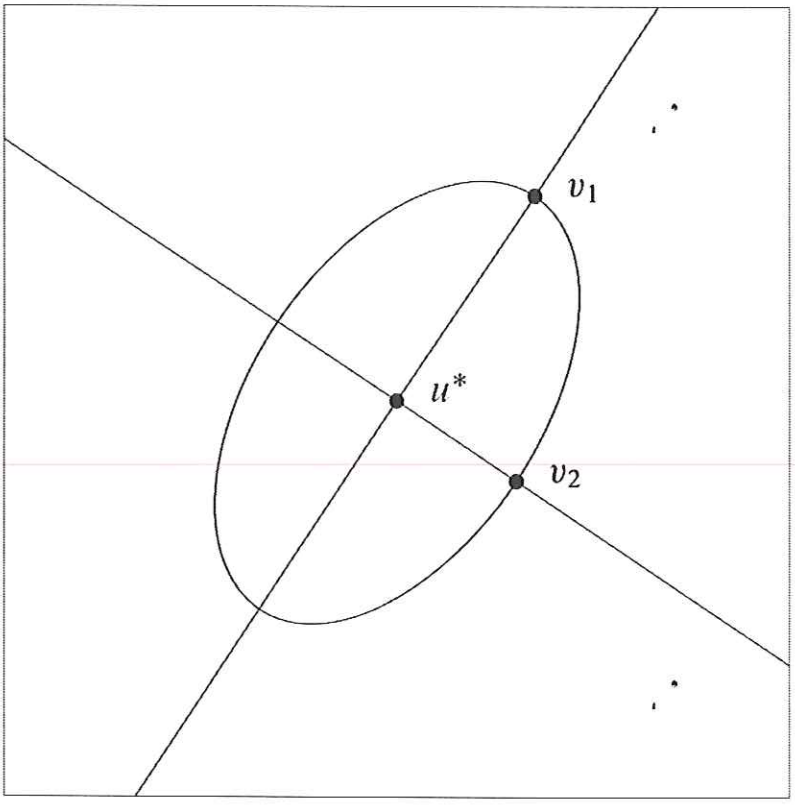


Figure 4.5. *The major and minor axes of the elliptical level set of $\phi(u)$ point in the directions of the eigenvectors of A.*

(B) Conjugate Gradient Method (CG)

→ The method of steepest descent can be generalized and accelerated if we choose a better search direction s_k, which will accelerate the iteration.

→ As already mentioned, the idea is to take

(i) $s_k = -\nabla\phi(u_k) = r_k$, &

(ii) $s_k = p_k$, where p_k satisfies $\underset{\sim}{A}$-conjugate

$$p_k^T(\underset{\sim}{A}p_j) = 0, \quad j = 0, \cdots, k-1,$$

which means that

p_k is orthogonal to all previous $\underset{\sim}{A}p_j$.

→ Note that $\underset{\sim}{A}$-conjugacy does NOT mean that, for $\underset{\sim}{A} \neq I$,

(i) $p_k \perp p_j$, $j = 0, \cdots, k-1$, but

(ii) $p_k \perp \underset{\sim}{A}p_j$, $j = 0, \cdots, k-1$,

→ If $\underset{\sim}{A} = I$, then this just means the conventional orthogonality of vectors p_k & p_j, $j = 0, \cdots, k-1$.

→ $\underset{\sim}{A}$-conjugacy is a generalized concept of orthogonality.

Rmk. (i) A very bad choice of p_k would be a direction orthogonal to $r_k = -\nabla\phi(v_k)$, which means p_k is tangent to the contour line of ϕ at v_k. (see Fig 4.3)

$\Rightarrow$ In this case, $\phi(v)$ increases along the tangent direction and the iteration will never converge.

(ii) We therefore want p_k s.t. $p_k^T r_k \neq 0$, then the new point v_{k+1} will be different from v_k, and $\phi(v_{k+1}) < \phi(v_k)$, $\forall k$. (See Fig. 4.6)

(iii) $p_k = -\nabla\phi(v_k)$ will reduce to the steepest descent.
→ We want better than this & iteration scheme: $v_{k+1} = v_k + \alpha_k p_k$.

(iv) Assuming v^* is the min. pt. that we want to obtain, the question is then "How do we find p_k without knowing v^* ?"

the idea is to observe that, since p_0 ($= r_0$) is tangent to the level set of ϕ at v_1,

$\Rightarrow p_0 \perp -\nabla\phi(v_1) = r_1 = f - A v_1$

$\Rightarrow 0 = p_0^T r_1 = p_0^T (f - A v_1)$, $f = A v^*$ (exact)

$= p_0^T A (v^* - v_1)$, $v^* - v_1 = \alpha_1 p_1$, for some $\alpha_1 \neq 0$.

$= p_0^T A \alpha_1 p_1$ --- (10)

⇒ (10) implies, written in another equivalent form, and after dividing by α_1

$$0 = \underbrace{p_0^T A p_1}_{\in \mathbb{R}} = \left(p_1^T A^T p_0\right)^T = \left(p_1^T A p_0\right)^T$$

(11) $$\boxed{p_1^T A p_0 = 0} \quad \cdots \quad (11)$$

⇒ (11) means that p_0 & p_1 are A-conjugate, and if so, using p_1 as a new search direction, the CG iteration converges in only <u>two steps</u>, which is, in general, true for $m = 2$.

⇒ If $m > 2$, then we can also show that <u>CG converges in, at most, m iterations.</u>

⇒ If A has n distinct eigenvalues, $A \in \mathbb{R}^m \times \mathbb{R}^m$, then we also see that <u>CG converges in at most n iterations, $n \leq m$.</u>

⇒ However, in practice, CG frequently "converges" to a sufficiently accurate appn to U^* <u>in much less</u> than n iterations.

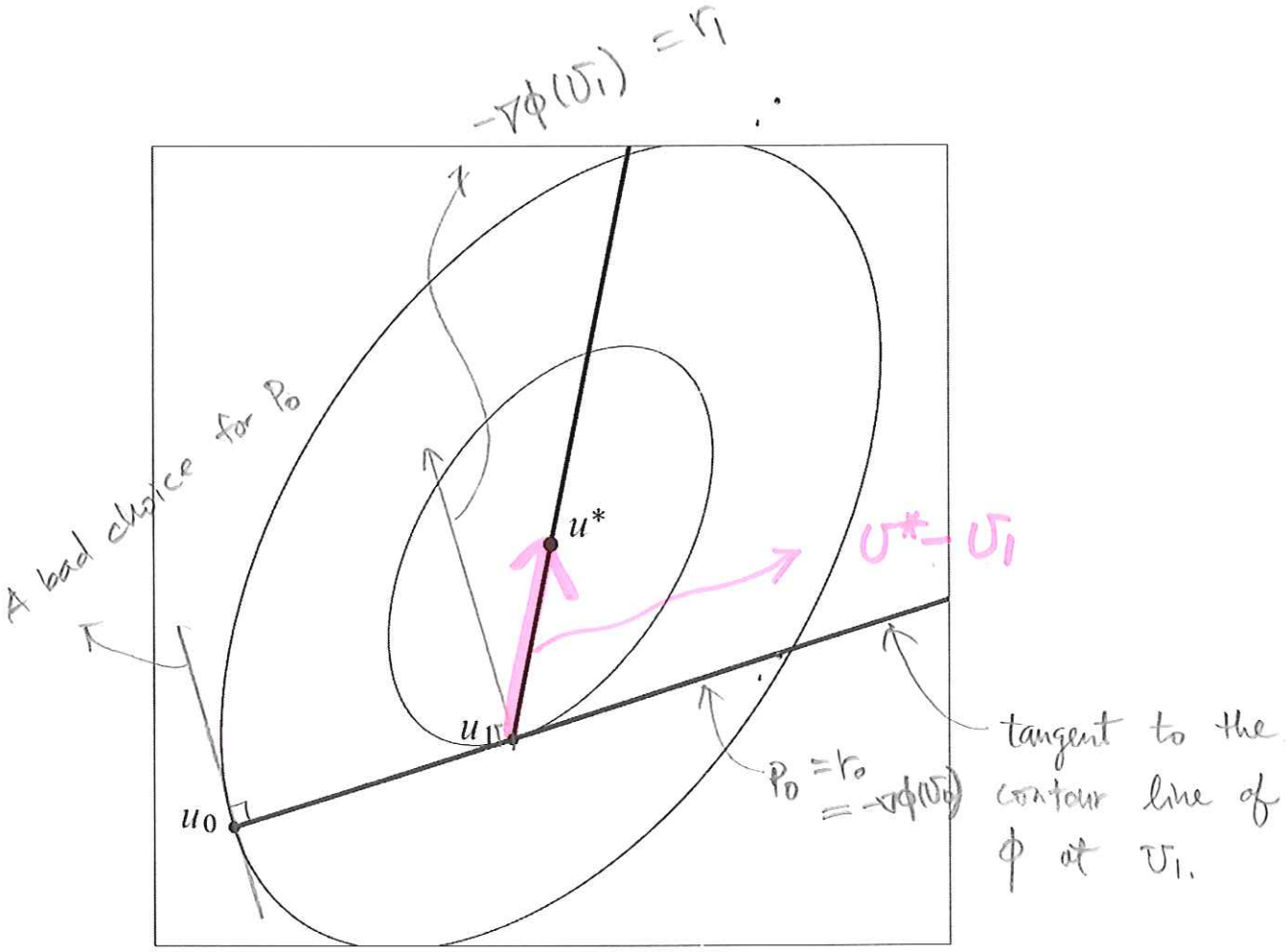


Figure 4.6. *The CG algorithm converges in two iterations from any initial guess u_0 in the case $m = 2$. The two search directions used are A-conjugate.*

Note that p_0 is tangent to the level set of ϕ at v_1,

$$\Rightarrow p_0 \perp -\nabla\phi(v_1) = r_1 = f - \underset{\sim}{A} v_1$$

$$\Rightarrow 0 = p_0^T r_1 = p_0^T (f - \underset{\sim}{A} v_1), \quad f = Av^*$$

$$= p_0^T \underset{\sim}{A} (v^* - v_1), \quad v^* - v_1 = \alpha p_1, \ \alpha \neq 0$$

$$= p_0^T \underset{\sim}{A} \alpha p_1$$

Algorithm

Choose initial guess v_0 (possibly $v_0 = 0$).

$r_0 = f - A v_0$ (initial residual)

$p_0 = r_0$ (initial search direction)

for $k = 1, 2, \cdots$

$w_{k-1} = A p_{k-1}$ (the only matrix-vector multiplication)

$\alpha_{k-1} = \dfrac{r_{k-1}^T r_{k-1}}{p_{k-1}^T w_{k-1}}$ (compute search parameter)

$v_k = v_{k-1} + \alpha_{k-1} p_{k-1}$ (update soln)

$r_k = r_{k-1} - \alpha_{k-1} w_{k-1}$ (compute new residual + search direction)

If $\|r_k\| < \varepsilon$, ε: tolerence, then

Stop

Else

$\beta_{k-1} = \dfrac{r_k^T r_k}{r_{k-1}^T r_{k-1}}$ (This is A-conjugacy $\Rightarrow p_k^T A p_j = 0$, $j = 0, 1, \cdots k-1$) ($\leftrightarrow$ See Thm 4.1 Lecture)

$p_k = r_k + \beta_{k-1} p_{k-1}$ (Compute new search direction)

Endif

Endfor

Compare this algorithm to the steepest descent algorithm presented on page 80. Up through the convergence check it is essentially the same except that the A-conjugate search direction p_{k-1} is used in place of the steepest descent search direction r_{k-1} in several places.

The final two lines in the loop determine the next search direction p_k. This simple choice gives a direction p_k with the required property that p_k is A-conjugate to all the previous search directions p_j for $j = 0, 1, \ldots, k-1$. This is part of the following theorem, which is similar to Theorem 38.1 of Trefethen and Bau [91], although there it is assumed that $u_0 = 0$. See also Theorem 2.3.2 in Greenbaum [39].

Theorem 4.1. *The vectors generated in the CG algorithm have the following properties, provided $r_k \neq 0$ (if $r_k = 0$, then we have converged):*

1. *p_k is A-conjugate to all the previous search directions,* i.e., *$p_k^T A p_j = 0$ for $j = 0, 1, \ldots, k-1$.*
2. *The residual r_k is orthogonal to all previous residuals, $r_k^T r_j = 0$ for $j = 0, 1, \ldots, k-1$.*
3. *The following three subspaces of $\mathbb{R}^m$ are identical:*

$$\begin{aligned} &\text{span}(p_0, p_1, p_2, \ldots, p_{k-1}), \\ &\text{span}(r_0, Ar_0, A^2 r_0, \ldots, A^{k-1} r_0), \\ &\text{span}(Ae_0, A^2 e_0, A^3 e_0, \ldots, A^k e_0). \end{aligned} \tag{4.43}$$

The subspace $\mathcal{K}_k = \text{span}(r_0, Ar_0, A^2 r_0, \ldots, A^{k-1} r_0)$ spanned by the vector r_0 and the first $k-1$ powers of A applied to this vector is called a *Krylov space* of dimension k associated with this vector.

The iterate u_k is formed by adding multiples of the search directions p_j to the initial guess u_0 and hence must lie in the affine spaces $u_0 + \mathcal{K}_k$ (i.e., the vector $u_k - u_0$ is in the linear space $\mathcal{K}_k$).

We have seen that the CG algorithm can be interpreted as minimizing the function $\phi(u)$ over the space $u_0 + \text{span}(p_0, p_1, \ldots, p_{k-1})$ in the kth iteration, and by the theorem above this is equivalent to minimizing $\phi(u)$ over the $u_0 + \mathcal{K}_k$. Many other iterative methods are also based on the idea of solving problems on an expanding sequence of Krylov spaces; see Section 4.4.

4.3.4 Convergence of conjugate gradient

The convergence theory for CG is related to the fact that u_k minimizes $\phi(u)$ over the affine space $u_0 + \mathcal{K}_k$ defined in the previous section. We now show that a certain norm of the error is also minimized over this space, which is useful in deriving estimates about the size of the error and rate of convergence.

Since A is assumed to be SPD, the A-norm defined by

$$\|e\|_A = \sqrt{e^T A e} \tag{4.44}$$

(C) Preconditioned CG (PCG).

→ Although it is a significant improvement over steepest descent, CG can still converge very slowly if

$\underline{\underline{A}}$ is ill-conditioned.

⇔ $\underline{\underline{A}}$ has a large condition number,

$\text{cond}(\underline{\underline{A}}) \overset{\text{def}}{=} \|A\| \cdot \|A^{-1}\|$.

⇔ $\underline{\underline{A}}$ becomes very close to singular and any numerical operations involving $\underline{\underline{A}}$ become very inaccurate, not to mention inverting it.

(Ex) $r_k = f - \underline{\underline{A}} U_k$ ← this becomes inaccurate

(Note; $\text{Cond}(\underline{\underline{A}}) \geq 1$, $\text{cond}(\underline{\underline{I}}) = 1$)

→ In this case, we preconditon $\underline{\underline{A}}$ by implicitly using $\underline{\underline{M}}^{-1}\underline{\underline{A}}$, instead of $\underline{\underline{A}}$, where

Given $AU_k = f$, we consider $M^{-1}AU_k = M^{-1}f$ instead.

(i) $\underline{\underline{M}}$ is a matrix for which $Mz_k = r_k$ is easily solved, with $r_k = f - \underline{\underline{A}} U_k$: residual, &

(ii) $\underline{\underline{M}}^{-1}\underline{\underline{A}}$ is relatively well-conditioned.

→ Typically, since CG only works for symmetric matrices, and since $\underline{\underline{M}}^{-1}\underline{\underline{A}}$ is not guranteed to be symmetric for CG, we in fact need to use

$L^{-1} A L$ instead of $M^{-1} A$, where $M = L L^T$.

→ However, fortunately, it turns out that the PCG algorithm can be suitably rearranged so that only M is used and the corresponding matrix L is not explicitly required. (See LeVeque)

→ The resulting PCG algorithm is essentially the same as the one for CG, but we solve the system

$$M z_k = r_k \quad \text{for} \quad z_k = M^{-1} r_k$$

$$= M^{-1}(f - A u_k)$$

$$= M^{-1} f - \underbrace{M^{-1} A u_k}_{\text{this is better than } A u_k}$$

and use z_k in place of r_k in the CG algorithm.

Algorithm : PCG

Choose initial guess $\underset{\sim}{U}_0$

$r_0 = f - \underset{\sim}{A} \underset{\sim}{U}_0$

Solve $\underset{\sim}{M} z_0 = r_0$ for z_0 $\quad (\Leftrightarrow z_0 = \underset{\sim}{M}^{-1} r_0)$

$p_0 = z_0$

for $k = 1, 2, \cdots$

$w_{k-1} = \underset{\sim}{A} p_{k-1} \quad (= \underset{\sim}{A} z_{k-1} = \underset{\sim}{A}\underset{\sim}{M}^{-1} r_{k-1})$ ← better than $\underset{\sim}{A} r_{k-1}$

$$\alpha_{k-1} = \frac{r_{k-1}^T r_{k-1}}{p_{k-1}^T w_{k-1}}$$

$U_k = U_{k-1} + \alpha_{k-1} p_{k-1}$

$r_k = r_{k-1} - \alpha_{k-1} w_{k-1}$

If $\|r_k\| < \varepsilon$, ε: tolerence, then

Stop

Else

Solve $\underset{\sim}{M} z_k = r_k$ for z_k $\quad \leftrightarrow \quad (z_k = \underset{\sim}{M}^{-1} r_k)$

$$\beta_{k-1} = \frac{z_k^T r_k}{r_{k-1}^T r_{k-1}}$$

$p_k = z_k + \beta_{k-1} p_{k-1}$

Endif

End for

→ The choice of an appropriate preconditioner $\underset{\sim}{M}$:

(i) depends on the trade-off between the gain in the convergence rate & the increased cost per iteration due to $\underset{\sim}{M}$.

(ii) is a big active area of research.

→ The most commonly used types of preconditioners :

(i) Diagonal (or Jacobi) : $\underset{\sim}{M} = \text{diag}(\underset{\sim}{A})$

(ii) Block diagonal (or Block Jacobi) :

$I = \{i \mid i = 1, \cdots, m\}$.

$I = I_1 \cup I_2 \cup \cdots \cup I_s$, I_k : disjoint each other

$$m_{ij} = \begin{cases} a_{ij}, & \text{if } i \& j \in I_k, \text{ for some } k, \\ 0, & \text{if } i \in I_p,\ j \in I_q,\ p \neq q. \end{cases}$$

$\underset{\sim}{M} = (m_{ij})_{ij}$

(iii) SSOR (symmetric SOR) :

$\underset{\sim}{A} = \underset{\sim}{L} + \underset{\sim}{D} + \underset{\sim}{L}^T$,

$\underset{\sim}{M} = (\underset{\sim}{D} + \underset{\sim}{L})\,\underset{\sim}{D}^{-1}\,(\underset{\sim}{D} + \underset{\sim}{L})^T$.

→ We can show that $\text{cond}(\underset{\sim}{M}^{-1}\underset{\sim}{A}) = O\left(\sqrt{\text{cond}(\underset{\sim}{A})}\right)$

(iv) incomplete Cholesky factorization,

(v) polynomial : $\underset{\sim}{M}^{-1}$ is taken to be a poly in $\underset{\sim}{A}$ that approximates $\underset{\sim}{A}^{-1}$.

(vi) Approximate inverse.