Note on Optimization Methods

Chunpai Wang

October 2015

1 Conjugate Gradient Method

1.1 The Linear Conjugate Gradient Method

1.1.1 Objective Function

The conjugate gradient method is an iterative method for solving a linear system of equations

$$Ax = b \tag{1}$$

where A is an $n \times n$ symmetric positive definite matrix. In general, the solution x lies at the intersection point of n hyperplanes.
The problem is equivalent to minimize the following convex quadratic function:

$$\min f(x) =: \frac{1}{2}x^T Ax - b^T x \tag{2}$$

And the gradient of $f(x)$ equals the residual of the linear system, that is,

$$\nabla f(x) = Ax - b = r(x) \tag{3}$$

so when $x = x_k$ we have

$$r_k = Ax_k - b \tag{4}$$

1.1.2 Conjugate Direction Method

Conjugacy of A Set of Vectors: A set of nonzero vectors $\{p_0, p_1, ..., p_l\}$ is said to be conjugate with respect to the symmetric positive definite matrix A (or called A-orthogonal) if

$$p_i^T Ap_j = 0, \quad \text{for all } \ i \neq j \tag{5}$$

Note that, any set of vectors satisfying this property is also linear independent or orthogonal.
The importance of conjugacy is that we can minimize $f(x)$ in n steps by successively minimizing it along the individual directions in a conjugate set.

Conjugate Direction Method: Given a starting point $x_0 \in \mathbb{R}^n$ and a set of conjugate directions $\{p_0, p_1, ..., p_{n-1}\}$, let's generate the sequence $\{x_k\}$ by setting:

$$x_{k+1} = x_k + \alpha_k p_k \tag{6}$$

where α_k is the one-dimensional minimizer of the quadratic function $f(\cdot)$ along $x_k + \alpha_k p_k$, that is

$$\alpha_k = arg\min_{\alpha} f(x_k + \alpha p_k) = -\frac{r_k^T p_k}{p_k^T Ap_k} \tag{7}$$

This is one of examples of exact line search method.

Theorem 1.1. *For any $x_0 \in \mathbb{R}^n$ the sequence $\{x_k\}$ generated by the conjugate direction algorithm(formula 6 and 7) converges to the solution x^* of the linear system (formula 1) in at most n steps.*

Proof. Let us first compute the α_k:

$$\begin{aligned}
f(x_k + \alpha p_k) &= \frac{1}{2}(x_k + \alpha p_k)^T A(x_k + \alpha p_k) - b^T(x_k + \alpha p_k) \\
&= \frac{1}{2}\alpha^2 p_k^T A p_k + \frac{1}{2}\alpha x_k^T A p_k + \frac{1}{2}\alpha p_k^T A x_k + \frac{1}{2}x_k^T A x_k - b^T x_k - b^T \alpha p_k \\
&= \frac{1}{2}\alpha^2 p_k^T A p_k + (\alpha x_k^T A p_k - \alpha b^T p_k) + (\frac{1}{2}x_k^T A x_k - b^T x_k) \\
&= \frac{1}{2}\alpha^2 p_k^T A p_k + \alpha(x_k^T A - b^T)p_k + f(x_k) \\
&= \frac{1}{2}\alpha^2 p_k^T A p_k + \alpha(A^T x_k - b)^T p_k + f(x_k) \\
&= \frac{1}{2}\alpha^2 p_k^T A p_k + \alpha(A x_k - b)^T p_k + f(x_k) \\
&= \frac{1}{2}\alpha^2 p_k^T A p_k + \alpha \nabla f(x_k)^T p_k + f(x_k)
\end{aligned}$$

Thus, let

$$f'(x_k + \alpha p_k) = \alpha^2 p_k^T A p_k + \nabla f(x_k)^T p_k = 0$$

we obtain:

$$\alpha_k = -\frac{\nabla f(x_k)^T p_k}{p_k^T A p_k} = -\frac{r_k^T p_k}{p_k^T A p_k}$$

Now note that, the set of conjugate direction $\{p_0, p_1, ..., p_{n-1}\}$ is given. Since the directions are linearly independent, they must span the whole space $\mathbb{R}^n$. Hence, we can write

$$x^* - x_0 = \sigma_0 p_0 + \sigma_1 p_1 + ... + \sigma_{n-1} p_{n-1} \tag{8}$$

for some choice of scalars σ_k. And since

$$x_n = x_0 + \alpha_0 p_0 + \alpha_1 p_1 + ... + \alpha_{n-1} p_{n-1} \tag{9}$$

generated by formula (6) and (7), the result is established if we can show that $\sigma_k = \alpha_k$.

By premultiplying this expression by $p_k^T A$ $(0 \le k \le n-1)$ and using the conjugacy property (formula 5), we obtain

$$p_k^T A(x^* - x_0) = p_k^T A(\sigma_0 p_0 + \sigma_1 p_1 + ... + \sigma_{n-1} p_{n-1}) = p_k^T A \sigma_k p_k \tag{10}$$

$$\sigma_k = \frac{p_k^T A(x^* - x_0)}{p_k^T A p_k} \tag{11}$$

If x_k is generated by formula (6) and (7), then we have

$$x_k = x_0 + \alpha_0 p_0 + \alpha_1 p_1 + ... + \alpha_{k-1} p_{k-1} \tag{12}$$

By premultiplying this expression by $p_k^T A$ and using the conjugacy property, we have that

$$p_k^T A(x_k - x_0) = 0 \quad \Rightarrow \quad p_k^T A x_k = p_k^T A x_0 \tag{13}$$

and therefore

$$p_k^T A(x^* - x_0) = p_k^T A(x^* - x_k) = p_k^T(b - A x_k) = -p_k^T r_k \tag{14}$$

$$\sigma_k = \frac{p_k^T A(x^* - x_0)}{p_k^T A p_k} = \frac{-p_k^T r_k}{p_k^T A p_k} = \alpha_k \tag{15}$$

□

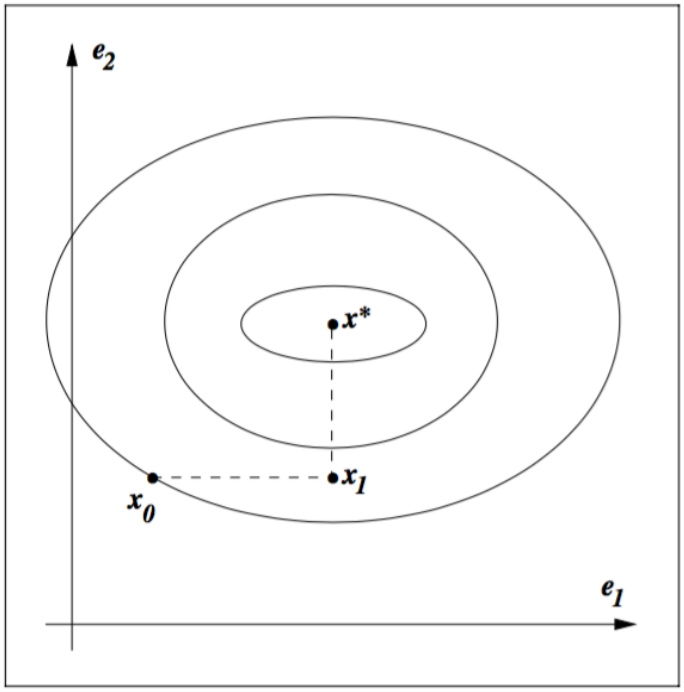
Interpretation:
If the matrix A in formula (2) is diagonal, the contours of the function $f(\cdot)$ are ellipses whose axes are aligned with coordinate directions.

Theorem 1.2 (Expanding Subspace Minimization)**.** *Let $x_0 \in \mathbb{R}^n$ be any starting point and suppose that the sequence $\{x_k\}$ is generated by the conjugate direction algorithm formula(6)(7). Then*

$$r_k^T p_i = 0, \quad \text{for } \ i = 0, 1, ..., k-1, \tag{16}$$

and x_k is the minimizer of $f(x) = \frac{1}{2}x^T A x - b^T x$ over the set

$$\{x | x = x_0 + span\{p_0, p_1, ..., p_{k-1}\}\} \tag{17}$$



[t]

Figure 1: Successive minimizations along the coordinate directions find the minimizer of a quadratic with a diagonal Hessian in n iterations. But if A is not diagonal, then it may not find the minimizer in n iterations. We have to use some tricks so as to make it diagonal to use this method(See textbook for details)

Proof. We begin by showing that a point $\hat{x}$ minimizes f over the set (17) if and only if

$$r(\hat{x})^T p_i = 0, \text{ for each } i = 0, 1, ..., k-1,$$

and then we will show that x_k satisfies $r(x_k)^T p_i = 0$, which implies that x_k is the minimizer over the set (17). First, minimizing $f(x)$ s.t. $x \in x_0 + span\{p_0, p_1, ..., p_{k-1}\}$ is equivalent to minimizing

$$h(\sigma) = f(x_0 + \sigma_0 p_0 + ... + \sigma_{k-1} p_{k-1}) = f(x_0 + P\sigma) \quad s.t. \;\; \sigma \in \mathbb{R}^k \tag{18}$$

where $\sigma = (\sigma_0, ..., \sigma_{k-1})^T$ and $P = (p_0, ..., p_{k-1}) \in \mathbb{R}^{n \times k}$.
Since f is strictly convex, $h(\sigma)$ is also a strictly convex quadratic, and it has a unique minimizer σ^* that satisfies

$$\nabla h(\sigma^*) = P^T \nabla f(x_0 + P\sigma^*) = 0 \tag{19}$$

This equation implies that

$$p_i^T \nabla f(x_0 + P\sigma^*) = p_i^T \nabla f(x_0 + \sigma_0^* p_0 + ... + \sigma_{k-1}^* p_{k-1}) = 0 \quad \text{for each} \;\; i = 0, 1, ..., k-1 \tag{20}$$

if $\hat{x}$ minimizes the f, then $\hat{x} = x_0 + \sigma_0^* p_0 + ... + \sigma_{k-1}^* p_{k-1}$, therefore $\nabla f(x_0 + \sigma_0^* p_0 + ... + \sigma_{k-1}^* p_{k-1}) = r(\hat{x})$ according to formula (3) and $r(\hat{x})^T p_i = 0$, as claimed.

We now use induction to show that x_k satisfies $r_k^T p_i = 0$. For the case $k = 1$, we have from the fact that $x_1 = x_0 + \alpha_0 p_0$ minimizes f along direction p_0 that $r_1^T p_0 = 0$, since $h(\alpha_0) = f(x_0 + \alpha_0 p_0)$ and $\nabla h(\alpha_0) = p_0 \nabla f(x_0 + \alpha_0 p_0) = p_0^T r_1 = r_1^T p_0 = 0$. Let us now make the induction hypothesis, namely, that $r_{k-1}^T p_i = 0$ for $i = 0, 1, ..., k-2$. According to formula 4 and 6, we have

$$r_k = r_{k-1} + \alpha_{k-1} A p_{k-1} \tag{21}$$

and according to formula 7, we have

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

so that

$$p_{k-1}^T r_k = p_{k-1}^T r_{k-1} + \alpha_{k-1} p_{k-1}^T A p_{k-1} = 0 \tag{22}$$

Meanwhile, for the other vectors $p_i, i = 0, 1, ..., k-1$, we have

$$p_i^T r_k = p_i^T r_{k-1} + \alpha_{k-1} p_i^T A p_{k-1} = 0 \tag{23}$$

where $p_i^T r_{k-1} = 0$ because of the induction hypothesis and $p_i^T A p_{k-1} = 0$ because of conjugacy of the vectors p_i. Therefore, we have shown $r_K^T p_i = 0$ for $i = 0, 1, ..., k-1$.

□

Remark: the proof shows an important property that the current residual r_k is orthogonal to all previous search directions, as expressed in formula (16).

1.1.3 Conjugate Gradient Method

Problem: for conjugate direction method,

- the conjugate direction set $\{p_0, p_1 m..., p_{n-1}\}$ is given at the beginning, and there are many ways to choose it.
- For instance, the eigen-vectors $v_1, v_2, ..., v_n$ of A are mutually orthogonal as well as conjugate with respect to A, so these could be used as the vector $\{p_0, p_1, ..., p_{n-1}\}$.
- too expensive to compute the complete set of eigenvectors.
- Gram-Schmidt orthogonalization process, but still requires to store the entire direction set.

Intuition: the conjugate gradient method is a conjugate direction method with a very special property:

- can compute a new vector p_k by using only previous vector p_{k-1}.
- it does not need to know all the previous elements $p_0, p_1, ..., p_{k-2}$ of the conjugate set
- p_k is automatically conjugate to all previous vectors.
- requires only little storage and computation.

In the conjugate gradient method,

$$p_k = -r_k + \beta_k p_{k-1}, \tag{24}$$

where $-r_k$ is the steepest descent direction for the function $f(\cdot)$ (by formula 3), p_{k-1} is the previous direction, and β_k is to be determined by the requirement that p_{k-1} and p_k must be conjugate with respect to A or called A-orthogonal (like the secant condition for Quasi-newton method).
By premultiplying formula(24) by $p_{k-1}^T A$ and imposing the condition $p_{k-1}^T A p_k = 0$, we find that

$$\begin{aligned} p_{k-1}^T A p_k &= p_{k-1}^T A(-r_k + \beta_k p_{k-1}) \\ 0 &= p_{k-1}^T A(-r_k) + p_{k-1}^T A \beta_k p_{k-1} \\ r_k^T A p_{k-1} &= \beta_k p_{k-1}^T A p_{k-1} \\ \beta_k &= \frac{r_k^T A p_{k-1}}{p_{k-1}^T A p_{k-1}} \end{aligned}$$

Algorithm 1: Conjugate Gradient Descent - Preliminary Version

Given starting point x_0; Set $r_0 \leftarrow Ax_0 - b,\ p_0 \leftarrow -r_0,\ k \leftarrow 0$; **repeat**

$$\alpha_k \leftarrow -\frac{r_k^T p_k}{p_k^T A p_k}; \tag{a1}$$

$$x_{k+1} \leftarrow x_k + \alpha_k p_k; \tag{a2}$$

$$r_{k+1} \leftarrow A x_{k+1} - b; \tag{a3}$$

$$\beta_{k+1} \leftarrow \frac{r_{k+1}^T A p_k}{p_k^T A p_k}; \tag{a4}$$

$$p_{k+1} \leftarrow -r_{k+1} + \beta_{k+1} p_k; \tag{a5}$$

$$k \leftarrow k + 1; \tag{a6}$$

until $r_k \neq 0$;

We will show that the directions $p_0, p_1, ..., p_{n-1}$ are indeed conjugate, which also implies above algorithm terminates in n steps.

Theorem 1.3. *Suppose that the k^{th} iterate generated by the conjugate gradient method is not the solution point x^*. The following 4 properties hold:*

$$r_k^T r_i = 0, \quad \text{for } i = 0, 1, ..., k-1, \tag{25}$$

$$span\{r_0, r_1, ..., r_k\} = span\{r_0, Ar_0, AAr_0, ..., A^k r_0\}, \tag{26}$$

$$span\{p_0, p_1, ..., p_k\} = span\{r_0, Ar_0, AAr_0, ..., A^k r_0\}, \tag{27}$$

$$p_k^T A p_i = 0, \quad for\ i = 0, 1, ..., k-1. \tag{28}$$

Therefore, the sequence $\{x_k\}$ converges to x^* in at most n steps.

Proof. (By hypothesis induction). The equations (26)(27) holds for k = 0, while (28) holds for $k = 1$. Assuming now that these equations are true for some k (the induction hypothesis), we show that they continue to hold for $k + 1$.

- To prove (26), we need to show that the set on the left-hand side is subset of the set of right-hand side, and vice versa. Because of the induction hypothesis, we have

  $$r_k \in span\{r_0, Ar_0, AAr_0, ..., A^k r_0\}, \quad p_k \in span\{r_0, Ar_0, AAr_0, ..., A^k r_0\},$$

  and thus

  $$Ap_k \in span\{Ar_0, AAr_0, AAAr_0, ..., A^{k+1} r_0\} \tag{29}$$

  By applying formula (21) $r_{k+1} = r_k + \alpha_k A p_k$, we find that

  $$r_{k+1} \in span\{r_0, Ar_0, AAr_0, ..., A^k r_0, A^{k+1} r_0\} \tag{30}$$

  By combining this expression with the induction hypothesis for formula (26), we conclude that

  $$span\{r_0, r_1, ..., r_k, r_{k+1}\} \subset span\{r_0, Ar_0, AAr_0, ..., A^k r_0, A^{k+1} r_0\}, \tag{31}$$

  To prove that the reverse inclusion holds as well, we use the induction hypothesis on formula (27) to deduce that

  $$A^k r_0 \in span\{r_0, Ar_0, AAr_0, ..., A^k r_0\} = span\{p_0, p_1, ..., p_k\} \tag{32}$$

  $$A(A^k r_0) \in span\{Ap_0, Ap_1, ..., Ap_k\} \tag{33}$$

  $$A^{k+1} r_0 \in span\{Ap_0, Ap_1, ..., Ap_k\} \tag{34}$$

  Since by formula (21) $r_{k+1} = r_k + \alpha_k A p_k$, we have $Ap_i = (r_{i+1} - r_i)/\alpha_i$ for $i = 0, 1, ..., k$, which means $span\{Ap_i\} = span\{r_i, r_{i+1}\}$, therefore it follows that

  $$span\{Ap_0, Ap_1, ..., Ap_k\} = span\{r_0, r_1, ..., r_{k+1}\}. \tag{35}$$

  $$A^{k+1} r_0 \in span\{r_0, r_1, ..., r_{k+1}\} \tag{36}$$

  By combining this expression with the induction hypothesis for formula(26), we find that

  $$span\{r_0, Ar_0, AAr_0, ..., A^k r_0, A^{k+1} r_0\} \subset span\{r_0, r_1, ..., r_k, r_{k+1}\} \tag{37}$$

  Therefore, formula(26) is proved.

- Now we prove (27) continues to hold for k+1

  $$\begin{aligned} span\{p_0, p_1, ..., p_k, p_{k+1}\} &= span\{p_0, p_1, ..., p_k, r_{k+1}\} && \text{by (24)} \\ &= span\{r_0, Ar_0, AAr_0, ..., A^k r_0, r_{k+1}\} && \text{by hypothesis for (27)} \\ &= span\{r_0, r_1, ..., r_k, r_{k+1}\} && \text{by (26)} \\ &= span\{r_0, Ar_0, AAr_0, ..., A^k r_0, A^{k+1} r_0\} && \text{by (26) for k+1} \end{aligned}$$

  Therefore, (27) is proved, and (26) and (27) also implies

  $$span\{r_0, r_1, ..., r_k\} = span\{p_0, p_1, ..., p_k\} \tag{38}$$

  .

- Next, we prove (28) with k replaced by $k + 1$
  According to (24), we multiply (24) by Ap_i, we obtain

  $$p_{k+1}^T A p_I = -r_{k+1}^T A p_i + \beta_{k+1} p_k^T A p_i \tag{39}$$

Since in algorithm we will update $\beta_{k+1} = \frac{r_{k+1}^T A p_k}{p_k^T A p_k}$, right-hand side of (38) equals to 0 when $i = k$. Note that our induction hypothesis for (28) implies the directions $p_0, p_1, ..., p_k$ are conjugate, so can apply Theorem 4.2 to deduce that

$$r_{k+1}^T p_i = 0, \qquad \text{for } \ i = 0, 1, ..., k. \tag{40}$$

Second, by repeatedly applying (27), we find that for $i = 0, 1, ..., k-1$, the following inclusion holds:

$$\begin{aligned} Ap_i \in A \ \text{span}\{r_0, Ar_0, ..., A^i r_0\} &= \text{span}\{Ar_0, A^2 r_0, ..., A^{i+1} r_0\} \\ &\subset \text{span}\{r_0, Ar_0, A^2 r_0, ..., A^{i+1} r_0\} \\ &= \text{span}\{p_0, p_1, ..., p_{i+1}\} \end{aligned} \tag{41}$$

By (40), we knew that r_{k+1} is orthogonal with p_i for $i = 0, 1, 2, ..., k$, hence, r_{k+1} is orthogonal with $\text{span}\{p_0, ..., p_{i+1}\}$ for $i = 0, 1, 2, ..., k-1$. And from (41), we got $Ap_i \in \text{span}\{p_0, ..., p_{i+1}\}$, which implies Ap_i is orthogonal with r_{k+1}, that is

$$r_{k+1}^T A p_i = 0, \qquad \text{for } \ i = 0, 1, ..., k-1, \tag{42}$$

so the first term in the right-hand side of (39) vanishes for $i = 0, 1, ..., k-1$. Because of the induction hypothesis for (28), the second term equals to 0 as well, and we conclude that

$$p_{k+1}^T A p_i = 0, \quad \text{for } \ i = 0, 1, ..., k.$$

Therefore, (28) is proved. That is, the direction set generated by the conjugate gradient method is indeed a conjugate direction set, so Theorem4.1 tells us that the algorithm terminates in at most n iterations.

- Finally, we prove (25). Because the direction set is conjugate, we have from (16) that $r_k^T p_i = 0$ for all $i = 0, 1, ..., k-1$ and any $k = 1, 2, ..., n-1$. By (24), we find that

$$p_i = -r_i + \beta_i p_{i-1} \tag{43}$$

$$r_i = \beta_i p_{i-1} - p_i \tag{44}$$

so that $r_i \in \text{span}\{p_i, p_{i-1}\}$ for all $i = 1, ..., k-1$. We conclude that r_k is orthogonal with r_i for all $i = 0, 1, ..., k-1$, that is $r_k^T r_i = 0$. And in algorithm, $p_0 = -r_0$, we note that $r_k^T r_0 = -r_k^T p_0 = 0$. Therefore, (25) is proved

□

Remark: The proof of this theorem relies on the fact that the first direction p_0 is the steepest descent direction $-r_0$; in fact, the result does not hold for other choices of p_0. By (25), since the gradients r_k are mutually orthogonal, the term "conjugate gradient method" is actually a misnomer.It is the search direction, not the gradients.

By (16) and multiplying (24) by r_k^T, we get $r_k^T p_k = -r_k^T r_k$. Second, we have $r_{k+1} = r_k + \alpha_k A p_k$, and $\alpha_k A p_k = r_{k+1} - r_k$, and premultiplying by r_{k+1}^T and p_k, we get $r_{k+1}^T A p_k = \alpha_k r_{k+1}^T r_{k+1}$ and $p_k^T A p_k = r_k^T r_k$.

Algorithm 2: Conjugate Gradient Descent

Given starting point x_0; Set $r_0 \leftarrow Ax_0 - b,\ p_0 \leftarrow -r_0,\ k \leftarrow 0$; **repeat**

$$\alpha_k \leftarrow \frac{r_k^T r_k}{p_k^T A p_k}; \tag{a1}$$

$$x_{k+1} \leftarrow x_k + \alpha_k p_k; \tag{a2}$$

$$r_{k+1} \leftarrow r_k + \alpha_k A p_k; \tag{a3}$$

$$\beta_{k+1} \leftarrow \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}; \tag{a4}$$

$$p_{k+1} \leftarrow -r_{k+1} + \beta_{k+1} p_k; \tag{a5}$$

$$k \leftarrow k + 1 \tag{a6}$$

until $r_k \neq 0$;

- The advantage is at any given point in this algorithm we never need to know the vectors x, r, and p for more than the last two iterations.
- In theory, convergence in at most n steps
- In practice, due to rounding errors, CG method can take $>> n$ steps or fail
- with luck (good spectrum of A), good approximation in $<< n$ steps
- attractive if matrix-vector products are inexpensive
- The CG methods is recommended only for large problems; otherwise, Gaussian elimination or other factorization algorithms such as the singular value decomposition are to be preferred, since they are less sensitive to rounding errors.

1.1.4 Rate of Convergence

- In exact arithmetic, the conjugate gradient descent will terminate at the solution at most n iterations.
- When the distribution of the eigenvalues of A has certain favorable features, the algorithm will identify the solution in many fewer than n iterations.
- Associated with theorem 4.2 expanding subspace minimization property

Krylov Subspace: Krylov subspace of degree k for r_0 can be expressed as:

$$\mathbb{K}(r_0, k) =: \text{span}\{r_0, Ar_0, ..., A^k r_0\}. \tag{45}$$

From (a2) in algorithm 3 and (27), we have that

$$\begin{aligned} x_{k+1} &= x_0 + \alpha_0 p_0 + ... + \alpha_k p_k \\ &= x_0 + \gamma r_0 + \gamma_1 A r_0 + ... + \gamma_k A^k r_0 \end{aligned} \tag{46}$$

for some constants γ_i. Now we express it in following way,

$$x_{k+1} = x_0 + P_k^*(A) r_0, \tag{47}$$

which we define

$$P_k^*(A) = \gamma_0 I + \gamma_1 A + ... + \gamma_k A^k \tag{48}$$

$P_k^*(\cdot)$ is a polynomial of degree k with coefficients $\gamma_0, \gamma_1, ..., \gamma_k$. Like any polynomials, P_k^* can take either a scalar or a square matrix as its argument.
Recall that quadratic norm (weighted Frobenius norm) used in steepest descent method:

$$||z||_A^2 = z^T A z$$

We now show among all possible methods whose first k steps are restricted to the Krylov subspace $\mathbb{K}(r_0, k)$, algorithm 3 does the best job of minimizing the distance to the solution after k steps, when this distance is measured by the weighted $||\cdot||_A$.

- Using the quadratic norm and the definition of f, and the fact that x^* minimizes f, it is easy to show that
$$\frac{1}{2}||x - x^*||_A^2 = \frac{1}{2}(x - x^*)^T A (x - x^*) = f(x) - f(x^*) \tag{49}$$
- Theorem 4.2 states that x_k minimizes f over the set $x_0 + \text{span}\{p_0, p_1, ..., p_{k-1}\}$, which is the same as x_{k+1} minimizes f over the set $x_0 + \text{span}\{p_0, p_1, ..., p_k\} = x_0 + \text{span}\{r_0, Ar_0, ..., A^k r_0\}$.
- We know that x^* minimizes f, and x_{k+1} minimizes f over set $x_0 + \text{span}\{r_0, Ar_0, ..., A^k r_0\}$, therefore x_{k+1} minimizes $\frac{1}{2}||x - x^*||_A^2 = f(x) - f(x^*)$ over set $x_0 + \text{span}\{r_0, Ar_0, ..., A^k r_0\}$.
- By (47), $x_{k+1} = x_0 + P_k^*(A) r_0$. We are going to find the optimal of P_k so as to minimize the $||x_{k+1} - x^*||_A^2$ that is
$$\min_{P_k} ||x_0 + P_k(A) r_0 - x^*||_A^2 \tag{50}$$
and the optimal P_k is just P_k^*. Therefore, the polynomial P_k^* generated by the CG method is optimal with respect to quadratic norm.

We exploit this optimality property repeatedly in the remainder of the section.

- Since
$$r_0 = Ax_0 - b = Ax_0 - Ax^* = A(x_0 - x^*), \tag{51}$$
we have that
$$\begin{aligned} x_{k+1} - x^* &= x_0 + P_k^*(A) r_0 - x^* \\ &= x_0 - x^* + P_k^*(A)(A(x_0 - x^*)) \quad (52) \\ &= [I + P_k^*(A) A](x_0 - x^*) \quad (53) \end{aligned}$$

- Let $0 < \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ be the eigenvalues of A, and let $v_1, v_2, ..., v_n$ be the corresponding orthonormal eigenvectors, so that

$$A = \sum_{i=1}^{n} \lambda_i v_i v_i^T, \quad \text{with} \quad v_i^T v_i = 1, v_i^T v_j = 0 \text{ for } i \neq j \tag{54}$$

  Since the eigenvectors can span the whole space $\mathbb{R}^n$, we can write

$$x_0 - x^* = \sum_{i=1}^{n} \delta_i v_i \tag{55}$$

  for some coefficients δ_i.

- $$Av_i = \lambda_i v_i, \tag{56}$$

$$\Rightarrow P_k(A)v_i = P_k(\lambda_i)v_i, \qquad i = 1, 2, ..., n \tag{57}$$

  for any polynomial P_k.

- By substituting (55) into (53), we have

$$\begin{aligned}
x_{k+1} - x^* &= [I + P_k^*(A)A] \sum_{i=1}^{n} \delta_i v_i && (58)\\
&= \sum_{i=1}^{n} [v_i + P_k^*(A)Av_i]\delta_i && (59)\\
&= \sum_{i=1}^{n} [v_i + P_k^*(A)\lambda_i v_i]\delta_i && (60)\\
&= \sum_{i=1}^{n} [v_i + \lambda_i P_k^*(A)v_i]\delta_i && (61)\\
&= \sum_{i=1}^{n} [v_i + \lambda_i P_k^*(\lambda_i)v_i]\delta_i && (62)\\
&= \sum_{i=1}^{n} [1 + \lambda_i P_k^*(\lambda_i)]\delta_i v_i && (63)
\end{aligned}$$

- By using (54), we know that

$$\begin{aligned}
||z||_A^2 = z^T A z &= z^T (\sum_{i=1}^{n} \lambda_i v_i v_i^T) z && (64)\\
&= \sum_{i=1}^{n} \lambda_i (z^T v_i v_i^T z) && (65)\\
&= \sum_{i=1}^{n} \lambda_i (v_i^T z)^T (v_i^T z) && (66)\\
&= \sum_{i=1}^{n} \lambda_i (v_i^T z)^2 && (67)
\end{aligned}$$

and apply it to $z = x_{k+1} - x^*$, we get

$$||x_{k+1} - x^*||_A^2 = \sum_{i=1}^{n} \lambda_i [v_i^T (x_{k+1} - x^*)]^2 \tag{68}$$

$$= \sum_{i=1}^{n} \lambda_i [v_i^T (\sum_{i=1}^{n} (1 + \lambda_i P_k^*(\lambda_i)) \delta_i v_i)]^2 \tag{69}$$

$$= \sum_{i=1}^{n} \lambda_i [v_i^T (1 + \lambda_i P_k^*(\lambda_i)) \delta_i v_i)]^2 \tag{70}$$

$$= \sum_{i=1}^{n} \lambda_i [(1 + \lambda_i P_k^*(\lambda_i)) \delta_i v_i^T v_i)]^2 \tag{71}$$

$$= \sum_{i=1}^{n} \lambda_i [1 + \lambda_i P_k^*(\lambda_i)) \delta_i 1]^2 \qquad \text{(orthonormal eigenvectors property)}$$

$$= \sum_{i=1}^{n} \lambda_i [1 + \lambda_i P_k^*(\lambda_i))]^2 \delta_i^2 \tag{72}$$

$$\tag{73}$$

- By (50), we can convert the problem into

$$||x_{k+1} - x^*||_A^2 = \min_{P_K} \sum_{i=1}^{n} \lambda_i [1 + \lambda_i P_k^*(\lambda_i))]^2 \delta_i^2 \tag{74}$$

- By extracting the largest of the terms $[1 + \lambda_i P_k^*(\lambda_i))]^2$ from above expression, we obtain that

$$||x_{k+1} - x^*||_A^2 \leq \min_{P_K} \max_{1 \leq i \leq n} [1 + \lambda_i P_k^*(\lambda_i))]^2 \Big(\sum_{j=1}^{n} \lambda_j \delta_j^2 \Big) \tag{75}$$

$$= \min_{P_K} \max_{1 \leq i \leq n} [1 + \lambda_i P_k^*(\lambda_i))]^2 ||x_0 - x^*||_A^2 \tag{76}$$

where we have used the fact that, (by (55) and (67))

$$||x_0 - x^*||_A^2 = \sum_{j=1}^{n} \lambda_j (\delta_i v_i)^2 \tag{77}$$

$$= \sum_{j=1}^{n} \lambda_j (v_j^T \delta_j v_j)^2 \tag{78}$$

$$= \sum_{j=1}^{n} \lambda_j \delta_j^2 \tag{79}$$

- You can find that in equations (75) (76), there is a relation between $x_{k+1} - x^*$ and $x_0 - x^*$, and it allows us to quantify the convergence rate of the CG method by estimating the non-negative scalar quantity

$$\min_{P_k} \max_{1 \leq i \leq n} [1 + \lambda_i P_k(\lambda_i)]^2 \tag{80}$$

In other words, the polynomial P_k effects the convergence rate, and we need to search for a P_k to make this expression as small as possible.

Theorem 1.4. *If A has only r distinct eigenvalues, then the CG iteration will terminate at the solution in at most r iterations.*

Proof. We will show that the size of distinct eigenvalues will effect the polynomial P_k we are searching for in formula (80), and therefore effect the convergence rate of CG method.

- Suppose that the eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$ only exists r distint values $\theta_1 < \theta_2 < ... < \theta_n$. We define a polynomial $Q_r(\lambda)$ by

$$Q_r(\lambda) = \frac{(-1)^r}{\theta_1 \theta_2 \cdots \theta_r} (\lambda - \theta_1)(\lambda - \theta_2) \cdots (\lambda - \theta_r). \tag{81}$$

Note that, $Q_r(\lambda_i) = 0$ for $i = 1, 2, ..., n$ and $Q_r(0) = 1$.

- We can deduce that $Q_r(\lambda) - 1$ is a polynomial of degree r with a root at $\lambda = 0$, so by polynomial division, the function $\bar{P}_{r-1}$ defined by

$$\bar{P}_{r-1}(\lambda) = \frac{Q_r(\lambda) - 1}{\lambda} \tag{82}$$

  is a polynomial of degree $r - 1$.(Degree means the highest power of λ in polynomial)

- By setting $k = r - 1$ in (80), we have

$$\begin{aligned} 0 &\leq \min_{P_{r-1}} \max_{1 \leq i \leq n} [1 + \lambda_i P_{r-1}(\lambda_i)]^2 && (83)\\ &\leq \max_{1 \leq i \leq n} [1 + \lambda_i \bar{P}_{r-1}(\lambda_i)]^2 && (84)\\ &= \max_{1 \leq i \leq n} [1 + \lambda_i \frac{Q_r(\lambda_i) - 1}{\lambda_i}]^2 && (85)\\ &= \max_{1 \leq i \leq n} Q_r^2(\lambda_i) && (86)\\ &= 0 \end{aligned}$$

- Hence, the non-negative scale constant of (80) is zero for the value $k = r - 1$, so we have that (76) $||x_r - x^*||_A^2 = 0$, and therefore $x_r = x^*$, as claimed.

□

1.2 The Nonlinear Conjugate Gradient Method

1.2.1 Motivation

Algorithm 3 can be viewed as a minimization algorithm for the convex quadratic function f defined by (2). It is natural to ask whether we can adapt the approach to minimize general convex functions, or even general nonlinear functions f.

1.2.2 The Fletcher-Reeves Method

Extend the conjugate gradient method to nonlinear functions f (differentiable) by making 2 simple changes in algorithm 3

- for the step length α_k (which minimizes f along the search direction p_k), we need to perform a line search that identifies an approximate minimum of the nonlinear function f along p_k.
- the residual r (which is simply the gradient of f in algorithm 3) must be replaced by the gradient of the nonlinear objective f.

Algorithm 3: The Fletcher-Reeves Method

Given starting point x_0; Evaluate $f_0 = f(x_0), \nabla f_0 = \nabla f(x_0)$; Set $p_0 \leftarrow -\nabla f_0, k \leftarrow 0$; **repeat**

Compute α_k with line search and set $x_{k+1} = x_k + \alpha_k p_k$; Evaluate ∇f_{k+1}

$$\beta_{k+1}^{FR} \leftarrow \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k}; \tag{a1}$$

$$p_{k+1} \leftarrow -\nabla f_{k+1} + \beta_{k+1}^{FR} p_k; \tag{a2}$$

$$k \leftarrow k + 1; \tag{a3}$$

until $\nabla f_k \neq 0$ *or* $||\nabla f_k||_2 \geq \epsilon$;

Observations of FR Update

- First iteration is a gradient step; practical implementations restart the algorithm by taking a gradient step, for example, every n iterations.
- If we choose f to be strongly convex quadratic, and use exact line search to find $\alpha_k = arg\min_t f(x_k + tp_k)$, this algorithm reduces to the linear conjugate gradient method (algorithm 3).
- Good for large scale optimization problem, because each iteration only requires the evaluation of its objective function and gradient. No matrix operations are required for the step computation, and just few vectors of storage are required.
- In algorithm 3, it needs matrix-vector computation to get r_{k+1}, but here we only need to compute the gradient of $f(x_{k+1})$.
- The choice of line search parameter α is important, because it may effect that the search direction p_{k+1} in (a4) of algorithm 4 fails to be a descent direction.
- By taking the inner product of (a2) in algorithm 4 with the gradient vector $\nabla f_(k+1)$, we obtain

$$\nabla f_{k+1}^T p_{k+1} = -||\nabla f_{k+1}||^2 + \beta_{k+1}^{FR} \nabla f_{k+1}^T p_k \tag{87}$$

If the line search is exact, so that α_k is a local minimizer of f along the direction p_k, that is make derivative of $f(x_k + \alpha_k p_k) = f(x_{k+1})$ respect to α_k equal to 0, we get $\nabla f_{k+1}^T p_k = 0$. We find that (87) $\nabla f_{k+1}^T p_{k+1} < 0$, so that p_{k+1} is indeed a descent direction. If the line search is not exact, we need to require the step length α_k to satisfy the strong Wolfe conditions, which are

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k \tag{88}$$

$$|\nabla f(x_k + \alpha_k p_k)^T p_k| \leq -c_2 \nabla_k^T p_k \tag{89}$$

where $0 < c_1 < c_2 < 1/2$. (Backtracking line search).

1.2.3 The Polak-Ribier Method and Variants

There are many variants of the Fletcher-Reeves method that differ from each other mainly in the choice of the paramemter β_k.
Polak-Ribier:

$$\beta_k^{PR} = \frac{\nabla f_{k+1}^T(\nabla f_{k+1} - \nabla f_k)}{||\nabla f_k||^2} \tag{90}$$

Hestenes-Stiefel:

$$\beta_k^{HS} = \frac{\nabla f_{k+1}^T(\nabla f_{k+1} - \nabla f_k)}{(\nabla f_{k+1} - \nabla f_k)^T p_k} \tag{91}$$

Those formulas are equivalent for quadratic f and exact line search.

1.2.4 Applications in Optimization

Nonlinear conjugate gradient method

- extend linear CG method to nonquadratic functions
- local convergence similar to linear CG
- limited global convergence theory

Inexact and truncated Newton method

- use conjugate gradient method to compute (approximate) Newton step
- less reliable than exact Newton methods, but handle very large problems

1.2.5 Global Convergence Rate