

## Lecture # 19 The Conjugate Gradient Method

We wish to solve

$$A\mathbf{x} = \mathbf{b} \tag{1}$$

where  $A \in \mathbf{R}^{n \times n}$  is symmetric and positive definite (SPD). We then of  $n$  are being VERY LARGE, say,  $n = 10^6$  or  $n = 10^7$ . Usually, the matrix is also sparse (mostly zeros) and Cholesky factorization is not feasible.

When  $A$  is SPD, solving (1) is equivalent to finding  $\mathbf{x}^*$  such that

$$\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathbf{R}^n} \frac{1}{2} \mathbf{x}^T A \mathbf{x} - \mathbf{x}^T \mathbf{b}.$$

We let  $\phi(x) = \frac{1}{2} \mathbf{x}^T A \mathbf{x} - \mathbf{x}^T \mathbf{b}$ . Then the gradient of  $\phi$  is given by

$$\nabla \phi(x) = A\mathbf{x} - \mathbf{b}.$$

Thus  $\nabla \phi(\mathbf{x}^*) = 0$  if  $\mathbf{x}^*$  solves (1). As an exercise, you can show that if  $\mathbf{x} = \mathbf{x}^* + \mathbf{y}$  then

$$\phi(\mathbf{x}) = \phi(\mathbf{x}^*) + \frac{1}{2} \mathbf{y}^T A \mathbf{y}.$$

Since  $A$  is SPD,  $\frac{1}{2} \mathbf{y}^T A \mathbf{y} > 0$  for all  $\mathbf{y} \neq 0$  so  $\mathbf{x}^*$  is the unique minimum.

Our strategy for solving (1), will be to produce a sequence of iterates  $\{\mathbf{x}_k\}$  such that

$$\phi(\mathbf{x}_{k+1}) < \phi(\mathbf{x}_k)$$

and

$$\lim_{k \rightarrow \infty} \mathbf{x}_k = \mathbf{x}^*.$$

In fact, it is better if the iterates converge to  $\mathbf{x}^*$  in a finite number of steps. Our strategy will be to let

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_{k+1} \mathbf{p}_{k+1}$$

where  $\alpha_{k+1}$  is a step length and  $\mathbf{p}_{k+1}$  are directions that can be generated using matrix-vector multiplications with  $A$ .

If we look at

$$\phi(x_{k+1}) = \frac{1}{2} \mathbf{x}_{k+1}^T A \mathbf{x}_{k+1} - \mathbf{x}_{k+1}^T \mathbf{b}$$

$$\begin{aligned}
&= \frac{1}{2}(\mathbf{x}_k + \alpha_{k+1}\mathbf{p}_{k+1})^T A(\mathbf{x}_{k+1} + \alpha_{k+1}\mathbf{p}_{k+1}) - (\mathbf{x}_{k+1} + \alpha_{k+1}\mathbf{p}_{k+1})^T \mathbf{b} \\
&= \frac{1}{2}\mathbf{x}_k^T A\mathbf{x}_k - \mathbf{x}_k^T \mathbf{b} + \frac{1}{2}\alpha_{k+1}^2 \mathbf{p}_{k+1}^T A\mathbf{p}_{k+1} + \alpha_{k+1}\mathbf{p}_{k+1}^T A\mathbf{x}_k - \alpha_{k+1}\mathbf{p}_{k+1}^T \mathbf{b} \\
&= \phi(\mathbf{x}_k) + \frac{1}{2}\alpha_{k+1}^2 \mathbf{p}_{k+1}^T A\mathbf{p}_{k+1} - \alpha_{k+1}\mathbf{p}_{k+1}^T \mathbf{r}_k
\end{aligned}$$

where  $\mathbf{r}_k = -\nabla\phi(\mathbf{x}_k) = \mathbf{b} - A\mathbf{x}_k$  is the residual (and the negative gradient). If we differentiate with respect to  $\alpha_{k+1}$  and set to zero, the best choice of  $\alpha_{k+1}$  is

$$\alpha_{k+1} = \frac{\mathbf{p}_{k+1}^T \mathbf{r}_k}{\mathbf{p}_{k+1}^T A\mathbf{p}_{k+1}}. \quad (2)$$

Thus,  $\mathbf{p}_{k+1}$  cannot be orthogonal to  $\mathbf{r}_k$ . If we choose  $\mathbf{p}_{k+1} = \mathbf{r}_k$ , we get the method of steepest descent. Although this method converges, when we substitute  $\alpha_{k+1} = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{r}_k^T A\mathbf{r}_k}$  back into the above equations we get

$$\phi(\mathbf{x}_{k+1}) = \phi(\mathbf{x}_k) - \frac{1}{2}\|\mathbf{r}_k\|_2^4 / (\mathbf{r}_k^T A\mathbf{r}_k).$$

Thus, when  $\mathbf{r}_k$  gets small, steepest descent will slow down.

The idea of Hestenes and Stiefel (1952) was to choose directions  $\mathbf{p}_1, \dots, \mathbf{p}_k, \dots$ , such that

$$\mathbf{p}_j^T A\mathbf{p}_k = 0, \quad k \neq j. \quad (3)$$

Equation (3) states that the  $\mathbf{p}_j$ 's are *A-orthogonal* or *A-conjugate*. It also means that they are linearly independent, so we can construct no more than  $n$  of them. *Although the conjugate gradient algorithm terminates in  $n$  iterations in exact arithmetic, it does not in floating point arithmetic. In practice, that scarcely matters, since  $n$  is very large and we don't want to do that many iterations anyway!*

We choose

$$\mathbf{p}_1 = \mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$$

then let

$$\mathbf{p}_{k+1} = \mathbf{r}_k + \beta_k \mathbf{p}_k.$$

Enforcing  $\mathbf{p}_{k+1}^T A\mathbf{p}_k = 0$  yields

$$\beta_k = \frac{-\mathbf{r}_k^T A\mathbf{p}_k}{\mathbf{p}_k^T A\mathbf{p}_k}. \quad (4)$$

Let then let

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_{k+1} A \mathbf{p}_{k+1}.$$

Some simplifications occur that allows us to do only one matrix-vector multiplication with  $A$  per iteration.

Theorem 3, pp.238–239 show the following properties of the directions and the residuals.

$$\begin{aligned} \mathbf{r}_k^T \mathbf{r}_j &= 0, & k \neq j \\ \mathbf{p}_j^T A \mathbf{p}_k &= 0, & k \neq j \\ \mathbf{r}_k^T \mathbf{p}_j &= 0, & j < k \\ \mathbf{r}_k^T \mathbf{r}_k &= \mathbf{r}_k^T A \mathbf{p}_{k+1} \end{aligned}$$

*These relations do not hold in floating point arithmetic, in general. But the good behavior of the the conjugate gradient algorithm holds up for the most part.*

These results simplify the expressions for  $\beta_k$  and  $\alpha_{k+1}$  to

$$\beta_k = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{r}_{k-1}^T \mathbf{r}_{k-1}} \quad (5)$$

$$\alpha_{k+1} = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{p}_{k+1}^T A \mathbf{p}_{k+1}}. \quad (6)$$

This leads to an elegant, simple algorithm.

### Algorithm 1 (The Conjugate Gradient Algorithm)

$\mathbf{x}_0$  initial guess (usually 0).  
 $\mathbf{p}_1 = \mathbf{r}_0 = \mathbf{b} - A \mathbf{x}_0$ ;  
 $\mathbf{w} = A \mathbf{p}_1$ ;  
 $\alpha_1 = \mathbf{r}_0^T \mathbf{r}_0 / (\mathbf{p}_1^T \mathbf{w})$ ;  
 $\mathbf{x}_1 = \mathbf{x}_0 + \alpha_1 \mathbf{p}_1$  ;  
 $\mathbf{r}_1 = \mathbf{r}_0 - \alpha_1 \mathbf{w}$ ;  
 $k = 1$  ;  
**while**  $\|\mathbf{r}_k\|_2 > \epsilon$  (some tolerance)  
 $\beta_k = \mathbf{r}_k^T \mathbf{r}_k / (\mathbf{r}_{k-1}^T \mathbf{r}_{k-1})$ ;

$$\begin{aligned}
\mathbf{p}_{k+1} &= \mathbf{r}_k + \beta_k \mathbf{p}_k; \\
\mathbf{w} &= A\mathbf{p}_{k+1}; \\
\alpha_{k+1} &= \mathbf{r}_k^T \mathbf{r}_k / (\mathbf{p}_{k+1}^T \mathbf{w}); \\
\mathbf{x}_{k+1} &= \mathbf{x}_k + \alpha_{k+1} \mathbf{p}_{k+1}; \\
\mathbf{r}_{k+1} &= \mathbf{r}_k - \alpha_{k+1} \mathbf{w}; \\
k &= k + 1;
\end{aligned}$$

**end;**

$\mathbf{x} = \mathbf{x}_k;$

The conjugate gradient algorithm is essentially optimal in the following sense. We note that

$$\mathbf{x}_k = \mathbf{x}_0 + \sum_{j=1}^k \alpha_j \mathbf{p}_j = \mathbf{x}_0 + P_k \mathbf{a}$$

where

$$\begin{aligned}
P_k &= (\mathbf{p}_1, \dots, \mathbf{p}_k) \\
\mathbf{a} &= (\alpha_1, \dots, \alpha_k)^T
\end{aligned}$$

One can show that

$$\mathbf{x}_k = \arg \min_{\mathbf{x} = \mathbf{x}_0 + P_k \mathbf{c}} \frac{1}{2} \mathbf{x}^T A \mathbf{x} - \mathbf{x}^T \mathbf{b}.$$

If we let the  $A$ -norm be given by

$$\|\mathbf{w}\|_A = \sqrt{\mathbf{w}^T A \mathbf{w}}$$

then one can show that

$$\|\mathbf{x}_k - \mathbf{x}^*\|_A \leq 2 \|\mathbf{x}_0 - \mathbf{x}^*\|_A \left( \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^k$$

where

$$\kappa = \kappa_2(A) = \|A^{-1}\|_2 \|A\|_2.$$

Thus if  $A$  is well conditioned, the conjugate gradient method converges quickly. This bound is actually very pessimistic! However, if  $A$  is badly conditioned, conjugate gradient can converge slowly. That problem is discussed in the next lecture.